PROGRAMMEZ.COM

PROGRAMMEZ!

Mensuel n°182 - Février 2015

le magazine du développeur

Développeur : le plus beau métier du monde !

C++ HTML5 PHP JAVA C#

Pourquoi devenir développeur ?
Les salaires
Les développeurs témoignent !

Unity
Créer des jeux 2D / 3D

Android
Android Studio : Eclipse est mort !

Windows
Intégrer Cortana en quelques minutes

Outils
Visual Studio 2013 Community

Coding4Fun - Geek
Microsoft Band
Kinect for Windows : le nouveau SDK

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

M 04319 - 182 - F: 5,95 € - RD

Quelle interopérabilité entre mes différents fournisseurs Cloud ?

Avec Aruba Cloud,

vous avez l'assurance de ne pas être prisonnier d'un fournisseur. Nos services sont intégrés au **driver DeltaCloud** et compatibles **S3**. De plus, vous pouvez utiliser des formats standards d'images de machines virtuelles, **avec VHD et VMDK**, ainsi que des modèles personnalisés provenant éventuellement d'autres sources.

3 hyperviseurs

6 datacenters en Europe

APIs et connecteurs

70+ templates

Contrôle des coûts

Nous avons choisi Aruba Cloud car nous bénéficions d'un haut niveau de performance, à des coûts contrôlés et surtout car ils sont à dimension humaine, comme nous. Xavier Dufour - Directeur R&D - ITMP

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.

Si tu es fier (fière) d'être développeur (euse), frappe dans tes mains !

Cela fait 17 ans que Programmez ! existe, et, depuis le début, nous parlons du développeur. Et durant toutes ces années, nous sommes passés par tous les superlatifs et commentaires : il existe, il n'existe plus, il est fier, etc. Dingue, comment le développeur est devenu à la mode et visible partout... ??? !!! Depuis le début de cette médiatisation et des diverses initiatives pour l'apprentissage de la programmation à l'école, je suis à la fois excité, content et sceptique. Parfois j'ai même l'impression que rien n'a changé depuis mes premiers jobs dans le développement et les tests.

Développeur est un job en or pour celles et ceux qui en veulent, qui sont motivés, qui ont de la passion et la technologie dans le cerveau. Je l'ai toujours dit et je le dirai encore longtemps. Trop longtemps, notre métier a été dénigré, marginalisé par les entreprises et la société dans son ensemble. Le regard change en espérant que tout ceci ne soit pas seulement qu'un buzz médiatique. Il ne faut pas dégoûter les jeunes de la programmation avec des maths, de l'algo. La programmation c'est fun et passionnant.

Nous avons voulu frapper fort avec un gros dossier pour toi, gentil développeur et dire pourquoi développeur, c'est le plus beau métier du monde. Nous avons donné la parole à plusieurs développeurs pour qu'ils partagent leurs expériences, leurs passions, mais aussi les problèmes et les contraintes. Développeur, c'est aussi un métier exigeant. Nous avons aussi fait un large « récap » des salaires selon les profils et les technologies. Sans surprise, les écarts sont énormes.

Si je voulais résumer ce dossier : nous n'avons pas besoin d'armées de développeurs, mais de bons développeurs (la nuance est importante), et, comme me le disait Nicolas Sadirac (école 42), les compétences sont les mêmes que l'on soit dév mobile, dév embarqué, dév Web... ce qui va changer ce sont les compétences spécifiques et les contraintes du secteur.

Dans ce numéro 182, vous allez trouver de vrais morceaux de codes et de geeks : MS Band, Kinect pour Windows v2, JavaScript à la mode fonctionnelle, Wordpress, HTML 5, Android Studio, Cortana, des frameworks, de l'écologie numérique et du bon gros Unity.

Développeur un jour, développeur toujours !

10 Circuit Scribe

72 Coding4fun

5 Agenda

18 OForth : un nouveau langage

59 Android Studio, l'Eclipse killer

12 Microsoft Band

54 Intégrer Cortana en 5 mn

4 Tableau de bord

62 Wordpress, les tests

48 Wordpress 3e partie

36 Je débute avec... Unity

16 R&D : Epitech

6 Éco-conception

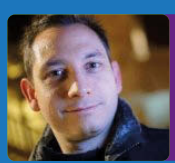
22 **Développeur, le plus beau métier du monde !**

65 HTML 5

81 Visual Studio Community 2013

52 Framework Meteor

68 JavaScript



78 Design tactile

54 Kinect for Windows v2

15 Abonnez-vous !

75 Framework Akka 2e partie

à lire dans le prochain numéro n°183 en kiosque le 28 Février 2015

WINDOWS 10
Quoi de neuf pour le développeur ?

LES ÉCOLES INFORMATIQUE
Comment choisir ?
Quel cursus ?
Quel diplôme (ou pas) ?

ACCESSIBILITÉ
Comment rendre accessible vos applications et sites Web ?
Quels sont les standards ?

Voiture connectée :

nouveau terrain de jeux des éditeurs technologiques

Samsung

va-t-il se repositionner, coincé entre Apple sur le haut de gamme et les constructeurs chinois sur le bas de gamme ?

BIENVENUE DANS LA MATRICE, SUITE DE LA SUITE : OSVR

La réalité virtuelle n'est plus virtuelle. L'intérêt pour Occulus Rift montre qu'il s'agit d'un marché en devenir, avec des perspectives immenses. Pour faire contrepoids et proposer une plate-forme open source, Leap Motion, Razer, gearbox et d'autres ont décidé de s'associer pour créer l'Open Source Virtual Reality. Cette initiative concerne le jeu en réalité virtuelle. L'OSVR aborde le matériel, le logiciel et le développement. L'OSVR propose même un casque de référence, avec un tarif très prometteur (199 $). Mais pour le moment, les principaux constructeurs de casques sont absents de cette initiative...

Site : http://www.osvr.com

R.I.P. Google TV :

l'expérience Google TV est définitivement morte. Google la remplace par Android TV, en cohérence avec les déclinaisons systèmes annoncées mi-2014.

Screen images simulated

WINDOWS PHONE : IL FAUT RÉAGIR

Kantar a dévoilé quelques chiffres sur le marché des systèmes mobiles pour la période septembre – novembre 2014, pour la France :

- Android : 68,3 %
- iOS : 20,8 %
- Windows : 9,8 %

En un an, Windows a perdu 3 points, alors que Android et iOS ont gagné des marchés. Même constat au niveau européen. Malgré les qualités du système, Microsoft a toujours le plus grand mal à convaincre. Windows 10 apparaît donc comme crucial sur les terminaux mobiles, mais le nouveau Windows ne réglera pas tout à lui tout seul.

Miroir, miroir, quel est le langage le plus utilisé ?

L'index TIOBE donne chaque mois les langages les plus utilisés par les développeurs. Il s'agit d'un indicateur basé sur les recherches web. Des % à manipuler avec précaution.

Janvier 2015	Janvier 2014	Tendance	Langage	%	Evolution en %
1	1	↓	C	16,703	-1,24
2	2	↓	Java	15,528	-1
3	3	↓↓	Objective-C	6,953	-4,14
4	4	↓	C++	6,705	-0,86
5	5	↓	C#	5,045	-0,80
6	6	↓	PHP	3,784	-0,82
7	9	↑↑	JavaScript	3,274	+1,70
8	8	↑	Python	2,613	+0,24
9	13	↑↑	Perl	2,256	+1,33
10	17	↑↑	PL/SQL	2,014	+1,38

Le podium reste identique : C, Java et Objective-C. La plus forte baisse est pour Objective-C. Loin derrière, Swift est 25e, Dart 34e. VB.Net et VB arrivent 16e et 17e.

SPARTAN : MICROSOFT VEUT-IL RELANCER LA GUERRE DES NAVIGATEURS ?

Microsoft travaillerait sur un nouveau navigateur : Spartan. Le projet fait beaucoup parler de lui. Spartan pourrait être une solution, alternative à Internet Explorer, pour permettre à l'éditeur de reprendre de la vigueur sur le marché des navigateurs Web. Chrome, depuis sa sortie, a eu un impact considérable, même Firefox a du mal à résister. Spartan pourrait plus ressembler à ces concurrents directs qu'à Internet Explorer, mais il utiliserait les mêmes moteurs de rendus que IE. L'utilisation de WebKit aurait pu être une alternative intéressante. Attendons maintenant sa présentation officielle...

L'INNOVATION DANS LE MONDE : OUPS, OÙ EST PARIS ?

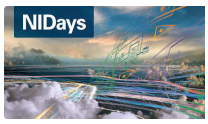
Le cabinet KPMG a dévoilé son étude « global technology innovation insights – fall 2014 ». Les analystes dévoilent la carte mondiale de l'innovation, les villes et pays les plus actifs et les technopoles les plus innovantes. Les États-Unis sont toujours vus comme un centre d'innovation très fort, mais la Chine est aujourd'hui très active. Les villes perçues comme les plus innovantes, selon KPMG, sont :

1 Shanghai
2 Tokyo
3 Pékin
4 New York
5 Séoul
6 Londres
7 Mumbai (Inde)
8 Tel-Aviv
9 Hong Kong
10 Boston

L'effet French Tech n'a, semble-t-il, pas impressionné les analystes. Paris et la France sont absentes. Peut-être dans la prochaine édition ? Il reste encore du boulot pour se faire une place visible dans l'innovation...
L'étude complète : http://goo.gl/cjO77L

février

NI DAYS



Le 3 février, National Instruments organise sa journée technique à Paris. De multiples conférences et sessions techniques seront proposées autour des outils de l'éditeur. Les concours robotiques seront eux aussi au rendez-vous ! Site : http://france.ni.com/nidays

TECHDAYS 2015

Les 10, 11 et 12 février, Microsoft organise son événement annuel : les TechDays. Comme chaque année, plus de 300 sessions seront jouées sur les 3 jours. Le grand thème de cette édition sera l'« ambient Intelligence » avec la mobilité, le Big Data, le Cloud, le machine learning, les objets connectés. Bien entendu, DevOps sera présent ainsi que les outils de développement. Beaucoup de sessions sur le Cloud, DevOps, la mobilité, sans oublier beaucoup de développement Web ! Comme d'habitude, deux énormes sessions à ne pas rater : Geek in da house, Coding4Fun. Elles s'annoncent titanesques. François Tonic coanimera, comme dans l'édition 2014, une session DevOps... Comme chaque année, Programmez ! sera présent durant les 3 jours. Venez nous rencontrer. Pour en savoir plus : https://techdays.microsoft.fr

En février chez Zenika

NightClazz Docker

Découverte le 5 février 2015 (Paris)
http://www.zenika.com/nightclazz-docker-decouverte.html

Après avoir hebergé le Docker Tour de France, Zenika continue son investissement sur Docker avec une soirée découverte animée par ses consultants, vainqueurs du Docker Hackathon de Décembre.

Lyon Hadoop Meetup - Batch sur Hadoop avec Cascading, le 5 février 2015 (Lyon) :
http://www.zenika.com/lyon-hadoop-meetups-meetup3.html
Le Lyon Hadoop Meetup est un User Group réunissant les utilisateurs et les personnes désirant en savoir plus sur Hadoop dans la région Rhône-Alpes

avril

JOURNÉE FRANÇAISE DES TESTS LOGICIELS 7e édition

Le 14 avril se déroulera la nouvelle édition de la journée des tests logiciels. Cette année, l'évènement change de lieu. Comme à chaque édition, la journée sera ponctuée de sessions générales et plus « techniques ». Une journée sera consacrée aux tutoriels (13 avril), la journée ouverte étant celle du 14 avril. De nombreux éditeurs et SSII seront présents. Pour en savoir plus : http://www.jftl.org

Agenda des JUG

Le Paris JUG annonce les soirées suivantes :
- 10 février : soirée Cassandra
- 9 mars : thème ouvert

Site : http://www.parisjug.org
BreizhJug http://www.breizhjug.org
- 5 février : soirée Cassandra

Riviera JUG http://www.rivierajug.org
- 19 février : soirée Gradie
- 19 mars : soirée jOOQ

ElsassJug
- 24 février : soirée DDD, CQRS, event sourcing

Lyon JUG http://www.lyonjug.org/evenements
- 24 février : tests d'intégration
- 10 mars : soirée human talks Lyon
- 16 & 17 avril : événement Mix-IT
- 19 mai : Akka sur cluster Raspberry Pi
- 16 juin : Cassandra et Spark

Hébergement : jouez à cache cache ! 6e et dernière partie

Dernière étape d'un long processus, la mise en production offre une dernière chance de réduire les impacts d'un site en optimisant son hébergement. Les bonnes pratiques sont nombreuses à ce niveau.

Frédéric **Bordage,**
Expert écoconception logicielle, GreenIT.fr, @greenit
Jérémy **Chatard,**
Directeur technique de Breek, @breekfr

Même si un site Web accumule des défauts de conception et un code de piètre qualité, il est souvent possible de *rattraper le coup* lors du passage en production. Une étape cruciale, qui devrait être intégrée dès la conception et le choix des technologies, tant l'hébergement joue un rôle clé dans le coût de fonctionnement final et les impacts environnementaux associés au site. Les points d'attention ne manquent pas concernant l'hébergement. Ils concernent à la fois la mise en cache tout au long de la chaîne applicative, la réduction du nombre de requêtes et de la taille du flux de données à transférer, jusqu'à l'optimisation des serveurs et des ressources services. Sans oublier le choix d'un hébergeur réellement engagé dans la préservation de l'environnement.

Réduire le nombre de requêtes HTTP par page

Plus on évite les allers-retours entre le navigateur de l'internaute et le serveur HTTP et moins les impacts environnementaux associés au site seront importants. En effet, chaque requête HTTP nécessite un dialogue entre le client et le serveur afin qu'ils se mettent d'accord sur les ressources à échanger. Un fichier de 100 Ko transmis à l'aide d'un seul GET HTTP consommera moins de ressources serveurs que 10 fichiers de 10 Ko transmis via 10 GET, car il ne sollicitera qu'une seule session HTTP, sans temps de latence induit par le téléchargement préalable d'une autre ressource. Au-delà d'une conception sobre, vous pouvez réduire le nombre de requêtes en regroupant les ressources au sein d'un seul fichier à transférer.

C'est par exemple le cas des images regroupées dans un sprite que nous avions vu dans une précédente partie. Vous pouvez faire la même chose en combinant les librairies Javascript et CSS. Des outils, comme le module PageSpeed de Google pour Apache, automatisent ce processus. Ce dernier fournit un ensemble de filtres spécialisés, parmi lesquels (liste non exhaustive) :

- combine_css. Combine les CSS en une seule.
- combine_heads. Combine plusieurs entêtes <head> en une seule.
- combine_javascript. Combine plusieurs fichiers JavaScript en un seul.

Enfin, héberger les ressources – images, CSS, Javascript, etc. - sur un domaine sans cookie évite au serveur d'envoyer un cookie pour chaque ressource... alors que celui-ci est inutile. On réduit ainsi la bande passante consommée tout en réduisant le nombre de ressources transférées (et donc le nombre de requêtes HTTP).

Optimiser la mise en cache des ressources statiques

Il faut rapprocher le plus possible le contenu du site de l'internaute. Cela commence par définir correctement la durée de vie des ressources statiques telles que les images, librairies javascript, CSS, etc. La première méthode consiste à ajouter des entêtes Expires ou Cache-Control à la requête HTTP pour préciser au navigateur la date d'expiration des ressources. Ainsi le navigateur ne redemandera pas ces éléments au serveur. Ce qui économisera des requêtes HTTP, de la bande passante et de la charge côté serveur. Si vous n'avez qu'un seul serveur HTTP, vous pouvez aussi utiliser un Entity Tag (ETag) pour obtenir le même résultat. Un ETag est une signature jointe à une réponse du serveur. Il permet de différencier deux documents. Si le client demande une ressource (page HTML, image, CSS, etc.) dont le ETag est identique à celui qu'il a déjà, le serveur Web lui répondra qu'il n'a pas besoin de télécharger la ressource et qu'il est préférable d'utiliser celle dont il dispose déjà. Sur Apache et Nginx, il suffit d'ajouter une ligne dans le fichier de configuration.

Apache : http.conf

```
FileETag MTime Size
```

nginx : nginx.conf

```
http {
  [...]
  etag on;
}
```

Eviter de solliciter inutilement le serveur http

Les reverse-proxies sont optimisés pour servir des ressources de façon rapide en consommant le moins de cycle CPU possible. Ils préservent le serveur applicatif du trafic et sont donc généralement une source importante d'économie en matériel, et donc en coûts et en impacts environnementaux. Selon le contexte, vous pouvez utiliser Squid, Varnish ou Nginx en frontal du / des serveur(s) Web pour stocker et fournir les éléments statiques et les pages HTML déjà calculées. Varnish permet même de cacher des fragments de pages HTML pour décharger le serveur d'application.

Décharger le serveur d'applications et la base de données

Dans la même logique, plus on s'enfonce profondément dans la chaîne applicative et plus le coût de chaque requête est important en terme de ressources (RAM, CPU). On peut donc commencer par décharger le serveur d'applications et la base de données en activant tous les caches du CMS. Par exemple, dans Drupal il est possible de cacher les requêtes des vues, les résultats des vues, les blocs affichant les vues et la page dans son ensemble. On peut également décharger le serveur d'applications en utilisant un cache de code intermédiaire (opcode dans le cas de PHP et bytecode dans le cas de Java).

Certains accélérateurs permettent de mettre en cache ce code binaire, ce qui évite de le recompiler à partir du code source à chaque fois que la page est demandée (si elle n'a pas changé). Il y a donc une réduction du temps de compilation, donc économie de cycle CPU et de mémoire vive (RAM). Les caches pour opcode les plus utilisés sont APC et Zend OpCache. A noter également la machine virtuelle JIT (Just In Time) HHVM qui compile du code source PHP en C++ puis en binaire via g++. Evidemment, si le cache de code intermédiaire s'exécute entièrement en RAM, les temps de réponse sont meilleurs et l'ensemble de la chaîne applicative est libérée plus tôt, permettant ainsi de servir plus d'internautes avec la même infrastructure physique. Les systèmes de caches doivent donc être montés entièrement en RAM lorsque c'est possible.

DzMob, agence mobile.

Paris - Alger

Contactez-nous au

06 95 91 20 74

Retrouvez plus d'informations sur

www.dzmob.fr

Notre proposition : Vous apporter une expertise complète autour du mobile.

DzMob est une agence mobile dynamique basée à Paris et à Alger et spécialisée dans le développement d'applications iPhone, iPad, Android, Windows Phone et Windows 8.

DzMob

djamel@dzmob.net

+33 9 83 90 20 74 / +33 6 95 91 20 74

19, rue des Champs, 92600 Asnières sur Seine

Cela évite les entrées/sorties sur les disques durs (ainsi que les instructions CPU pour le faire). La RAM est très rapide en termes d'accès lecture/écriture, la durée de consommation des ressources est donc très courte.

Réduire la taille des fichiers à transférer

Vous pouvez compresser les fichiers HTML, CSS et Javascript pour limiter l'utilisation de bande passante et améliorer les temps de chargement. L'idéal est de compresser les fichiers CSS et Javascript manuellement avant de passer en production et d'activer la compression automatique au niveau du serveur HTTP pour les pages HTML. On peut ainsi diviser par 4, 5 et même jusqu'à 6 la bande passante consommée. Par exemple, avec Apache, la directive deflate réduit la taille d'un fichier HTML de 26 Ko sans compression à 6 Ko après compression.
Configuration du fichier .htaccess d'Apache

```
# Lien vers le module
<IfModule mod_deflate.c>
 SetOutputFilter DEFLATE
 DeflateCompressionLevel 9

# Pour ne pas compresser inutilement les fichiers déjà compresses…
 SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png)$ no-gzip dont-vary
 SetEnvIfNoCase Request_URI \.(?:exe|t?gz|zip|bz2|sit|rar)$ no-gzip dont-vary

</IfModule>

# Compresser text, html, javascript, css, xml:
<Location />
 AddOutputFilterByType DEFLATE text/plain
 AddOutputFilterByType DEFLATE text/xml
 AddOutputFilterByType DEFLATE text/html
 AddOutputFilterByType DEFLATE text/css
 AddOutputFilterByType DEFLATE image/svg+xml
 AddOutputFilterByType DEFLATE application/xhtml+xml
 AddOutputFilterByType DEFLATE application/xml
 AddOutputFilterByType DEFLATE application/rss+xml
 AddOutputFilterByType DEFLATE application/atom_xml
 AddOutputFilterByType DEFLATE application/x-javascript

# Pour les proxies
 Header append Vary User-Agent env=!dont-vary
</Location>

# Certains fichiers via leur extension
<files *.html>
SetOutputFilter DEFLATE
</files>
```

En plus de compresser les ressources statiques, il faut s'assurer qu'elles ne contiennent pas de données inutiles lorsque le site est en production : espaces, commentaires, sauts de ligne, etc. On peut même utiliser certains outils comme YUI Compressor qui vont jusqu'à compresser les noms de variables locales. Un fichier Javascript de référence de 248 Ko ne pèse plus que 97 Ko après minification et moins de 50 Ko après compression (deflate / gzip), soit un rapport de 1 à 5 avec et sans optimisation. Le module PageSpeed de Google pour Apache permet d'automatiser ces opérations (minification, compression, combinaison de plusieurs fichiers en un seul, etc.).

Optimiser les images

Choisir le bon format d'image ne consiste pas à se limiter au choix d'un seul format pour tout le site. En effet, selon le besoin - transparence, palette étendue ou limitée de couleurs, etc. – le format pourra varier entre GIF, PNG et JPEG. Chacun de ces formats possède des qualités qu'il faut exploiter au maximum. Par exemple, le format JPEG est excellent pour compresser les photos tandis que le format GIF peut s'avérer très économe pour de petites icônes possédant moins de 256 couleurs. Vous pouvez optimiser les images de votre site en utilisant PNGCrunch, ImageMagick ou bien encore jpegtran. Il n'y a pas de petits gains, surtout lorsque, comme certains de nos clients, vous affichez plus d'un milliard de pages par an, ou que vous payez la bande passante au Mo ou bien encore que les internautes se connectent à vos services depuis des connexions mobiles de piètre qualité (2G, etc.).

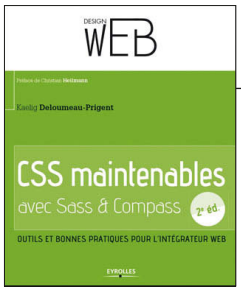


Image d'origine

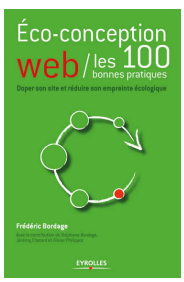
Exemple de gain de poids pour une image

Avant	Après
Format : PNG	Format : GIF
Nombre de couleurs : 10535 couleurs	Nombre de couleurs : 254 couleurs
Taille : 317x384	Taille : 140x200
Taille affichée : 140x200	Taille affichée : 140x200
Besoin de transparence : non	Besoin de transparence : non
Poids : 75 Ko	Poids : 10 Ko

Choisir un hébergeur respectueux de l'environnement

Quand vous avez fait tout votre possible au niveau du code et des machines, il ne vous reste plus qu'à choisir un hébergeur réellement engagé dans la protection de l'environnement. Il n'existe pas de label fiable dans ce domaine. Il faut donc mener son enquête en prenant en compte 6 critères importants :

1. la gestion des DEEE. L'hébergeur privilégie-t-il une longue durée de vie des serveurs et leur reconditionnement plutôt que leur recyclage ?
2. PUE. Calcule-t-il son Power Usage Effectiveness et celui-ci est-il en dessous de la moyenne nationale ?
3. Politique d'achat. Les serveurs et autres équipements sont-ils écolabélisés ?
4. Aspects sociétaux. L'hébergeur privilégie-t-il les acteurs de l'économie sociale et solidaire (ESS) ?
5. Energies renouvelables. Le data center est-il alimenté avec une énergie renouvelable ? Par exemple via l'achat de certificat EECS.
6. Compensation Carbone. Enfin, l'hébergeur compense-t-il les émissions de gaz à effet de serre qu'il n'a pas pu éviter ?



Retrouver ces bonnes pratiques dans le livre EcoConception.

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

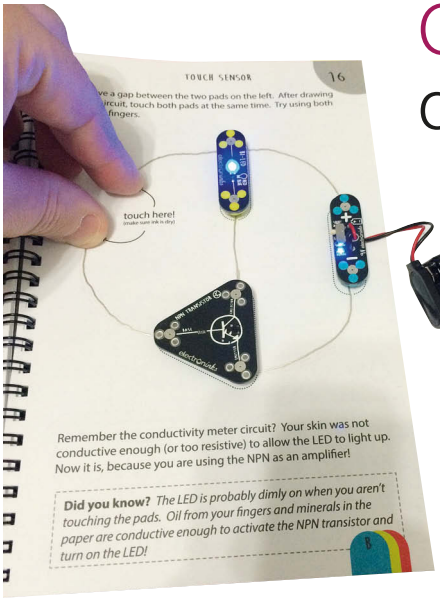
Cloud Privé
Infogérance
PRA/PCA
Haute disponibilité
Datacenter

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



Circuit Scribe : le stylo magique des geeks et des makers

Non, il n'y pas que les Arduino et Raspberry Pi dans le monde des geeks et des makers. Nous avons découvert et testé un outil absolument génial : Circuit Scribe. Conseillé à partir de 13 ans, Circuit Scribe est un projet né sur KickStarter.

François Tonic

Les fondateurs du projet espéraient réunir 85 000 $ pour lancer l'idée et sortir les premiers kits. Ce sont finalement 674 425 $ qui ont été récoltés ! Impressionnant. Un an après, le projet existe, les premiers kits sont disponibles et la communauté commence à se développer. En un mot : c'est énorme !

C'est quoi ?

L'idée de départ est très simple : tracer, dessiner des circuits sur une feuille avec un stylo (un rollerball) contenant une encre conductrice. Quand on achète un kit, on dispose de : un stylo, une plaque métallique, d'un petit guide, de connecteurs et de composants (variant selon le kit). On se prend vite au jeu ! Circuit Scribe n'est pas le premier à proposer une électronique simple comme un stylo mais la finesse de l'encre fait la différence.

Comment ça fonctionne ?

Le fonctionnement est très simple :

- Sur une feuille de papier (ou une autre matière, mais nous n'avons pas testé), on dessine, avec le stylo livré, les points de connexions pour les composants et le circuit d'ensemble,
- Placer en-dessous, du papier, une plaque ou une feuille métallique,
- Positionner les différents modules et la source d'énergie : bien vérifier que les points de contacts entre l'encre et le module se font bien et que le circuit dessiné est fermé (c'est à dire qu'il est intègre, sans ligne coupée)
- tester le circuit

C'est très simple.

Attention : Circuit Scribe n'est pas vendu en France. Vous pouvez commander les kits et modules aux Etats-Unis. Plusieurs kits et modules sont en rupture de stock.

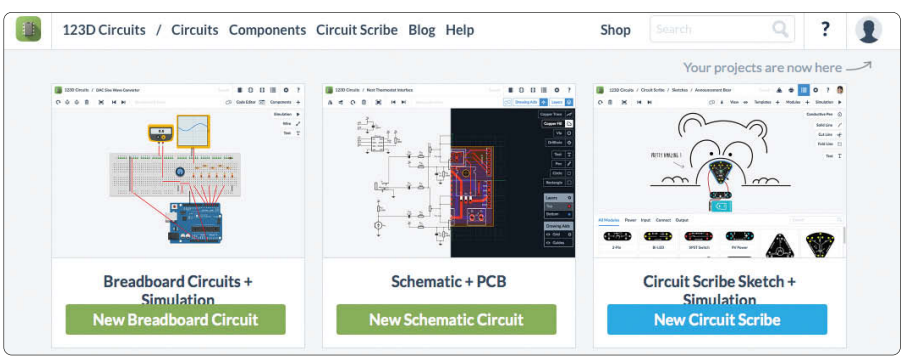
Il existe 4 types de modules :

- Module power (bleu clair) : pour connecter une pile ou toute autre alimentation (pile ou via l'USB)
- Module Input (rouge) : tous les modules pour contrôler les modules Output (en sortie). Nous trouvons les switchs, le potentiomètre,
- Module connect (gris) : tous les composants servant à connecter ou à amplifier les signaux,
- Module Output (jaune) : tous les modules de sortie tels que les leds, le buzzer, le moteur...).

Concrètement, on fait quoi avec ?

Bon, passé l'effet de surprise et quelques tests, la même question revient toujours : à quoi ça sert ? Circuit Scribe est avant tout un support pédagogique pour apprendre l'électronique et s'amuser sans soudures, sans circuits imprimés. Le kit basic permettra de tester la conductivité, les montages les plus basiques mais pour aller au-delà, soit vous utiliserez des composants que vous saurez monter et utiliser, soit vous achèterez les modules complémentaires. Le plus dur est de trouver un usage, un scénario. Vous pouvez imaginer des scénarii avancées : Surface tactile avec une arduino : paperduino est intéressant pour voir les possibilités (niveau : avancé / expert) : http://www.instructables.com/id/Paperduino-20-with-Circuit-Scribe/

Concevoir ses propres composants pour créer des circuits sur mesure. La notion d'open hardware est très importante pour electroninks (le fabriquant). Circuit Scribe fournit la plateforme technique mais de nombreux plans sont disponibles sur GitHub et vous pouvez créer vos propres modules. Un outil en ligne permet de créer et de simuler des schémas Circuit Scribe : 123d d'Autodesk. Si l'outil est en bêta, il permet de créer des circuits, de les sauvegarder, de les partager.



A suivre dans Programmez! 183...

Kits et modules

kit	contenu	tarif officiel
Basic Kit + Book	2 Bi-Directional LED 1 Switch 1 module pour résistance 1 Transistor NPN 1 adaptateur 9V 1 pile 9V 1 stylo 1 livre Des résistances Une plaque métallique	59,99 $
Maker Kit	contenu Basic Kit + 1 Led RGB + 1 Buzzer + 1 capteur photo + 1 relais	79,99 $
Developer Kit	contenu du Maker Kit, tous les modules sont doublés. + 10 connecteurs (câbles) + 1 module USB	194,99 $

Ressources

Vidéo démo :
http://www.youtube.com/watch?v=SGrGLNmphqs
GitHub : https://github.com/Circuit-Scribe/Modules
Electroninks : http://www.electroninks.com

Sur abonnement ou en kiosque

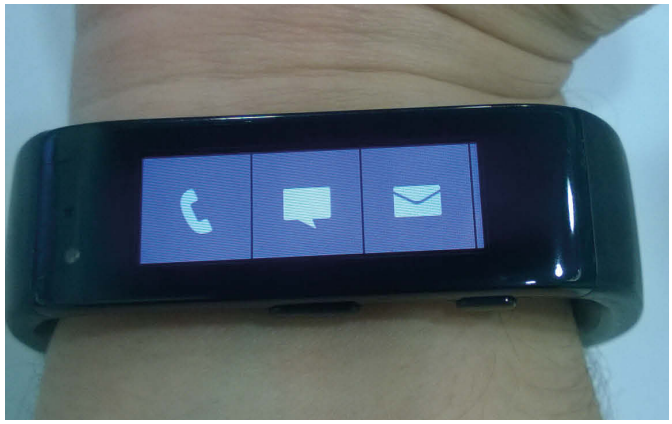
Le magazine des pros de l'IT

Mais aussi sur le web

Ou encore sur votre tablette

L'INFORMATICIEN

Microsoft Band : un bracelet connecté



Annoncé fin novembre 2014 et mis en vente dans la foulée aux Etats-Unis, le Microsoft Band se veut le fer de lance de la plateforme Microsoft Health à l'instar des tablettes Surface pour le marché des tablettes sous Windows 8. Ce bandeau est bardé d'une dizaine de capteurs permettant, entre autres, de compter le nombre de pas, de prendre le rythme cardiaque ou la chaleur de la peau, de mesurer la quantité de rayons ultra-violets ou encore de localiser le porteur grâce à une puce GPS.

John **Thiriet**
Consultant/Formateur .NET
MVP Windows Platform Development

A la différence des montres connectées telles que l'Apple Watch (sortie prévue en février ou mars 2015, NDLR), ou les montres Google Wear qui nécessitent respectivement un iPhone ou un téléphone sous Android pour fonctionner, le Microsoft Band est compatible avec ces derniers en plus de Windows Phone.
C'est un produit à cheval sur deux marchés, celui de la santé et celui des montres connectées. Finalement c'est un mélange des genres qui, nous le verrons, n'a pas pour but de remplacer complètement bracelets et montres, mais qui pourra satisfaire un grand nombre d'utilisateurs dont je fais désormais partie. Dans cet article, je vous présenterai le produit et j'exposerai mon ressenti après un mois d'utilisation quasi-quotidienne de ce bracelet et de son application mobile. Je vous montrerai comment il m'a incité à marcher plus et comment il peut servir au final à mieux vivre au quotidien. Nous parlerons aussi des possibilités à venir que ce genre d'appareil peut apporter en termes de développement d'applications.

La plateforme

Avant de commencer à parler du Microsoft Band, il convient de parler de Microsoft Health. Toutes les données captées par le Microsoft Band sont synchronisées dans le Cloud sur cette dernière. Outre le fait de stocker ces données, elle a pour but de donner des conseils à ses utilisateurs pour mener une vie en bonne santé en fonction des objectifs qu'ils se seront définis.
Il faut noter que Microsoft Health n'est pas réservé aux produits Microsoft mais est une plateforme ouverte. Il est possible à d'autres entreprises de ce segment de marché d'y stocker et d'y traiter leurs données dont RunKeeper fait notamment partie.

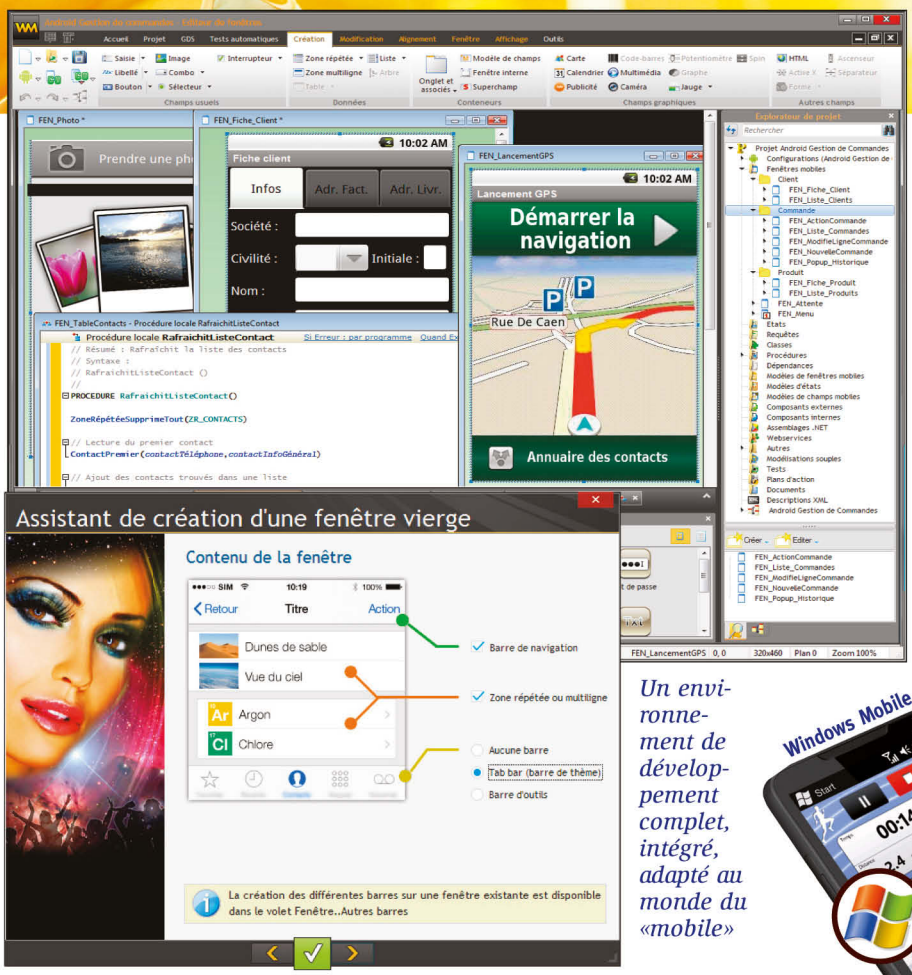
Le bracelet

Lorsque j'ai mis le Microsoft Band pour la première fois, j'ai tout de suite constaté son poids et la place qu'il prend. On l'oublie difficilement au cours de la journée, tout du moins les premiers jours car après un mois on s'y habitue. On peut le porter à la manière d'une montre ou positionner l'écran vers l'intérieur de poignet. Le bracelet semble avoir été pensé pour cette dernière position. Il est en effet plus aisé d'utiliser l'écran tactile dû à son format paysage. Le bracelet fait ce qu'on attend d'une montre : il donne l'heure et la date mais il rajoute quelques informations comme le rythme cardiaque, le nombre de pas effectués et la distance parcourue dans la journée ou encore les calories consommées. Son interface est organisée grâce à des tuiles, ce qui ne choque pas si l'on vient de Windows Phone ou Windows 8. En bref, on retrouve très facilement ses marques si l'on est déjà utilisateur de ces systèmes.
Il est également possible de personnaliser l'affichage du bracelet en choisissant les applications installées, l'ordre d'affichage ou encore les couleurs à utiliser. En plus de son écran tactile, le bracelet dispose de deux boutons : le bouton d'accueil et le bouton d'action, qui nous permettent respectivement de revenir à l'écran d'accueil et de lancer une activité. Parmi les fonctionnalités proposées et attendues d'une montre connectée, on trouve toute une panoplie de notifications :

- Appels entrants et appels manqués,
- Réception des mails avec prévisualisation du début du mail,
- Réception des SMS avec possibilité d'y répondre à la voix en fonction du téléphone utilisé,
- Alertes du calendrier,
- Notifications émises par les applications tierces.

Outre les applications de santé, il existe aussi quelques applications tierces comme Facebook, Twitter et Starbucks aux Etats-Unis. Cette dernière permet de payer directement avec son bracelet. J'ai fait le test à Seattle, et ça marche vraiment très bien ! Le barista m'a dit en anglais avec l'entrain caractéristique des américains : « C'est la chose la plus cool que j'ai jamais vue ! ».
Au quotidien j'ai trouvé trois choses extrêmement pratiques. Le compteur de pas et de distance, la surveillance du sommeil, et les notifications mails et SMS. Les notifications sur le bracelet sont un excellent moyen de savoir si cela vaut le coup de sortir le téléphone pour le lire ou répondre au SMS. En effet, si je suis en

WINDEV MOBILE 20
LE DÉVELOPPEMENT NATIF SUR TOUS LES MOBILES



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

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, tablettes, netbook,...

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 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, Gestionnaire de Sources collaboratif, Générateur de dossier de programmation, Suivi des plannings,...

TOUS LES CONSTRUCTEURS

Les applications réalisées avec **WINDEV Mobile 20** fonctionnent sur les terminaux de tous les constructeurs: Datalogic, Falcon, Intermec, Symbol, PSC, PAXAR, Psion Teklogix, Pidion, Gotive, HHP... Tous les smartphones sont supportés: Apple iPhone, Qtek, Toshiba, HP, Asus, Acer, Compaq, Orange, Samsung, Paragon, HTC, Motorola, LG... Toutes les tablettes: Apple iPad, Samsung Galaxy Tab, Galaxy Note, Nexus, Kindle, Acer, Asus, Archos, Microsoft, Sony, Msi, HP, Toshiba, Motorola, HTC, Lenovo, LG, Huawei...

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

Fournisseur Officiel de la Préparation Olympique

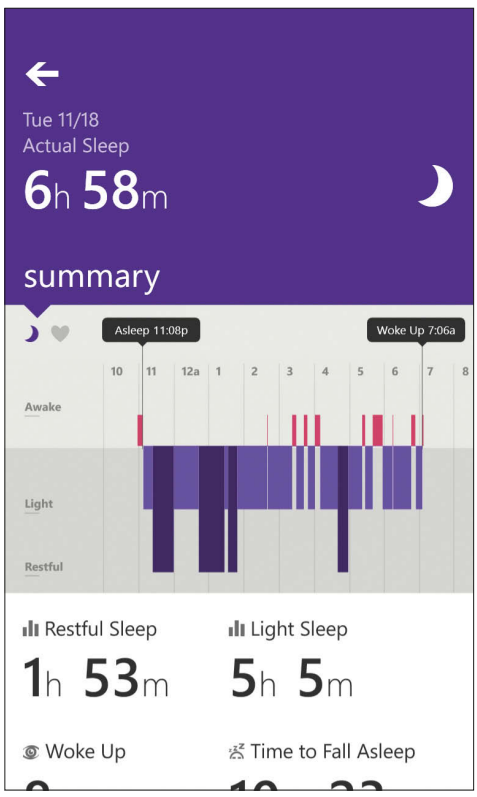
www.pcsoft.fr
Des centaines de témoignages sur le site

réunion cela me permet tout de suite de savoir si un message est vraiment important ou pas. Ce sont des choses dont j'aurais du mal à me passer maintenant.

L'application

L'application associée au bracelet porte le même nom que la plateforme Cloud sous-jacente c'est-à-dire Microsoft Health. Elle n'est pour l'instant disponible que sur le store américain de chaque plateforme mobile cible. N'ayant testé que la version Windows Phone de l'application, c'est donc à celle-ci que je ferai référence, et mes propos ne présagent en rien de la qualité des versions Android ou iPhone. De la même manière je parle de la version de l'application publiée au moment où j'écris ces lignes. Il en sera peut-être autrement, mais pour l'instant ce n'est pas le cas. L'écran principal est facile d'utilisation. Il regroupe les résumés des dernières activités en cours et permet facilement de voir encore le nombre de calories brûlées, si on a bien dormi ou si on marché suffisamment.

Lorsque l'on va un peu dans les pages de détails, les graphiques sont clairs et détaillés. L'application nous donne toujours des informations sympathiques permettant de mieux se rendre compte du travail effectué, ou permettant de nous donner quelques conseils comme des temps de repos. C'est aussi dans ces pages de détails que l'on va configurer nos objectifs de nombre de pas à faire, ou de calories à brûler quotidiennement.



Dans l'ensemble l'application est fonctionnelle, mais on sent bien que c'est une première version qui pourra apporter beaucoup plus lors de prochaines mise à jour. Je regrette aussi le manque d'explications en ce qui concerne les résultats de la surveillance du sommeil.

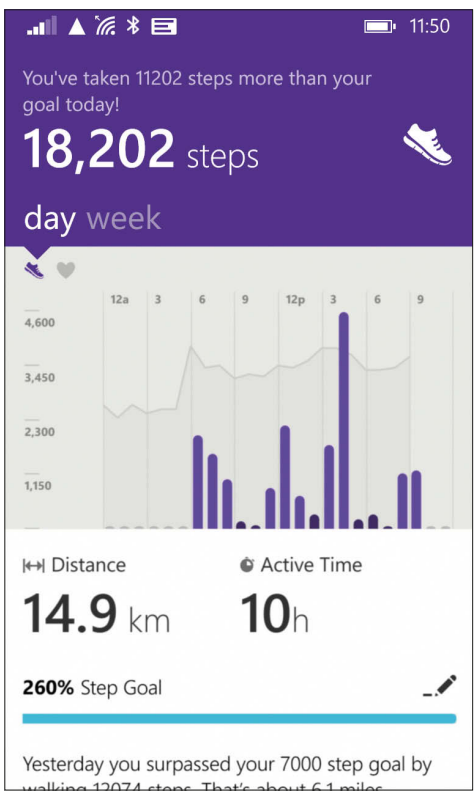
Santé

Vivre en bonne santé est un souhait universel dont la concrétisation se heurtait à la difficulté de mesurer au quotidien ce qui définit l'état de santé. L'arrivée des bracelets connectés a commencé à changer la donne, et c'est dans cette ligne que le Microsoft Band se situe. Son positionnement est néanmoins légèrement différent car le but de Microsoft ici n'est pas directement de vendre un bracelet, mais bien de vendre sa plateforme de santé. C'est dans cette optique que tous les capteurs présents dans le Microsoft Band sont ouverts à licences et donc utilisables par tous ceux qui seraient intéressés. Cela ouvre la voie à un grand nombre possible de variations de bracelets dont l'avenir nous montrera l'étendue.

En plus des applications liées aux données de santé recueillies en permanence, on a la possibilité de lancer quelques applications spécifiques. Parmi celles-ci, on retrouve l'application de course, l'application de sommeil, celle d'entraînement libre et celle de plan d'entraînement. A noter que cette dernière se configure via l'application téléphone. On y choisit les types d'entraînements que l'on souhaite effectuer en regardant les textes et les vidéos d'explications, et en envoyant ces plans d'entraînement sur le bracelet.

J'ai particulièrement apprécié le détail des informations fournies par le bandeau avec l'application de course. En plus de la récupération des coordonnées GPS (sans besoin d'un téléphone) et de la construction d'une carte de la course effectuée, le bracelet affiche en temps réel la distance parcourue, le rythme cardiaque, le temps écoulé et les calories brulées.

Avoir toutes ces données accessibles en permanence incite à en faire un peu plus à chaque fois. Le bandeau nous informe quand les objectifs que l'on a définis ont été dépassés. Cela incite aussi à remonter les objectifs et permet d'enclencher un cercle vertueux. Par exemple je suis passé d'une moyenne de marche journalière d'environ 5km à environ 10km en troquant un trajet en métro par un peu de marche.



Conclusion

Avant l'annonce du Microsoft Band, cela faisait quelques temps que je réfléchissais au fait de me procurer un bracelet connecté pour le sport. Cependant les bracelets déjà commercialisés ne contiennent pas souvent de GPS intégré. On m'a dit que je pouvais prendre mon téléphone avec moi pour cela mais je n'apprécie pas du tout de courir avec mon téléphone. Et lorsque j'ai lu les caractéristiques du Microsoft Band j'ai su que c'était le bracelet adapté pour moi.

Certains bracelets offrent des autonomies très grandes bien adaptées aux grands sportifs, randonneurs etc... au prix des fonctionnalités de montres connectées. Je ne fais pas partie de cette catégorie de personnes, aussi, recharger le bracelet tous les deux jours n'est pas vraiment gênant. L'intégration du GPS au bracelet ainsi que l'ensemble des fonctionnalités d'une montre connectée précédemment évoquées dans cet article m'ont vraiment incité à l'essayer. Selon moi le Band est déjà un très bon produit dans sa version actuelle.

L'application gagnerait à être améliorée mais cela viendra avec le temps. L'ouverture à la fois de la plateforme Microsoft Health et des licences matérielles vont probablement permettre à un vrai écosystème d'émerger dont le Microsoft Band est le pionnier et le fer de lance. On peut regretter l'absence de SDK public à l'heure actuelle, mais Microsoft devrait nous en préparer un pour l'année 2015 ce qui ouvrira la voie à de nombreuses applications innovantes.

Projet : Vocalys

Comme chaque année, l'école Epitech organise son forum de l'innovation (Epitech Innovative Project). Les projets des étudiants y sont exposés. Nous avons été particulièrement impressionnés par deux projets : Vocalys et Digital Wardrobe. La suite dans Programmez ! 183. Bravo à toutes les équipes.

La rédaction

Thomas Solignas
en 5e année à **Epitech**

Il y a presque trois ans, naissait « Jarvis », un projet nommé en référence à l'assistant vocal de Tony Stark dans Iron Man. Une équipe de 6 personnes avec un objectif un peu fou : permettre aux machines de comprendre le langage humain. Un prototype plus tard, l'équipe s'agrandit, et le projet prend le nom de Vocalys. C'est le début d'une course de 2 ans, avec une armée de 16 étudiants, pour faire de la commande vocale une réalité.

La vision

Passionné d'Intelligence artificielle, et plus largement de tout ce qui naît de l'interconnexion entre les sciences humaines et la programmation, le traitement du langage naturel s'est imposé comme LE sujet que je voulais travailler. C'est de cette envie que j'ai lancé Vocalys, un projet qui pourrait faire de nos films de science-fiction une réalité. Dans un premier temps, Vocalys se voulait être un assistant vocal. C'est à dire, un programme autonome qui comprend le langage, et qui est fait des suggestions. Petit à petit, le projet a évolué, et s'est défini plus clairement. Notamment en supprimant l'aspect suggestion, et en insistant sur l'aspect IHM (Interface homme-machine). Vocalys, c'est la technologie qui permettra à toute machine, à tout logiciel, d'être contrôlée par la voix.

La problématique

Certains logiciels de contrôle vocal sont très populaires, on pense en particulier à Siri (iOS), ou encore à Dragon (ordinateur). Si ce contrôle vocal est possible, pourquoi n'est-il pas popularisé ?

Un constat nous est apparu clairement : le contrôle vocal ne pourra prendre sens que s'il est présent par défaut, et qu'il est disponible partout. La capacité à faire des suggestions, c. a. d, l'aspect assistant, est secondaire. C'est la capacité à pouvoir commander une machine sans avoir besoin d'accéder à son clavier/souris, qui donne tout l'intérêt. Lancer une application — comme Siri ou Google Talk —, c'est déjà beaucoup trop pénible pour un utilisateur. La direction du projet est alors claire : permettre un contrôle vocal disponible par défaut, partout.

Le contrôle vocal, pourquoi ?

L'intérêt du contrôle vocal n'est pas évident, en particulier parce que l'utilisation du clavier/souris s'est très largement inscrite dans nos habitudes. C'est pourquoi Vocalys n'est pas positionné en remplacement de ces outils, mais en complément. L'alternance entre les différents modes d'interaction se fait librement en fonction de ce qui paraît le plus intuitif. Le contrôle vocal prendra naturellement sens dans les situations suivantes :

- Effectuer l'action à distance,
- Garder les mains libres,
- Ne pas avoir à retrouver le bouton dans les menus,
- La commande s'actionne plus rapidement par la voix.

Les moments clés

Vocalys est un EIP (Epitech Innovative Project), projet libre qui s'étale sur les deux dernières années d'Epitech. L'équipe se forme et dispose donc de quelques mois pour formaliser le projet (faire les documentations, les premières ébauches d'architecture...). La quatrième année arrivant, les étudiants se dispersent partout dans le monde dans le cadre du cursus international. On se retrouve donc avec une équipe de 16 personnes dispersées sur 7 créneaux horaires. Au bout d'environ un an, l'équipe se retrouve à nouveau et dispose de quelques mois pour se préparer au salon des EIP, moment de présentation publique du projet.

Gestion d'équipe ou gestion de masse ?

Initiateur du projet, j'ai pris la position de manager dés le départ. Difficiles au premier abord, ces conditions singulières furent l'opportunité d'avoir une grande force de frappe (la plupart des équipes sont moins d'une dizaine), mais aussi d'expérimenter des méthodes de management.

Ayant déjà porté quelques projets sur un fonctionnement agile, j'ai initié Vocalys sur un fonctionnement de scrum. Et, chose importante, j'ai considéré l'organisation sous forme de scrum comme un objet lui-même soumis à des itérations successives. L'idée étant d'adapter l'organisation au fur et à mesure du projet. La première phrase du projet, correspondant aux premiers mois de l'EIP, a principalement été la période d'idéation, de définition du projet (création d'un carnet produit sous forme de story), de priorisation, et de mise en place de la méthode de travail (organisation humaine & technique). La réalisation technique a donc démarré à un moment où l'équipe était déjà dispersée dans le monde.

Du distribué et de l'émergence

Comment faire travailler ensemble 16 membres qui ne sont disponibles ni au même endroit ni au même moment ?

Tout en conservant les éléments fondateurs de l'agile (itérations successives, validation de l'avancement, etc.), la méthode a été adaptée pour fonctionner avec des sous-équipes. Chaque équipe est de petite taille (1 à 3

personnes), diminuant ainsi la complexité d'interaction. Et chaque équipe comprend un « leader », rôle arbitraire qui donne la responsabilité d'organiser la sous-équipe. La formation des équipes n'est pas définitive, elle marche par émergence, en fonction des intérêts de chacun. Les itérations sont d'une durée d'une semaine, et en fonction de l'avancement, les équipes se reforment à chaque fin de sprint. En dehors de ces sous-équipes, deux rôles fixes restent, classiquement le « scrum master » (rôle de gestion humaine et de mise en place du scrum) et le « project director » (responsable des décisions techniques et de la cohérence globale du projet) Fig.A.

Un début difficile, mais une rigueur fructueuse

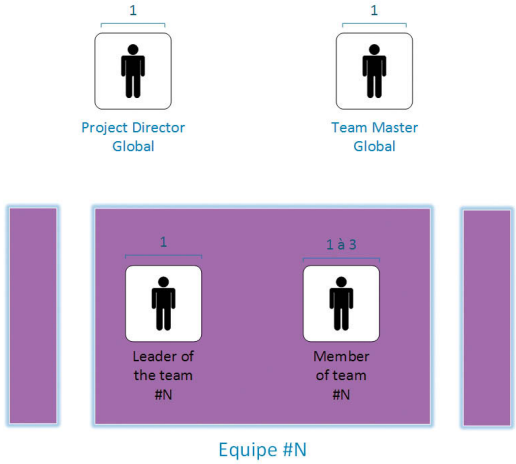
Les premières itérations furent chaotiques. En cause principalement, la complexité de l'environnement technique, ainsi que le « middle management » (prendre des ordres de ses N+1 pour les appliquer sur les N-1) qui ne conviennent pas pour notre projet. Nous sommes rapidement revenus sur une hiérarchie à deux niveaux uniquement, en favorisant les groupes de petite taille (1 voir 2 membres au maximum). Le manager intermédiaire ne peut être efficace que s'il peut remplacer à son niveau le project director et le scrum master, au moins dans une certaine limite. Dans les faits, ce n'est pas le cas. Là où il devrait fluidifier les échanges, au contraire, il les rend plus lourds. C'est pourquoi la suppression de ce niveau de gestion a été un gain d'efficacité. Cette évolution a été possible grâce au fait que la méthode de management soit elle-même soumise à des itérations régulières : est-ce que cela a fonctionné ? Pourquoi cela a-t-il fonctionné ? Est-ce que l'on reproduit la méthode sur une prochaine itération ?

C++, dépendance et portabilité : l'usine à gaz

Les membres de l'équipe travaillent sous Mac, Windows et Unix; on a donc axé notre méthode de travail sur la portabilité. La réalisation du projet s'appuie sur de nombreuses bibliothèques, certaines assez standards (boost, sfml), et d'autres beaucoup moins (outils d'analyse linguistique, sphinx, etc.). Notre souci organisationnel majeur sur le plan technique a été de permettre un fonctionnement relativement unifié pour permettre à chacun de développer dans son environnement sans même parler de déploiement. La solution adoptée, bien qu'étant relativement fiable, est d'une étonnante complexité. Le projet est sur un dépôt git qui comprend des « submodules », qui sont en fait des sous-dépôts git. Il existe un dépôt par distribution (Mac, Unix, Windows), et chacun clone le dépôt correspondant à sa machine. Intégralité d'une initialisation classique sous Unix :

```
git clone git@xxx.git
cd Vocalys
git submodule update -- init data
```

Fig.A




```
git submodule update -- init libs/linux64
mkdir build
cd build
cmake.. && make
```

L'ensemble des sources est géré par CMake qui produit un Makefile ou une solution Visual Studio (2012 ou 2013), selon la configuration. CMake est configuré pour faire le lien avec la version des librairies compatible avec le système utilisé. L'ajout d'une bibliothèque est documenté par un petit « How to ? ».
Cette méthode, à défaut d'être élégante, a permis de travailler avec de nombreuses dépendances (une douzaine sur la version finale), en limitant les problématiques d'installation, et ce indépendamment du système visé.

Les outils linguistiques : un appui fragile

Vocalys s'appuie sur des outils linguistiques pour tout le traitement sémantique. En particulier, sur des dictionnaires et sur des PosTagger (outil d'étiquetage morphosyntaxique). Nous avons constaté que les dictionnaires sont rares, il existe très peu de bases de données linguistiques destinées à être traitées par ordinateur. Concernant les outils d'analyse du langage, très peu sont facilement intégrables. Beaucoup de ces outils proviennent de travaux faits par des doctorants en linguistique, et prennent donc la forme d'un package ésotérique, mélange de langages haut niveau (perl, python) et de script. Rien n'est donc prévu pour l'interfaçage ou simplement pour la réutilisation en dehors du contexte initial pour lequel il a été conçu.

La reconnaissance vocale, pas beaucoup mieux

Un objectif ambitieux et des ressources limitées, laissent peu de temps pour recréer des choses depuis zéro, nous utilisons donc des outils de reconnaissance vocale déjà existants. Après un essai non concluant de CMU Sphinx (la librairie développée par la Carnegie Mellon University), nous avons décidé d'adopter Web Speech API (Google), plus simple d'utilisation. On a donc opté pour interface utilisateur sous forme de page Web statique.

```
var recognition = new webkitSpeechRecognition();
recognition.onresult = function(event)
{
 console.log (event)
}
recognition.start () ;
```

Il nous a fallu peu de temps (et de lignes de code) pour obtenir un résultat concluant. Google fournit un outil de synthèse vocale tout aussi simple.

```
var msg = new SpeechSynthesisUtterance('Hello World');
window.speechSynthesis.speak (msg) ;
```

Néanmoins, Vocalys est basé sur l'idée que la captation audio est continue (c'est à dire, capturer en permanence des sons et réussir à détecter quand Vocalys est sollicité). C'est là que l'API Google a montré ses limites. Elle est prévue pour être activée au début de l'ordre et désactivée ensuite. Au bout de 30 secondes, elle se désactive d'elle-même. Avec quelques modifications, nous avons simulé une écoute quasiment continue ; une nouvelle écoute est lancée dès que la précédente se termine.

Le trophée des EIP

L'aventure Vocalys s'est conclue au salon des EIP, moment unique dans l'année pour présenter son projet et pour tenter de décrocher un trophée. Moment de consécration, Vocalys a obtenu le deuxième prix, sur 94 projets présentés cette année.

Découvrez le (nouveau) langage Oforth

Si vous ne connaissez pas le Forth, le langage de programmation Oforth va vous sembler atypique. Même si certaines syntaxes se rapprochent du C++ ou de Smalltalk, il hérite de la philosophie du Forth : tout tourne autour d'une pile de données qui contient les paramètres à envoyer aux fonctions ou aux méthodes, les résultats intermédiaires, les retours. Et, comme pour le Forth, Oforth utilise la notation polonaise inversée (RPN) : les arguments ou paramètres sont placés sur la pile avant d'appeler les fonctions ou méthodes.

Franck **Bensusan**, *créateur d'Oforth.*
franck.bensusan@oforth.com

Oforth apporte :

- Un modèle objet complet.
- Un garbage collector automatique.
- Bien qu'impératif, des éléments de programmation fonctionnelle.
- Un support complet à la programmation parallèle.

Oforth est un langage non typé et complètement dynamique : il n'y a pas de phase de compilation à réaliser avant d'exécuter un programme. Oforth est un interpréteur qui lit des commandes au clavier (ou en provenance d'un fichier) et les traite.

La pile de données, la notation RPN et l'interpréteur

Pour ceux qui ne sont pas déjà habitués à la notation RPN (Forth, Postscript, calculatrices HP, ...), il y a un ... petit moment d'adaptation à passer. La pile donnée est une pile LIFO (Last In First Out) qui va recevoir les paramètres et les retours des fonctions appelées. Il n'y a pas de séparateur d'instructions, il n'y a pas d'utilisation de parenthèses : le seul séparateur est l'espace. L'interpréteur est donc très simple : il lit un mot jusqu'au prochain espace et l'exécute. Une fois cette exécution terminée, il passe au prochain mot, et ainsi de suite jusqu'à ce que l'ensemble de l'entrée soit traité :

```
>1.2 "aaa" "bbb" Integer 12 13 + .s
[1] (Integer) 25
[2] (Class) Integer
[3] (String) bbb
[4] (String) aaa
[5] (Float) 1.2
```

Ici, l'interpréteur a lu plusieurs mots :

- Un float et 2 strings qu'il a placés dans la pile de données.
- Le mot Integer qui est une classe (mais aussi un objet), qu'il a aussi placé sur la pile.
- Deux entiers (12 et 13) qui ont eux aussi été envoyés sur la pile.

Ensuite l'interpréteur lit le mot "+" et détecte qu'il s'agit d'une méthode et l'exécute. A ce moment, le haut de la pile contient deux entiers, qui sont donc retirés de la pile et remplacés par le résultat. Puis l'interpréteur lit ".s" qui est une fonction qui affiche la pile. On voit que le haut de la pile contient maintenant 25 et que tous les objets sont là, chacun ayant une valeur et une classe.

Hello, World

Ecrivons notre première fonction Oforth (qui peut être saisie directement dans l'interpréteur) :

```
>func: hello { "Hello, World" println }
Ok
>hello
Hello, World
```

Une fonction est simplement une liste de mots que l'interpréteur va exécuter lorsque le nom de la fonction sera saisi : il va exécuter la string en la mettant sur la pile, puis va exécuter println, qui prend le haut de la pile et l'envoie sur la sortie standard, suivi d'un retour chariot. Comme tout est objet, cette fonction, qui vient d'être déclarée, est aussi un objet. L'objet correspondant est #hello. Cet objet, instance de la classe Function, peut être envoyé sur la pile, reçu en paramètre, envoyé en retour :

```
>#hello perform
Hello, World
ok
>#hello #perform #perform perform
Hello, World
ok
>#hello name println
hello
ok
>#hello times(4)
Hello, World
Hello, World
Hello, World
Hello, World
ok
>#hello bench
Hello, World
902
```

Ces exemples introduisent plusieurs autres fonctions ou méthodes : #perform qui permet d'exécuter l'objet au dessus de la pile, #times qui permet de l'exécuter n fois et #bench qui permet de calculer le temps passé pour exécuter le haut de la pile (ici 902 micro-secondes).

Manipulation de la pile

Il est quelquefois nécessaire de manipuler les objets de pile (les dupliquer, les supprimer, supprimer toute la pile, ...). Voici quelques fonctions qui permettent cela :

#dup : duplique le haut de la pile.

#drop : supprime le haut de la pile : le deuxième élément devient le haut de la pile.

#swap : échange les deux éléments en haut de la pile.

#clr : vide la pile.

#tuck : copie le haut de la pile sous le deuxième élément. Par exemple 1 2 3 tuck -> 1 3 2 3

Il existe d'autres fonctions de manipulation, mais nous n'en aurons pas besoin ici.

Les paramètres et variables locales

Les paramètres des fonctions ou méthodes sont envoyés sur la pile de données avant de réaliser l'appel. L'interpréteur fournit une simplification d'écriture pour réaliser cela. Dans l'exemple précédent, la méthode #times a besoin de 2 objets : l'objet que l'on souhaite exécuter et le nombre de fois que l'on veut exécuter l'objet. En notation RPN, la manière de réaliser cela est d'écrire :

>4 #hello times

Cela met les deux objets sur la pile, puis #times est exécuté. L'interpréteur permet de « clarifier » cette écriture en spécifiant que 4 est bien un paramètre de #times et on peut écrire :

>#hello times(4)

Notez qu'il s'agit vraiment d'une simplification d'écriture : l'interpréteur va transformer cet appel en : 4 #hello times avant de l'exécuter.

La déclaration de paramètres dans une fonction est un concept différent. Il ne s'agit pas uniquement d'une simplification d'écriture : si une fonction déclare des paramètres, au moment de l'exécution de la fonction, ces paramètres seront retirés de la pile de données et stockés dans l'environnement d'exécution de la fonction. Par exemple :

```
func: fact(n) { n 0 == ifTrue: [ 1 ] else: [ n fact(n 1 -) * ] }
```

Un paramètre a été déclaré pour la fonction #fact. Lorsque cette fonction sera exécutée, le haut de la pile sera retiré et stocké dans l'environnement d'exécution de la fonction sous le nom "n". Utiliser n dans le corps de la fonction permet de remettre la valeur de ce paramètre sur la pile. Si un paramètre est utilisé plusieurs fois, il peut être plus intéressant de déclarer un paramètre que de « jouer » avec la pile. Sans déclarer de paramètre, la fonction fact pourrait s'écrire de la manière suivante :

```
func: fact { dup 0 == ifTrue: [ drop 1 ] else: [ dup 1 - fact * ] }
```

Ces deux fonctions réalisent la même chose : elles prennent un entier sur la pile et le remplacent par sa factorielle. Ceux qui ont déjà programmé en Forth ou en RPN préfèreront (peut-être) la deuxième méthode. Il est probable que les autres préfèrent la première méthode J. Quelle que soit la méthode utilisée pour écrire #fact, cette fonction peut être appelée de deux manières (la deuxième étant remplacée par la première par l'interpréteur) :

```
>5 fact .s
120
>fact(5) .s
120
```

Il est aussi possible de déclarer des variables locales dans une fonction; elles permettent de conserver des résultats intermédiaires à la place de la pile de données. Ces variables locales, comme les paramètres, font partie de l'environnement d'exécution de la fonction :

```
>func: printSeq(n) { | i | n loop: i [ i println ] }
Ok
>printSeq(5)
1
2
3
4
5
```

La syntaxe loop: [instructions] retire un entier de la pile et exécute les instructions n fois. Pour faire simple, cette syntaxe est équivalente au C : for(i = 1 ; i <= n ; i++) { instructions }

Les blocs

Les blocs sont des fonctions anonymes. Ces objets peuvent être créés puis envoyés en paramètres à d'autres fonctions ou méthodes. Un bloc est créé sur la pile en utilisant la syntaxe : #[instructions] et est ensuite exécuté en utilisant #perform. Par exemple :

```
>12 #[ 1 + ] perform println
13
```

Les blocs peuvent être imbriqués, dynamiques (dépendre de paramètres), ... Un petit exercice : que fait la fonction suivante ? Le nom de la fonction est un indice, mais comprendre comment elle fonctionne est un exercice intéressant...

```
func: fib(n) { 0 1 #[ tuck + ] times(n) drop }
```

Un peu de programmation fonctionnelle

Bien que Oforth soit un langage impératif, le fait que tout soit objet permet de créer des fonctions qui prennent en paramètres d'autres fonctions ou retournent des blocs. Avant cela, introduisons quelques collections. Une liste est créée en utilisant la syntaxe :

[elem_1, elem_2, ..., elem_n] et peut contenir n'importe quel type d'objet :

```
>[ 1, 1.2, « aaa », Date now, #perform ] println
[1, 1.2, aaa, 2014-11-22 16:36:08,014, #perform]
```

Des listes particulières, les intervalles, peuvent être aussi créés sur la pile. Par exemple seq(10) renvoie la liste des entiers de 1 à 10.

Maintenant il est possible de faire un peu de programmation fonctionnelle :

- #apply(aRunnable) applique aRunnable sur chaque élément de la liste fournie en haut de la pile.
- #map(aRunnable) va faire de même mais crée une liste avec l'ensemble des résultats.

Il existe, bien sûr, bien d'autres méthodes (#filter, #reduce, #zip, ...)

Quelques exemples :

```
>0 seq(100) apply(#+) .s
[1] (Integer) 5050
```

#+ est appliqué sur chaque entier de 1 à 100. Le zero initial envoyé sur la pile accumule les résultats et, au final, on se retrouve avec la somme de ces entiers sur la pile.

```
>0 seq(1000) apply(#[ sqrt + ]) .s
[1] (Float) 21097.4558874807
```

On applique un bloc sur chacun des entiers de 1 à 1000. Celui-ci calcule la racine carrée de chaque entier et somme le tout.

```
>seq(5) map(#sqrt) .s
[1] (List) [1, 1.4142135623731, 1.73205080756888, 2, 2.23606797749979]
```

#map récupère tous les résultats intermédiaires et en fait une liste.

```
>seq(10000) filter(#isEven) map(#sqrt) sum .s
[1] (Float) 333383.040171148
```

#filter va renvoyer une liste des entiers pairs de 1 à 10000, on calcule la liste des racines carrées puis la somme.

```
>func: deriv(f) { | x | #[ ->x x 0.0000001 + f perform x f perform - 0.0000001 / ] }
```

Cette fonction génère un bloc dynamique (qui dépend d'un paramètre d'entrée f) et va calculer la dérivée de cette fonction au point x fourni sur la pile. Par exemple pour f(x) = x^2 + x, la dérivée au point 2.0 sera :

```
>2.0 deriv(#[ dup sq + ]) perform .s
[1] (Float) 5.00000009395762
```

Et un peu de programmation parallèle

Oforth a été conçu pour gérer facilement le parallélisme.
Le modèle de programmation parallèle d'Oforth est :

- L'unité d'exécution parallèle est la task. Chaque task a sa propre pile de données et les objets créés dans une task ne sont pas visibles par les autres tasks.
- Un objet ressource de type Channel peut être partagé entre plusieurs tasks. Il s'agit d'une file FIFO dans laquelle une (ou plusieurs) task peut envoyer des objets (s'ils sont immutables, ce qui est le défaut) et à partir de laquelle une (ou plusieurs) task peut recevoir des objets.
- Lorsqu'une task est en attente d'une ressource (un channel vide par exemple), elle se met en WAIT en attendant que la ressource soit disponible. Une task en mode WAIT ne consomme pas de CPU et ne bloque pas de thread.

Comme les objets partagés par les différentes tasks ne peuvent être qu'immutables, il n'est pas nécessaire d'avoir de mécanisme de concurrence d'accès (verrous, sémaphores, ...), ni de mécanisme de copie d'objet.
Oforth permet de lister toutes les tasks qui tournent à un moment donné sur le système en utilisant #.w. Au démarrage, une seul task existe, celle qui exécute l'interpréteur, et cette task est en mode WAIT, en attendant qu'une commande soit entrée au clavier. .w donne :

```
WORKR STAT SCHEDULE  TASK                          WAIT
   1  RUN  default  interpreter
```

Cela signifie qu'il y a actuellement un worker (en fait un thread système) et que ce worker fait tourner une task (l'interpréteur), qui est en mode RUN. Cette tâche n'est pas en mode WAIT car l'interpréteur est en train d'exécuter une commande (#.w justement).
Pour créer une task, il suffit d'utiliser #& sur une fonction, une méthode ou un bloc. Celui-ci sera alors exécuté en parallèle. Créons une nouvelle task qui va attendre pendant 5 secondes et regardons ce que donne #.w

```
>#[ System sleep(5000) "Done" println ] &
ok
>.w
WORKR STAT SCHEDULE  TASK                          WAIT
   1 WAIT  default  #[ System sleep ( 5000 " println ]       _sleep
   1  RUN  default  interpreter
```

Maintenant deux tasks apparaissent. Elles tournent toutes les deux sur le même worker et la nouvelle task est en mode WAIT, en train d'attendre dans la fonction _sleep. Le worker, lui, est disponible pour exécuter d'autres tasks (ici #.w). Un channel permet d'envoyer des objets d'une task vers une autre. Nous pouvons créer un petit pingpong entre deux tasks.

```
func: pong(n, pingCh, pongCh) { | i | n loop: i [ pongCh send(pingCh receive) drop ] }
```

```
func: pingpong(n) {
| pingCh pongCh i |

  Channel new ->pingCh
  Channel new ->pongCh
  #[ pong(n, pingCh, pongCh) ] &
  0 n loop: i [ pingCh send(2) drop pongCh receive + ] println
}
```

La fonction #pingpong crée deux channels puis lance la task #pong en parallèle. Ensuite, n fois, elle envoie l'entier 2 sur le channel pingCh et attend le retour sur le channel pongCh. Ce retour est accumulé et le total est affiché à l'écran à la fin de la boucle (il sera donc égal à 2n).
La fonction #pong récupère un objet sur le channel pingCh et le renvoie sur le channel pongCh.
La valeur du paramètre n peut être élevée. Par exemple :

```
>#[ pingpong(1000000) ] bench
2000000
434248
```

Ici 1000000 d'allers-retours ont été réalisés en 440 millisecondes. Si on souhaite voir comment apparaissent ces tâches dans la console, il faut encore plus de temps et lancer #pingpong elle-même en parallèle :

```
>#[ #[ pingpong(10000000) ] bench ] &
ok
>.w
WORKR STAT SCHEDULE  TASK                          WAIT
   1 WAIT  default  #[ pong ( pongCh pingCh n ]           _channelReceiveTimeout
   1 WAIT  default  #[ #[ pingpong ( 10000000 ] bench ]      _channelReceiveTimeout
   1  RUN  default  interpreter
ok
>20000000
4189266
```

On voit qu'il n'y a toujours qu'un seul worker utilisé pour exécuter ces 3 tasks en parallèle. Pendant que #.w est exécuté, les deux autres tasks sont en WAIT et attendent de recevoir un objet sur les channels. Ce modèle à base de tasks permet de limiter les threads créés et les changements de contexte de threads. La création de tasks est beaucoup moins coûteuse que la création de threads. Et les changements de contexte de tasks sont bien plus rapides. Ce modèle à base de Channel qui ne peut transporter que des objets immutables permet d'éviter la copie d'objets envoyés entre chaque task (tout passe par adresse) et supprime tout besoin de synchronisation entre les tasks. Il est à noter aussi que l'utilisation d'une pile de données est un facteur supplémentaire de performance : chaque task a sa propre pile de données et ces données n'ont pas besoin d'être sauvegardées/restaurées à chaque changement de contexte d'une task à une autre. Plus la pile de données est privilégiée, plus les changements de contextes de tasks seront rapides. Essayons de générer un grand nombre de tasks :

```
func: tconsume(channel, n) { | i | 0 n loop: i [ channel receive + ] println }

func: tcreate(n) {
| i channel |
  Channel new ->channel
  #[ #[ tconsume(channel, n) ] bench ] &
  n loop: i [ #[ channel send(3) drop ] & ]
}
```

La fonction #tcreate crée un channel et lance la task #tconsume. Ensuite, elle crée n tasks, chacune envoyant l'entier 3 sur le channel.
La fonction #tconsume récupère tous les objets envoyés sur le channel et en fait la somme. Une fois le dernier objet reçu, elle affiche la somme finale.

```
>tcreate(1000000)
ok
>3000000
522570
```

Il ne serait pas possible de réaliser cela avec des threads. D'ailleurs, il ne serait déjà pas possible de créer un million de threads...
Notons pour finir que toutes les ressources Oforth sont totalement compatibles avec le modèle de tasks : les channels, mais aussi les sockets, la console, ... Faire attendre une task sur une socket jusqu'à ce qu'une requête arrive (pour gérer un keep-alive de 3 secondes par exemple) est tout à fait possible. Il n'y a pas de risque que cela bloque le worker qui pilote cette task. Celui-ci pourra exécuter d'autres tasks qui ne sont pas en attente.

Conclusion

Oforth est un langage un peu atypique qui nécessite un effort de prise en main. La pile de données, la RPN et son interaction avec le modèle objet, s'ils n'ont pas déjà été maniés, ne se laissent pas dompter facilement. Mais ensuite, on a affaire à un langage complètement dynamique, concis, tout objet, qui offre des caractéristiques de programmation fonctionnelle et qui simplifie la programmation parallèle.
Bien sûr, énormément de concepts n'ont pas été abordés (les classes, les méthodes, les propriétés, le dictionnaire, l'immutabilité, les schedulers, les emitters, les sockets, les packages, les exceptions, ...), mais cette introduction donne déjà un bon éventail des possibilités de ce langage.
Bonne découverte !

Ur/Web : un langage unifié pour les applications Web ?

Et si demain, les développeurs avaient un unique langage pour développer des applications Web ? Aujourd'hui, ils jonglent avec HTML, CSS, JavaScript, XML, JSON, etc. Sans compter les problèmes de maintenance des codes... Adam Chlipala, chercheur au MIT, propose une vision radicalement différente, en proposant un seul et unique langage : Ur/Web.

Ur/Web est un langage compilé en code natif. Il est capable d'empaqueter les données, génère de lui-même le CSS nécessaire pour l'affichage, injecte le bon code JavaScript, en supprimant les effets de bord. Ur/Web est décrit comme un langage coordinateur, c'est à dire, qu'il fédère plusieurs technologies et langages en exposant son langage aux développeurs. Il s'agit aussi d'un langage concurrent, taillé par les performances et la communication client – serveur. D'autre part, le langage doit assurer une meilleure sécurité.

Ur/Web hérite fortement de la programmation fonctionnelle et particulièrement du langage Ur. Ur est lui-même héritier direct du langage Haskell.

Son créateur avoue que l'apprentissage du langage nécessite un effort non négligeable. Malgré les atouts et les avantages d'Ur/Web, il y a de fortes chances que ce langage reste très marginal et limité à quelques développements, notamment universitaires et dans la recherche.

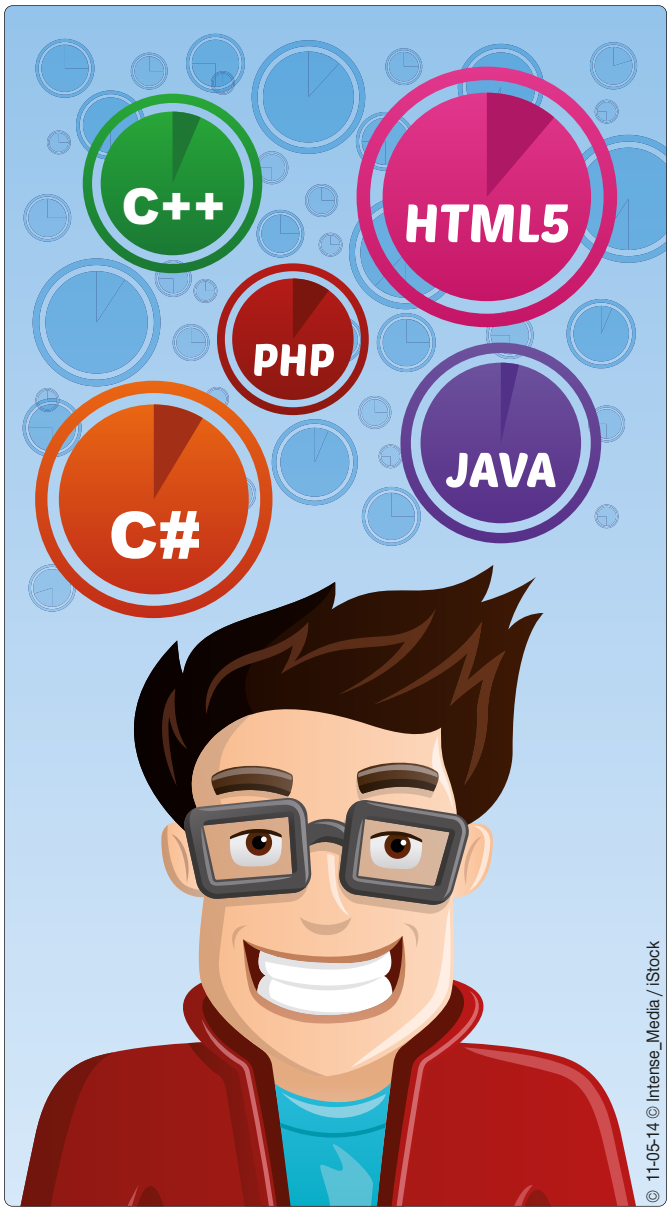
Pour en savoir plus :
http://www.impredicative.com/ur/

Développeur, le plus métier beau du monde

Le titre est volontairement provocateur et extrême. Nous aurions pu titrer « fier d'être développeur », « devenez développeur », etc. Il faut dire qu'on entend parler de développeurs un peu partout et parfois pour dire tout et un peu n'importe quoi.

Aux États-Unis, le développeur est devenu un enjeu national avec les initiatives pour découvrir la programmation. Apprendre le code aux enfants, sensibiliser aux métiers de l'informatique, miser sur la transition numérique et la société numérique. Cette couverture médiatique est nouvelle pour un métier que beaucoup de gens avaient du mal à comprendre ou à définir.

Les stéréotypes étaient, et sont toujours nombreux : asociaux, geek, nerd, des salariés un peu à part. Avec un régime alimentaire douteux : fraises tagada (un de mes vices j'avoue), pizzas, sodas, chips. Certaines séries TV n'ont pas aidé à améliorer notre côté sociable (IT Crowd, trop bon).



« *Le développeur est souvent artiste et passionné* »

C'est vrai que l'on peut rapidement tourner aux trolls quand des fanboys (ou fangirls) d'univers différents se rencontrent.

Bien entendu, on ne peut pas résumer le développeur et son job à cela. C'est vrai que nous sommes parfois extrêmes dans nos positions et nous aimons défendre nos convictions, nos technologies. Le développeur est (souvent) un passionné qui aime la technologie et le code.

Mais reconnaissons qu'il est souvent, et encore aujourd'hui, perçu comme un simple pisseur de codes, juste bon à parler techniques et à coder dans son coin. Une 5e roue du carrosse qu'il faut bien avoir dans les équipes. J'exagère un peu, volontairement.

Le métier de développeur était, globalement, peu valorisé. Faire une carrière de développeur en France ? Possible, mais il fallait en vouloir, car le must a longtemps été de devenir chef de projets ou architecte. Heureusement, l'image du développeur change. Il était temps.

L'économie numérique s'imposant aux entreprises, la multiplication des API, l'explosion des terminaux mobiles et l'invasion annoncée des objets connectés, tout contribue à la valorisation du développeur. Mais attention aux fausses promesses. Que cherche-t-on des armées de développeurs ou de bons développeurs ? La nuance est importante.

Dans ce dossier, nous avons voulu aborder de nombreux éléments : salaires, profils, carrières, compétences, formations, pourquoi devenir développeur et y faire carrière ?

Bac+x, fondamentaux, commandements

Pour simplifier à l'extrême, avant, vous aviez l'analyste-programmeur qui développait les projets selon le cahier des charges. L'arrivée du Web a introduit l'idée de webmaster. Aujourd'hui, finalement, nous pouvons dire que développeur veut tout dire et ne rien dire. Le métier même de développeur a considérablement éclaté, et s'est spécialisé, pour suivre, l'évolution des technologies et des plateformes. Même chose pour le Web. La notion de webmaster n'existe plus réellement...

François Tonic

A Programmez !, nous défendons le métier de développeur et nous essayons de le promouvoir, de l'expliquer depuis de nombreuses années. Oui, c'est un beau métier, oui on peut y faire carrière. Le métier commence à être mieux considéré par les entreprises, les écoles et même la société en général. Depuis un an, le développeur est devenu un « objet médiatique » notamment avec l'apprentissage du code à l'école, l'école 42 et les discours politiques.

Cependant, si le développeur est plus visible et est un composant de l'économie numérique, attention à ne pas tout confondre; le développeur peut rapidement se transformer en une mode médiatique, qui sera remplacé par autre chose. L'entreprise le considère moins comme un « pisseur de code », à un salarié un peu à part. Et la société va moins le regarder comme un asocial. On va rapidement lui coller l'étiquette de Geek ou de Nerd. Ce qui n'est pas totalement faux, mais il ne faut pas généraliser. Et surtout, on utilise les Geek et Nerd pour tout et n'importe quoi, malheureusement.

Autodidacte, bac+2, bac+5

Pour devenir développeur, il ne faut pas obligatoirement avoir un bac+5 / ingénieur. Vous pouvez opter pour des formations courtes :

- Bac +2 : BTS informatique puis un DUT ou IUT,
- Licence pro (bac+3) : licence professionnelle avec spécialisation programmation.

Attention : cela ne signifie pas que vous ne pouvez pas devenir développeur / développeuse si vous n'avez pas de diplôme ou de diplôme informatique. Vous pouvez parfaitement être développeur en vous formant, en participant à des formations de programmation, à passer des certifications, etc. Le développeur autodidacte existe toujours et heureusement. Mais, il devra prouver sa valeur et ses compétences, plus qu'un autre développeur; de nombreuses entreprises risquent d'être frileuses à prendre un autodidacte. De nombreuses annonces affichent bac+3 ou bac+5. Dans les PME, les SSII spécialisées sur une plateforme, une technologie (ou pure player technologique), l'autodidacte aura ses chances. Nicolas Sorel (reportage dans Programmez ! 179) est un exemple d'une success story avec Codes-Sources et Magma Mobile.

Des écoles en alternances ou l'école 42 permettent d'ouvrir plus largement les métiers du développement.

Nous reviendrons sur les écoles et les formations dans un dossier spécifique très prochainement.

Quelques fondations

Les fondations pour être un beau développeur sont nombreuses et très variées. Citons en vrac : maîtriser l'algorithmie, avoir rigueur et méthode, comprendre et maîtriser le C++ et les notions objets, avoir une vue générale du monde technologique, maîtriser les outils de développement, disposer d'une ouverture d'esprit et technique, maîtriser l'anglais, savoir travailler en équipe, savoir communiquer, savoir formuler et structurer une idée, connaître et appliquer les bonnes pratiques, s'auto-former tout le temps, aller aux conférences techniques, participer aux communautés, passer des certifications spécifiques, faire de la veille technologique. Cette liste est non exhaustive.

La maîtrise des algorithmes (et de la factorisation) est un élément important surtout quand vous allez vouloir trouver un stage ou un poste à l'étranger. Ne jamais oublier que l'informatique est un environnement « logique », binaire et mathématique.

L'Anglais est bien entendu indispensable. Beaucoup de documentations, de communautés sont en Anglais. Et dans de nombreuses entreprises, vous devrez communiquer en Anglais. On ne vous demande pas d'être bilingue mais de le lire, de le comprendre et de se faire comprendre. Il y a toujours une appréhension. N'hésitez pas à prendre des cours pour vous mettre à niveau.

5 commandements

En septembre 2014, j'avais animé une table ronde au salon LesJeudis.com sur le thème « le métier de développeur ». J'avais évoqué 5 commandements :

1. Montez en compétences,
2. Formez-vous,
3. Maîtrisez plusieurs langages,
4. Apprenez des communautés et des tendances,
5. Maîtrisez une ou plusieurs technologies.

Le développeur a parfois tendance à s'enfermer dans une technologie, une plateforme. Mais aujourd'hui, la technologie est tellement hétérogène qu'il est indispensable d'élargir ses compétences. Ayez toujours l'esprit ouvert. Regardez ce qui se passe dans les autres mondes.

La formation régulière est un moyen pour le développeur d'évoluer et de monter en compétences. Cette mise à jour constante peut être un « + » si vous souhaitez changer d'entreprise ou de poste en interne.

Installez, testez les nouveautés, les nouveaux outils. La veille technologique ne se fait pas une fois par mois quand vous avez le temps, mais tout le temps. Des entreprises, notamment les pures players, startups, donnent du temps aux développeurs pour leurs projets communautaires ou pour faire de la veille technologique.

Des salaires à géométrie (très) variable

Le développeur, comme nous l'avons vu, n'est pas un métier uniforme. Et les salaires sont à l'image du métier : divers, élastiques et variables. En 2014, Programmez ! avait publié plusieurs grilles salariales provenant de différentes études. Faisons une synthèse.

François Tonic

Expérience, employeur, profil, diplôme, île de France vs province, % variable du salaire, primes... La grille salariale est parfois compliquée à comprendre. Les critères sont nombreux. Et selon le type de développeur, le langage, tout peut changer. Et selon les études, vous ne trouverez pas la même chose. Essayons d'y voir un peu plus clair. Le salaire est fixe dans de nombreuses entreprises. C'est à dire que le montant reste identique. Dans certaines sociétés, une part variable pourra être introduite soit sur x % du salaire, soit en plus du salaire. Cette variabilité concerne essentiellement les primes d'objectifs (si les développements et projets ont été un succès, réalisés selon les plannings), et les primes de résultats (de l'entreprise). L'écart Île de France – Province est toujours valable, même si certaines régions sont très actives et se rapprochent des salaires parisiens. En revanche, des agglomérations et des départements moins dynamiques joueront négativement sur les salaires.

Attention : *comme vous le constaterez par vous-mêmes, les sources utilisées indiquent parfois des écarts assez significatifs. Si le cabinet de recrutement est « haut de gamme », les salaires seront généralement plus élevés.*

Exemple 1 : le développeur PHP débutant

En regardant les différentes études sur les salaires des métiers du numérique, on constate parfois des écarts très importants.

Tableau 1 : salaire d'un développeur PHP débutant (0 à 2 ans expérience)

	salaire moyen annuel brut, milliers €
Urbanlinker	25-30 k€ 0 à 1 an 30-34 k€ 1 à 2 ans
AFUP / Agence-e (baromètre salaires PHP 2014)	31 k€ (salaire médian)
Hays (étude 2014, profil général)	25/30 k€ (analyse programmeur) 36/40 k€ (ingénieur développeur)

Il est parfois très difficile de comparer les études salariales car le contour des métiers et les intitulés ne sont pas identiques. Cependant, nous pouvons estimer qu'un développeur débutant PHP démarrera sa carrière aux alentours de 30 000 € / bruts, sur la région île de France, moins en province (en général), soit 2 500 € brut / mois.

Mais, ce salaire médian va dépendre de votre diplôme (par exemple : bac+2, bac+5 niveau ingénieur), du profil de la société et si vous êtes en Île de France ou en province. Selon l'étude agence-e / Afup, l'écart pour un développeur débutant varie de 25 à 36 k€ (incluant le fixe et les primes diverses). La startup paie le moins (29 k€).

Exemple 2 : le développeur PHP avec compétence

Restons sur PHP. Un développeur PHP, même débutant, ayant une compétence spécifique, peut justifier un salaire plus élevé. Sa compétence a donc un impact direct, pouvant dépasser 10 %. Un développeur PHP généraliste avec 3-5 ans d'expérience sera aux alentours de 32-33 k€, une forte compétence Zend ou Symfony sera intéressante.

Tableau 2 : salaire d'un développeur PHP débutant (0 à 2 ans expérience)

En K€/an	0 - 2 ans	3 - 5 ans	6 - 10 ans	10 ans et +
Zend Framework	32,5	36	41,1	45,8
Symfony	31	37,2	42	45
Framework proprietaire	33	35	33,5	45
Drupal	28,8	32,6	39,3	47,3
Wordpress	30	31,8	38,4	39,7
Autre Framework ou CMS du marche	32	29,8	37,8	40,1
Solution E-Commerce	30	31,8	38,4	39,7
Multi-specialiste	32	34,5	40,5	43,6
Autre	30,3	34	40	48,5

Source : Agence-e / AFUP.

Vous l'aurez compris : une compétence, une spécialisation peut vous donner des arguments et surtout, de vous différencier par rapport à d'autres développeurs qui ne l'auront pas forcément.

Les études d'Urbanlinker sont différentes du baromètre Agence-e / AFUP, comme le montre le tableau 3. Cela prouve qu'il est très difficile d'établir des moyennes et des tendances claires.

Tableau 3

	PHP	PHP + Framework MVC(Zend, Symfony, ...)
Débutant 0 à 1 an	25-30 K€	30-35 K€
Intermédiaire 1 à 2 ans	30-34 K€	35-40 K€
Confirmé 2 à 4 ans	35-40 K€	40-45 K€
Sénior 4 à 6 ans	40-45 K€	45-50 K€
Expert / Architecte 6 ans et +	45-53 K€	50-70 K€
Chef de projet 8 ans et +	-	45-57 K€

Source : Urbanlinker.com / salaire 2014. En milliers € / brut par an.

Exemple 3 : le développeur mobile

Avec l'explosion des terminaux mobiles (smartphones, tablettes), les besoins en apps ne cessent de croître. On y trouve plusieurs profils :

- Développeur mobile natif (Java sur Android, Objective-C / Swift sur iOS),
- Développeur mobile polyvalent et/ou multiplateforme (par exemple avec Xamarin),
- Développeur mobile orienté Web.

Une étude Careerbuilder estime qu'un développeur mobile sera entre 30 et 45 k€ mais le développeur mobile Web peut monter jusqu'à 50 k€.

Dans un rapport de Robert Half (étude rémunération 2015), nous démarrons à 40 000 €, avec un maximum à 55 000 €.

Tableau 4 : salaire d'un développeur mobile

	Applications mobiles hybrides	Applications mobiles natives
Débutant 0 à 1 an	32-36 K€	34-40 K€
Intermédiaire 1 à 2 ans	36-42 K€	38-44 K€
Confirmé 2 à 4 ans	42-46 K€	43-50 K€
Sénior 4 à 6 ans	46-50 K€	50-60 K€
Expert / Architecte 6 ans et +	-	60-80 K€
Chef de Projet 4 à 8 ans	-	55-65 K€

Source : Urbanlinker.com / salaire 2014. En milliers € / brut par an.

Attention cependant, de plus en plus de développeurs arrivent sur le développement mobile, pouvant faire stagner ou baisser les salaires. Les profils atypiques (ex. : développeur mobile polyvalent Android / iOS / Windows Phone) sont des profils recherchés.

Exemple 4 : développeur « standard »

Le développeur « standard » est celui qui code en Java, .Net, C, Python, C++, souvent pour des applications desktop ou Cloud, avec du Web et du Mobile. Il n'a pas forcément une dominante. Les profils sont ici parfois très variés : autodidactes, indépendants.

Tableau 5

	JAVA	.NET	RUBY
Junior 0 à 1 an	35-39 K€	35-39 K€	32-36 K€
Intermédiaire 1 à 2 ans	39-42 K€	39-42 K€	36-45 K€
Confirmé 2 à 4 ans	42-46 K€	42-46 K€	45-50 K€
Sénior 4 à 6 ans	46-52 K€	46-55 K€	50-55 K€
Architecte / expert 6 ans et +	48-70 K€	55-85 K€	-
Chef de projet 4 à 8 ans	48-62 K€	48-65 K€	-

Source : Urbanlinker.com / salaire 2014. En milliers € / brut par an.

Tableau 6

ingénieur développement / expérience	en euros / brut / annuel
2 – 7 ans	40 – 57 000
7 – 15 ans	56 – 69 000

Source : Robert Half, salaires 2015

Tableau 7

	0-2 ans *	2-5 ans	5-10 ans	+ 10 ans
analyste-programmeur	25/30	30/35	35/40	40-45
ingénieur développeur	36/40	45/50	55/65	65/75

** expérience*
Source : Hays (étude 2014, profil général). En milliers € / brut / an.

Là encore, difficile de donner une idée précise du salaire moyen mais en général, on estime une fourchette de 30 – 33 000 € / an en Île de France, -30 000 € en province. Et encore, certains profils seront assez largement sous les 30 k€. Selon le langage, la plate-forme, le salaire peut subir une plus ou une moins value. Jobprod (salaires des développeurs Web 2014) propose une infographie intéressante : **Fig.A.**

Exemple 5 : développeur Web

Les métiers du Web sont très éclatés. Le webmaster n'existe plus en tant que tel, ni véritablement le développeur Web. Avec la multiplication des technologies, plateformes et des outils, les profils se sont multipliés : intégrateur, Web designer, directeur artistique, trafic manager, community manager, etc.

Tableau 8 : intégrateur HTML (Robert Half, salaires 2015)

expérience (ans)	salaire (€, brut / annuel)
0-2	25 – 36 000
2 – 5	36 – 46 000
+5	46 – 51 000

Tableau 9 : jobprod (les salaires des développeurs web en 2014, Île de France)

expérience (ans)	HTML / CSS (€, brut / annuel)	JavaScript (€, brut / annuel)
-2	30 000	31 600
2-5	39 900	39 200
+5	49 300	50 700

SALAIRES DANS LE JEU VIDÉO

Le syndicat national du jeu vidéo, avec Opcalia, avait publié en 2012 un référentiel détaillé sur les métiers et les salaires. Cette étude, n'a pas été, à notre connaissance, profondément réactualisée.

Metier	Qualification privilegiée	Salaire minimum	Salaire maximum
Directeur technique	Bac +5 et plus	(20 k€) 44 k€	54 k€ (110 k€)
Lead programmeur	Bac +5 et plus	(18 k€) 32 k€	42 k€ (67 k€)
Programmeur moteur	Bac +5 et plus	(18 kE) 30 kE	40 kE (60 kE)
Programmeur gameplay	Bac +5 et plus	(15 kE) 27 kE	37 kE (60 kE)
Programmeur specialise (IA, outils, physique, etc.)	Bac +5 et plus	(18 kE) 27 kE	40 kE (60 kE)

(n) salaire minimum et maximum observe aupres des entreprises ayant participe a l'enquete
Source : SNVJ / Opcalia, 2012

Nous reviendrons sur les métiers du jeu vidéo dans Programmez! 183.

Tableau 10 : développeur front end Web

Rémunération	Intégration / HTML5 / CSS3 / Javascript	HTML5 /CSS3 / Javascript / New frameworks JS + responsive Web design	Dev fullstack JS NodeJs + framework front (Angular, backbone)
Débutant 0 à 1 an	28-32 K€	33-37 K€	36-38 K€
Intermédiaire 1 à 2 ans	32-38 K€	37-44 K€	38-45 K€
Confirmé / Chef de projet 2 à 4 ans	38-42 K€	44-49 K€	45-50 K€
Sénior 4 ans et +	42-45 K€	49-46 K€	50-65 K€

Source : Urbanlinker.com / salaire 2014. En milliers € / brut par an.

L'intégrateur, ou développeur Web est aujourd'hui un métier de base du Web. Les profils plus complets, avec par exemple des compétences en responsive design ou côté serveur) se valorisent mieux comme l'indique le tableau 10. L'expérience sera aussi un élément important.
Des profils recherchés sont à un autre niveau :

- Développeur Rails : un profil + 5 ans d'expérience peut espérer entre 50 et 80 000 € (source : chooseyourboss)

Exemple 6 : bon pour être indépendant ?

Si vous ne voulez pas être en entreprise, SSII ou chez un pure player, il reste l'option : indépendant/free-lance. Être free-lance n'est pas toujours évident, il faut savoir se vendre, gérer la paperasse, la comptabilité, payer les charges, etc. Mais vous êtes votre patron et choisir des projets qui vous plaisent. Hopwork a sorti un baromètre de l'emploi free-lance pour l'année 2014 (https://www.hopwork.com/stats/barometer/2014). Hopwork référence 3958 free-lance et +10 000 missions.

Tableau 11 : tarifs journalistes par langage et par ville.

Langage	Recherches (en %)	Tarif moyen	Paris	Marseille	Lyon	Toulouse
PHP	40,2 %	327 €	390 €	340 €	372 €	384 €
java	34,3 %	444 €	535 €	477 €	458 €	501 €
ruby	7,8 %	448 €	500 €	425 €	390 €	575 €
JavaScript	5,4 %	386 €	441 €	359 €	412 €	399 €
python	5,1 %	428 €	516 €	402 €	445 €	405 €
scala	3,2 %	576 €	618 €	-	600 €	585 €
c++	2,4 %	369 €	415 €	350 €	425 €	322 €
c#	1,6 %	332 €	419 €	440 €	391 €	510 €

Ce baromètre évoque aussi les bases de données. Par exemple, le tarif moyen pour un expert MySQL s'élève à 346 €, contre 462 € pour Oracle, 638 € pour Cassandra ou encore 483 € pour MongoDB. Certaines technologies récentes tirent les tarifs vers le haut. Et les développeurs compétences sont encore peu nombreux.

Peut-on faire une carrière de développeur en France ?

Maillon essentiel de la réussite des projets informatiques, et du fait de l'évolution rapide des technologies, le développeur doit faire preuve de capacités d'adaptation certaines pour rester performant. Alors qu'il devrait logiquement bénéficier de la reconnaissance accompagnant son niveau de compétences, le développeur souffre d'un manque de considération important. Puisqu'il semble s'agir d'un problème bien français, la question mérite d'être posée : peut-on faire une carrière de développeur en France ?

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

Un problème de formation

Le métier de développeur exige en premier lieu de solides qualités techniques de par la complexité toujours plus grande des applications d'entreprise et l'évolution rapide des technologies. De fait, les filières techniques supérieures devraient être mieux valorisées, puisqu'elles permettent dès le début des études supérieures de coder de manière intensive. Las, en France le T des filières de type BTS et IUT semble être la première lettre d'un mot tabou qui conduit tous les conseillers d'orientation à déconseiller consciencieusement les filières techniques aux lycéens les plus prometteurs qui souhaitent devenir développeurs. Ainsi, la plupart du temps ils sont dirigés vers l'université ou des écoles d'ingénieur spécialisées qui n'auront de cesse de prôner l'importance de viser un poste de chef de projet au plus vite, au détriment du métier de développeur.

Même pour les écoles où l'étudiant pratique de manière intensive sur de nombreux projets de mise en situation, le rôle de chef de projet est toujours placé comme une sorte de Graal dont l'obtention est essentielle à la réussite d'une carrière. En somme, ne pas être chef de projet et rester développeur passé un certain cap serait symbole d'échec ! Beaucoup d'étudiants sont de fait convaincus que dès la sortie de l'école, ils sont fin prêts à devenir chef de projet (junior dans le pire des cas) et, de fait, délaissent le développement. Cette idée est confortée par le côté presque péjoratif du mot développeur en France. On préfère ainsi largement le titre plus pompeux d'Ingénieur d'Etudes qui, s'il est justifié, ne doit pas pour autant donner un côté rabaissant ou avilissant à la qualification de développeur.

La valorisation du côté technique essentiel au métier de développeur doit donc débuter dès la formation avec, si possible, une intervention plus importante de professionnels du secteur qui ne seraient pas seulement issus de SSII bien souvent en quête de publicité et de futurs « talents » pas chers. En outre, il est primordial que les universités prennent enfin en compte le bassin d'emploi dans lequel elles se trouvent afin d'adapter de manière adéquate le contenu de leurs formations. En effet, quel est l'intérêt de faire du C/C++ ou du Prolog en dernière année d'un Master Professionnel lorsque la majorité des emplois proposés aux étudiants en sortie concerneront des environnements Java/JEE ? Pour mettre en place cette adaptation des formations au bassin d'emploi local, il est nécessaire de recourir à des intervenants extérieurs au fait des technologies, là où bien souvent les enseignants d'université ne peuvent malheureusement pas rester à la pointe. Ceci conduisant à privilégier au sein des programmes l'enseignement de matières pour lesquelles les enseignants sur place possèdent déjà des connaissances. Bien entendu, cela a un coût et constitue une problématique dépassant largement le cadre de la formation informatique en université.

Des SSII prépondérantes

L'autre problème majeur lié à la condition des développeurs provient des sociétés de service en ingénierie informatique plus connues sous le nom de SSII. Si elles se targuent d'être à la pointe de la technologie avec un niveau d'expertise frisant toujours l'excellence, elles n'ont parfois d'autre but que de maximiser à tout prix la rentabilité, quitte à renoncer à la création de valeur ajoutée. En France, elles ont fermé des grands comptes, barrant les entrées en direct aux indépendants. Les arguments justifiant ce système sont une diminution des risques pour les grands comptes grâce à des SSII pouvant remplacer facilement l'ingénieur vendu en prestation. Ce système franco-français contraint les développeurs à passer, souvent, par des SSII pour travailler chez des grands comptes, que ce soit en tant que salarié, ou en tant qu'indépendant. Certaines sociétés se spécialisant dans ces placements. On peut même pousser le trait plus loin en comparant ces sociétés de placements à de simples agences d'intérim pour développeurs. Pour les salariés en SSII, ce système conditionne souvent leur embauche à la seule réussite de l'entretien auprès du client final.

Les salaires proposés par les SSII aux développeurs en général desservent le métier de développeur. Comment bien se sentir en tant que développeur confirmé avec un salaire plafonné à 40K€ par an en France ? La reconnaissance de l'importance des développeurs confirmés dans la réussite des projets est absente, et l'on peut s'en rendre compte chaque année lors d'entretiens annuels où les seules perspectives intéressantes d'évolution mises en avant concernent des postes de chef de projet, voire

Comment percevez-vous le développeur ?

Une simple mode	5%
Un pisseur de code et c'est tout	8%
Un passionné et un artiste de la technologie	71%
Un job comme un autre	17%

Source : sondage express Programmez!, 393 votes.

de consultants fonctionnels. Les plus résistants et les plus habiles pouvant à l'occasion accéder à des postes d'architectes, qui paradoxalement, les éloignent également du code, alors qu'un architecte ne devrait être en réalité rien de plus qu'un développeur senior de grande qualité.

Une pénurie de développeurs ?

Devant le peu de considération dont bénéficient les développeurs, il ne faut pas s'étonner ensuite si bon nombre d'entre eux ne répondent pas à des offres de recrutement exigeant tout un tas de compétences de haut niveau pour un salaire trop bas. Devant le peu de candidats pour ce type d'offres, certaines SSII n'hésitent pas à se plaindre d'un marché du développement en pleine pénurie ! Le tout en expliquant qu'il faut inciter les jeunes générations à aller vers ces secteurs forts en emploi. Le marché des développeurs ne souffre en réalité pas de pénurie en France mais cela pourrait arriver un jour si le métier de développeur n'est pas mieux valorisé, que ce soit au sein des universités, ou au sein des entreprises. C'est plutôt que les développeurs n'ont pas spécialement envie de travailler pour des boîtes tirant les salaires vers le bas, tout en exigeant des compétences de haut niveau.

Un nouveau modèle de SSII

Des SSII commencent à prôner une vision du métier plus valorisante avec la création de valeur ajoutée au cœur des préoccupations, et avec une prédominance des projets au forfait, où les fortes capacités techniques sont essentielles. De fait, elles soignent leurs développeurs en les faisant progresser, et en les maintenant au fait des nouvelles technologies en organisant des ateliers dédiés sur des thématiques nouvelles technologies aussi bien en soirée qu'en journée. En outre, elles poussent leurs développeurs à participer à des réunions communautaires où ils apprennent mais peuvent également partager leur vision du métier.

Embauche chez un client final

Une porte de sortie pour le développeur peut venir d'une embauche chez un client final. En effet, le turnover y est moins élevé du fait des besoins de capitalisation sur la durée auxquels ils font face. Le statut de développeur peut quelquefois y être mieux valorisé. Las, des obstacles viennent encore se dresser sur la route. Satisfaits du fonctionnement des SSII, les clients grands comptes ne recrutent que peu de collaborateurs, préférant la prestation de services. Et pour le peu d'élus qui arriveront à être embauchés, il est fort à parier que les mêmes propositions de basculer vers du management soient au rendez-vous à terme.
Le second obstacle est lui commun aux grands comptes et clients finaux de taille plus modeste. Il concerne le passé en SSII du développeur qui peut constituer un obstacle à une embauche puisque ce dernier aurait de fait une culture trop services inadaptée à un client final.

30 ans l'heure du choix

Quelle que soit sa société, le temps passe vite pour le développeur et son management ne cesse de lui faire comprendre qu'une prise de responsabilités est une obligation pour évoluer. Ainsi à 30 ans vient l'heure du choix ! Basculer du côté management avec à la clé un poste de chef de projet assorti d'une belle rémunération (et enfin réussir sa vie en pouvant acheter sa Rolex ...) ou bien demeurer développeur sans réelles perspectives. Que choisir ? A voir les quelques résistants qui sont restés dans le développement avec toutes les peines du monde pour trouver des missions, le développeur fait vite son choix. Il doit faire une croix sur son IDE favori et passer à la gestion d'une pile de mails plutôt indigeste, à la conception de présentations Powerpoint, et à la découverte en profondeur d'Excel qui devient rapidement son nouveau compagnon de route.
Il faut dire que la France souffre d'un grave problème de considération des seniors en général et cela s'applique évidemment pour les développeurs. Les clients finaux poussant toujours pour des profils plus jeunes (et plus dynamiques paraît-il) lors des demandes en prestation de services. D'autre part, même en interne ils sont bien souvent pris de haut par de jeunes chefs de projets certains d'avoir déjà réussi leur carrière, contrairement à eux qui végètent toujours dans le développement. Pire encore, et c'est là l'ironie du développeur, certains juniors raillent ces seniors, pas toujours à la pointe des dernières nouveautés, ne se rendant même pas compte que le même sort les attend dans un futur pas si lointain !
En regardant la situation à l'étranger, on se rend compte que ce problème est bien franco-français. Prenons l'exemple des Etats-Unis. Là-bas, la scission management / développement est bien réelle entre 2 métiers totalement différents en termes de compétences. Etre développeur est un métier et être manager en est un autre. Le développeur programme quand le manager gère. De fait, il est possible de faire une carrière dans le développement avec une réelle valorisation. Il n'est ainsi pas rare que des développeurs émergent à plus de 150K$ par an ! Même si ce montant est à relativiser compte tenu des différences de prélèvements entre la France et les Etats-Unis, il existe bien un écart réel. Un développeur de qualité, qu'il soit sénior ou non, n'a pas de prix et les compétences qu'il amène sont essentielles à la réussite des projets de l'entreprise. On est bien loin de cette situation en France avec un salaire tenant compte avant tout des rapports hiérarchiques au sein de l'entreprise conduisant à des différences parfois abyssales entre managers et développeurs, quand bien même tous deux sont nécessaires à la réussite des projets.

Un métier sous-évalué

Au fond, la source du problème est sans doute la méconnaissance du métier de développeur par les managers. Pour certains, l'activité du développeur se limite à "pisser du code" à longueur de journée pour parler vulgairement. Le métier pâtit de cette image qui conduit à une mauvaise considération sociale et salariale. La programmation serait une tâche honteuse alors même qu'elle est le cœur d'un projet informatique. On tombe en plein paradoxe avec un titre de développeur prenant tout à coup presque une consonance péjorative !
L'enjeu est donc bien de mettre en lumière l'importance du développeur dans la réussite des projets. Le développement est une discipline complexe qui doit constituer un véritable choix de carrière. En effet, l'évolution rapide des technologies nécessite un investissement constant de veille et de remise en question pour s'adapter aux nouvelles méthodologies et pratiques. Lorsqu'il est choisi en toute connaissance de cause, il s'agit d'un métier passionnant où le plaisir d'apprendre est un véritable moteur. Etre développeur, c'est devoir produire dans les temps et c'est surtout un métier où il n'est pas possible de se cacher : le code doit être produit pour que les fonctionnalités devant être réalisées soient fonctionnelles.
L'image du geek asocial véhiculée par certains est aux antipodes du rôle joué par le développeur en entreprise. Si le métier de développeur nécessite bien une forme de talent, cela ne suffit pas ! Il faut savoir travailler en équipe, communiquer efficacement et s'organiser pour assurer la

réussite d'un projet. Un développeur est un artisan se bonifiant au fil des années et au gré de ses expériences. Enfin, et c'est le point central, la tâche du développeur ne s'arrête pas au code. Il doit avant tout être capable de prendre en entrée des besoins pas toujours clairs et les traduire en fonctionnalités sur un projet. Le développement est un processus cognitif complexe nécessitant un grand savoir-faire.

Une lueur d'espoir

A l'heure actuelle, il semble impossible de faire une carrière de développeur en France en SSII ou chez certains clients finaux, à moins d'accepter des conditions sociales et salariales minimales. En revanche, des portes de sortie existent en France, et vont s'ouvrir en grand dans le futur. Tout d'abord, il est bon de signaler que certaines SSII mettent en place des modèles plaçant le développeur au cœur de leur stratégie. Il en résulte une véritable valorisation aussi bien salariale que sociale, avec des ateliers ou réunions permettant aux développeurs de rester à la pointe de la technologie et d'échanger sur le métier. Ces sociétés vont même jusqu'à inviter leurs développeurs à des conférences de haut niveau qui se révèlent bien plus profitables que bon nombre de formations. La mise en place intensive des méthodes agiles dans leur fonctionnement n'est sans doute pas étrangère à cette mentalité. Néanmoins, il est dommage que ce type de SSII soit peu répandu hors région parisienne pour le moment. Dans la même veine, les startups développant des produits innovants sont une voie à explorer pour les développeurs, puisque par leur petite taille, chacun y joue un rôle crucial. En outre, elles présentent l'avantage d'utiliser bien souvent les dernières technologies en vogue, ce qui s'avère très motivant pour un développeur. Même si la sécurité d'emploi au sein d'une startup est moins grande que dans une grande société, le fait de pouvoir être un acteur clé de la réussite de sa société est un véritable challenge qui vaut sans doute le coup.

Enfin, l'indépendance semble être une voie à privilégier pour pouvoir faire une carrière de développeur en France. Etre à son compte permet de gérer sa carrière en choisissant ses missions et en gérant soi-même la relation client. La grande autonomie offerte par ce mode de fonctionnement nécessite une grande rigueur au quotidien. Il faut avant tout aimer son métier de développeur pour rester au fait de la technologie en assurant un fort travail de veille technologique. Néanmoins, ce n'est pas vraiment une contrainte pour les développeurs de qualité qui effectuent déjà ce travail en étant salariés. La souplesse conférée par ce statut permet, en tous cas au développeur, d'augmenter significativement ses revenus. A moins d'être en région parisienne et de bénéficier d'une grande variété de clients, il nécessite quelquefois de se déplacer pour trouver des missions intéressantes venant améliorer son profil.

Conclusion

Poser la question de la possibilité ou non d'effectuer une carrière de développeur en France en 2015 conduit à s'interroger sur l'ensemble de l'écosystème entourant le monde de l'informatique. Cela va de la formation ou le manque de souplesse des contrats de travail à durée indéterminée qui nuisent dans un certain sens aux possibilités d'embauche au sein de clients grands comptes. Ce questionnement nous amène ensuite tout naturellement au lobbying quasi constant qui est fait pour les rôles de managers qui bénéficient d'une meilleure image au sein de notre société. Alors que l'employabilité des séniors est au cœur de nombreux débats sociétaux, il serait grand temps de prendre en compte le métier de développeur comme un métier à part entière. De fait, les développeurs pourraient bénéficier de réelles perspectives d'évolution de carrière en continuant d'exercer leur expertise.

En attendant de possibles avancées du système, le développeur peut trouver dans le statut d'indépendant la reconnaissance de ses qualités techniques. Il reste maintenant à faire évoluer les mentalités pour sortir du système existant entre SSII et grands comptes afin d'ouvrir les portes aux indépendants. Cette ouverture permettrait à coup sûr de replacer la création de valeur ajoutée au centre des priorités des sociétés de services en ingénierie informatique.

« Développeur : le plus beau métier du monde » ? Pour moi assurément oui, même si tout n'est jamais tout blanc ou tout noir.

Stéphane Sibué
Directeur technique
chez GUYZMO
stephane.sibue@guyzmo.fr
www.guyzmo.fr

GUYZMO

J'ai 46 ans et je développe depuis l'âge de 12 ans. Je suis donc un « dévosaure ». Mais j'ai une chance incroyable, j'ai réussi à faire de ma passion mon métier, et, surtout, j'ai réussi à faire en sorte qu'après presque 35 ans j'ai toujours autant de plaisir.

Pour cela, il faut à mon sens deux ingrédients. Tout d'abord une passion sans limite, et aussi la capacité à changer de sujet régulièrement, pas seulement pour rester au top de ce qui est demandé, mais aussi pour ne pas entrer dans le piège de l'expertise unique, synonyme d'ennui à plus ou moins long terme. Développeur, quel beau métier, à condition d'aimer créer, d'aimer comprendre la manière de travailler des autres, d'aimer poser des questions (beaucoup de questions) et surtout d'aimer se former continuellement.

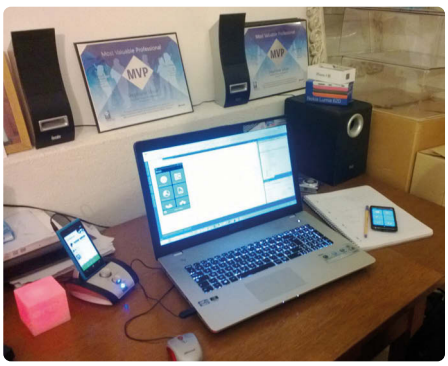
C'est un exercice simple quand vous avez 20 ans, ils l'est moins à 40 ! Parfois je regrette tout de même qu'une techno à peine « sèche » soit remplacée par une nouvelle qu'il faut s'approprier pour avancer (je pense entre autre à WinRT qui vient bousculer les développements Silverlight sous Windows Phone et qui est incontournable pour réaliser des applications universelles).

Coté formation je n'ai qu'un BTS informatique de gestion. Il me fallait un diplôme officiel au cas où. Comme j'ai créé ma société je n'ai pas eu à me battre pour entrer chez un employeur, car l'employeur c'était moi du coup ! Je pense qu'aujourd'hui il ne faut pas hésiter à viser plus haut, à essayer, dans la mesure du possible de décocher des diplômes plus « porteurs » car la concurrence est de plus en plus rude.

A titre professionnel, j'ai commencé à développer des applications en C pour MSDOS, puis je suis passé au développement pour Windows, d'abord en C++, puis

pour augmenter ma productivité je suis passé à ce nouveau langage (en 1992) qu'était Visual Basic. Avec cet outil j'arrivais à sortir les applications 3 fois plus vite, alors j'ai très vite adoré développer avec. C'était un paramètre très important car, au sortir de mon armée (eh oui, j'ai fait l'armée moi monsieur), j'ai créé ma société et très vite nous avons travaillé dans l'industrie, le tourisme et le médical. Quelle belle aventure !

En fait j'ai un problème. Si un projet dure plus de 6 mois, je commence très sérieusement à m'ennuyer. A cause de cette particularité génétique, je privilégie les projets courts mais apportant un défi technologique (c'est plus drôle). C'est pour cette raison qu'au bout de 8 ans de développement pour PC dans l'industrie, le



tourisme, et le médical je me suis tourné vers la mobilité. A l'époque personne n'y croyait et tout le monde me disait « c'est une mode » ou encore « qu'est-ce que tu veux en faire de tes petites machines ? ». Hommes de peu de foi, regardez maintenant comment ça se passe. N''avait-il pas raison le Steph avec ses petites machines ? C'est là que mon expertise de développement Windows pour PC s'est transmutée avec le temps en expertise de développement mobile (et hop, 2 cordes à mon arc de développeur). Aujourd'hui, je fais de plus en plus de missions d'expertise pour aider les jeunes entreprises et les startups à se lancer dans le développement mobile; je suis très fier de pouvoir dire que je sais développer des applications natives pour Android, iOS, Windows Phone, en plus de savoir développer des applications .NET pour PC. Ça fait plein de petites cordes à mon arc qui vont finir par constituer une vraie harpe à la longue. Et puis maintenant il y a le Cloud, les objets connectés, et une quantité infinie de nouvelles technologies qui vont sortir au fil du temps. On ne va pas s'ennuyer, ça c'est certain !

Pendant mes 25 ans de développement professionnel, j'ai pu m'essayer à différentes formes d'organisations. J'ai commencé comme gérant de société (on a été jusqu'à 15 personnes au plus haut de l'activité), puis comme salarié d'un grand groupe (250 personnes), puis « simple » salarié d'une petite structure (mais les projets étaient trop bien). Donc j'ai été salarié, travailleur indépendant, autoentrepreneur (on peut cumuler les statuts c'est parfois pratique). Tout ça pour dire qu'il n'y a pas de bon et mauvais statut dans ce métier, tout dépend de ce que vous recherchez (sécurité versus liberté, expertise centrée ou large, etc.).

Pour moi, vous l'avez compris, c'est une véritable histoire d'amour. Mais j'ai remarqué avec le temps qu'un développeur qui conçoit de belles applications est forcément habité par la passion du code, sinon ça ne marche pas. Développer c'est avant tout un acte de création, une forme d'art, et sans inspiration, sans passion, il n'y a pas d'art. Donc oui, développeur, quel beau métier !

"À 10 ans, j'ai vu une maquette de bateau en QBasic"

Maxtime Audouin
Magma Mobile

Un jour, quand j'avais 10 ans, mon père dessina une maquette de bateau en QBasic, j'ai vu l'écran et j'ai dit :
— Papa, comment fais-tu ça ?
Une vocation est alors née en moi. J'ai alors commencé à programmer en Qbasic, ensuite j'ai appris le C (avec un livre), puis quand on a eu Internet, le JavaScript et le PHP. Grâce au concours prologin, j'ai découvert tout un tas de problèmes d'algorithmes, présentés de façon ludique. J'ai goûté à un grand nombre de langages pour finalement approfondir le caml et le Java. J'ai fait un an d'école d'ingénieur puis deux ans de fac avant de me trouver un emploi d'assistant de recherche dans une entreprise qui développait un compilateur en ocaml.

Plus tard, je suis parti pour Magma Mobile, mon actuel employeur.

J'ai continué à coder chez moi, des jeux, des sites Web, des algorithmes, je pense que développer chez soi est la meilleure formation : on apprend à se sortir de n'importe quelle situation. On peut apprendre seul avec quelques livres et Internet, on s'installe un Linux et quelques compilateurs, on choisit son langage, son IDE, on apprend à faire un Makefile et c'est parti ! Parmi les livres qui m'ont le plus marqué, je recommande le Cormen, c'est une référence pour étudier les algorithmes. Pour être un bon développeur, il faut bien connaître ses outils : son système, son IDE, la chaîne de compilation, son langage, son gestionnaire de version, sans négliger tout ce qui tourne autour des connaissances (scientifiques ou non) qui vont avec le projet, et là, les sujets ne manquent pas : 3d, moteur physique, système de type, réseau, interface utilisateur, intelligence artificielle, algo, formats de fichiers, théorie de la communication, parsing, statistiques, etc.

Là où je travaillais avant, on utilisait Git et Linux. Dans mon poste actuel, j'ai du Mac, du Windows, du Linux, je programme avec 4 langages différents. Je ne travaille que depuis 6 ans et j'ai eu à configurer des machines, faire de l'IA, de l'algo, des bouts de moteurs physiques, un compilateur, des sites Internet, des jeux Web, IOS, Android, WP8, de la 3d, j'ai utilisé Git, svn, tfs. C'est vraiment un métier où les tâches répétitives n'existent pas, les missions changent tout le temps. Quand une tâche est répétitive, on l'automatise !

Quand on programme, on a envie d'arriver au résultat : de voir le programme tourner, et quand ça commence à fonctionner, on est fier d'avoir réussi. Personnellement, je suis fier d'avoir fait une IA d'échecs pour Magma Mobile et un compilateur Metalang. Quand on travaille dur sur un problème difficile et qu'on arrive enfin à voir les résultats, c'est un plaisir incroyable; quand en plus on arrive à avoir quelques utilisateurs ou joueurs, c'est encore mieux.

Si on considère la société comme étant l'ensemble des interactions au sein d'un groupe d'individus (définition de Benjamin Bayart), on peut considérer que l'informatique change la société, le développeur a un rôle majeur dans ce changement : Facebook a changé de façon radicale la façon de draguer, Wikipédia a changé la façon d'étudier, Internet a changé la façon de draguer, de consommer, de s'informer, de débattre, d'étudier, de jouer et j'en passe. L'imprimerie a permis aux gens de lire, Internet leur permet d'écrire (toujours de Benjamin Bayart). Chaque développeur peut révolutionner son domaine : s'il prend le temps, il peut créer un service (site Web ou application), et si son service est utile aux gens (loi du marché...) alors son service révolutionnera son domaine. Ce métier est probablement le plus important du siècle.

Les entreprises n'ont pas besoin de plus de développeurs … elles ont besoin de plus de bons développeurs.

Mathieu Nebra, entrepreneur, co-fondateur d'OpenClassrooms

C'est une confusion qui est trop souvent faite, notamment par les médias, les étudiants et ceux qui se reconvertissent dans l'informatique.

Un développeur et un bon développeur sont deux espèces complètement différentes :

- Le « développeur » va plutôt exécuter ce qu'on lui demande, pondre une fonctionnalité, un site, et passer à une autre demande.
- Le « bon développeur » est plus proche de l'artiste-peintre ou même de l'architecte. Il imagine, conçoit, parmi les milliers de possibilités qui s'offrent à lui, la façon qui sera la plus efficace. Et surtout, il le fait avec une très grande efficacité.

Le bon développeur est rarement polyglotte. Il connaît bien un langage, parfois deux, mais il sait qu'il n'a pas le temps de se spécialiser dans tous les langages qui existent. En revanche, il fait de la veille technologique, affectionne Hacker News et Product Hunt, « sait » que d'autres langages existent et à quoi ils peuvent servir, mais il les creuse rarement. Il est curieux, mais pas dispersé. Structuré, mais pas rigide. Passionné, mais pas aveuglé.

Question salaire, c'est systématiquement le développeur débutant qui se pose le plus de questions à ce niveau. Le « bon développeur » lui, sait déjà dans quelle fourchette il se situe, et bien souvent il n'a pas peur de changer de boîte et de réduire un peu son salaire tant que le projet lui plaît et qu'il sait qu'il va s'éclater avec une super équipe. Il est prêt à prendre le risque, car il sait de toute façon qu'il peut changer d'entreprise très facilement s'il le faut.

En région parisienne, le développeur débutant va se situer autour de 30 000 € à 35 000 € bruts annuels. Dans la fourchette 40 000 € à 50 000 € bruts annuels, vous vous situez probablement parmi les bons profils. Cela peut aller au-delà, notamment pour certains « architectes du code » expérimentés. Et bien entendu, ce n'est pas une règle absolue à prendre au pied de la lettre. On peut être payé moins, on peut être payé plus. Mon conseil est de suivre le développeur expérimenté : visez un beau projet, une belle équipe, plutôt que de tenter de grappiller 500 € annuels supplémentaires. Vous y gagnerez largement au final.

Mais y a-t-il vraiment du travail pour tous ?

Je réagis ici à cet article sur Rue89 : « On m'avait pourtant dit : *"Va dans l'informatique, il y a du travail..." ».* Cela fait plusieurs fois que je constate cet écart :

- D'un côté les entreprises qui répètent : « *On manque de développeurs, s'il vous plaît on en veut plus !* »,
- Et de l'autre, des personnes qui ne trouvent pas de travail, comme celle à l'origine de cet article.

Alors, est-ce un mensonge de dire que l'on manque de développeurs ? Non, plutôt une incompréhension, qui a amené à croire que le travail était facile à trouver du temps qu'on disait « Je suis dans l'informatique ».

L'informatique, c'est vaste. Les compétences à l'intérieur sont très variées. Et un bon développeur vaut 10 à 100 développeurs moyens. En fait, ajouter des développeurs à un projet a plutôt tendance à le faire régresser qu'à l'accélérer.

Ce ne sont pas les développeurs expérimentés qui me contrediront. ;o)

Donc oui, on manque de développeurs : de bons développeurs. Je signe, je re-signe et je confirme. Je vois la difficulté de toutes les boîtes autour de moi à recruter des personnes qui ont la culture technique et la capacité d'apprendre rapidement. Ce sont des oiseaux rares.

Voici quelques pistes pour ce développeur (et ceux qui se reconnaîtront en lui), mais aussi pour les entreprises, car il n'est clairement pas le seul en faute.

Ce que ce développeur devrait faire

- Faire évaluer son niveau par un développeur en poste, pour obtenir des conseils sur ce qui l'empêche de trouver un job.
- Créer ses propres projets, commencer à vendre en freelance, les valoriser sur son CV.
- Aller dans une grande ville. On peut le déplorer, mais en France les jobs sont très concentrés dans les grandes villes (pour ne pas dire Paris). Vous aurez en général un meilleur salaire en région parisienne (20%, 30% de plus...) mais bien entendu le coût de la vie sera plus élevé. En revanche, vous pouvez être certain d'avoir plus de choix. Si vous êtes vraiment allergique à Paris, il existe des bassins technologiques dynamiques qui méritent que l'on s'y attarde : Nantes, Bordeaux, Antibes... Le télétravail est encore hélas assez rare en France et plutôt réservé aux très bons profils. Ne pas hésiter aussi à aller dans un autre pays pour changer de culture, l'expérience est souvent très positive.
- Prendre du recul sur sa façon de rechercher un emploi, en suivant par exemple le cours « Trouvez un job qui vous correspond » de Julie Coudry par exemple. La méthode de recherche et la façon de se vendre importent bien plus que la somme des compétences techniques acquises à l'école.

Ce que les entreprises devraient faire

- Cesser de juger systématiquement sur diplôme (de nos jours, dans ce secteur, ça présage rarement de la qualité du candidat). On a vu de très bons développeurs passer chez OpenClassrooms sans le Bac. Faites évaluer vos recrues par vos bons développeurs en interne... et si vous n'en avez pas, faites-vous conseiller.
- Cesser d'exiger une expertise dans une techno trop précise. Juger plutôt la capacité d'apprendre du candidat et sa façon d'appréhender des problèmes. S'il est bon, il sera capable de se mettre très vite dans votre techno.
- Ne pas sous-évaluer les compétences humaines, le fit culturel. En fait, je vous recommande de recruter des personnes qui vous semblent « bien humainement » et de les former. Vous aurez plus de réussite qu'en essayant le profil inverse.

Bien entendu, ces listes ne sont pas exhaustives. Il s'agit de simples conseils en toute humilité sur ce qui, je pense, pourrait améliorer la situation pour les deux parties. :) ⊞

Le développement passera par moi !

Développeur, le plus beau métier du monde ? Pour ma part ma réponse sera : oui. Tout le monde peut prétendre que son métier est mieux qu'un autre, mais, selon moi, le métier de développeur est unique, et est probablement l'un des meilleurs qui soient.

Kévin Sibué
ksibue@gmail.com
http://www.kevinsibue.com

Je me présente, je m'appelle Kévin Sibué et je suis actuellement en BTS SIO 2e année. Je souhaite par la suite faire une licence professionnelle dans le développement. Et, qui sait, peut-être un Master pour finir. Je me prépare donc à devenir développeur. Depuis que je suis tout petit, j'adore les ordinateurs en général, mais tout particulièrement le développement. C'est un monde unique où tout est possible. Chaque idée que vous avez peut prendre vie, tout ce dont vous avez besoin c'est d'un ordinateur, des connaissances et du temps. Certains se spécialisent dans un domaine bien précis dans le large éventail qu'offre ce domaine. Personnellement j'essaye de rester le plus ouvert possible : Desktop, Mobile, Web et Jeux.

Un vaste monde

Tout m'intéresse. De la conception d'un logiciel de gestion, de la création d'applications mobiles de géolocalisation, en passant par la réalisation d'un site Web de messagerie instantanée. Vous l'aurez facilement compris, le développement est pour moi une passion. Même si je l'avoue la formation que l'on nous propose (tout du moins dans le BTS que je fais) n'est pas nécessairement à la hauteur de nos attentes. Elle est très superficielle et de plus en plus « administrative ». La partie technique n'est plus omniprésente. De nombreuses tâches comme les nombreux modèles (MCD, Gantt, Pert) ou encore des rapports, et même de la documentation en très grande quantité nous sont demandés. Mais le plus dommage dans tout cela est la vétusté des informations que l'on nous transmet. Lorsque l'on nous apprend le développement dans une technologie, par exemple Android, les cours que l'on nous fait, ne parlent pas des dernières versions du système d'exploitation avec toutes ses nouveautés, mais des premières versions (qui aujourd'hui commencent à dater). Mais voyons le bon côté des choses, ça me laisse plus de temps pour apprendre de mon côté tout un tas de nouvelles technos et de faire mes propres applications. Et j'en viens à l'une des caractéristiques les plus fondamentales de ce travail : son évolution. Notre aire de jeu ne cesse de grandir. Nous avions d'abord les ordinateurs, puis le Web, ensuite les smartphones et maintenant les montres. Et demain qu'est-ce que l'on aura de nouveau ? De nouvelles technologies apparaissent chaque jour. Impossible de s'ennuyer ! C'est même parfois un problème. À peine vous avez maîtrisé une technologie que celle-ci change déjà et que votre travail n'est plus à jour. C'est peut-être l'un des seuls points négatifs. Et l'autoformation est très souvent le seul moyen de se mettre à la page. Ce qui est bien dommage. Heureusement, on ne manque pas de documentation ! De nombreux sites nous permettent d'en apprendre encore plus grâce à leurs cours, leurs tutoriels et leurs livres. De nombreux forums sont ouverts sur le Web et nous permettent de poser toutes nos questions tandis que les communautés de développeurs sont toujours là pour y répondre. Et c'est ça qui est génial, c'est que n'importe qui souhaitant apprendre le développement, quel que soit le langage ou la plateforme, peut apprendre gratuitement et devenir un développeur.

Les +/les -

L'avantage et l'inconvénient de ce métier, c'est le grand nombre de personnes qui le font. Vous trouverez toujours du travail, mais attention, de mon point de vue, la concurrence semble rude. Des géants sont toujours dans les parages, et même nos meilleures idées ne peuvent pas toujours passer avec eux, ce qui peut parfois vous empêcher de vous lancer dans des projets. Et de ce fait une pression peut se faire sentir. Autre point positif, où que vous habitiez, du moment que vous possédez une connexion Internet, vous pourrez toujours faire ce métier. Campagne ou Ville, peu importe. Qui plus est le télétravail est souvent pratiqué, ce qui peut vous faire gagner, à vous et à votre entreprise, du temps et de l'argent. De plus il est très simple dans cette branche de devenir autoentrepreneur. Vous aurez simplement besoin de votre ordinateur. Bien sûr la plupart des grands évènements liés au développement se passent dans les grandes villes, mais cela n'empêchera pas un développeur provincial de faire le déplacement pour y assister.
Je pense que pour devenir développeur, il faut aimer le changement et aimer apprendre. La patience peut être un atout non négligeable et bien sûr un esprit logique.

Une passion avant tout

Je suis développeur indépendant depuis plus de dix ans, et coder, c'est à peu près toute ma vie. En effet, au démarrage, mes premiers ordinateurs vous invitaient obligatoirement à écrire des lignes de code en BASIC. Je vous mentirais si je ne criais pas haut et fort que Développeur est pour moi le plus beau métier du monde ! C'est en effet une passion avant tout.

Christophe Peugnet
MVP Windows Plateform Devlopment
chez **SodeaSoft / EBLM**
http://www.sodeasoft.com
Blog : http://www.peug.net
Twitter : @tossnet1

Concevoir un logiciel de A à Z, peaufiner des algorithmes, imaginer l'interface la plus pratique pour votre utilisateur. Je pense qu'il faut être très pragmatique pour être un bon développeur. Ça peut paraître bizarre à dire mais j'aime mon code, et il m'arrive souvent d'y revenir pour l'améliorer, c'est peut-être là l'esprit d'optimisation que l'on a dû avoir avec notre petit programme sur nos calculatrices CASIO fx où chaque caractère comptait.

Le métier de développeur n'est pas pour autant de tout repos, c'est un perpétuel apprentissage, les technologies évoluant constamment, il faut s'accrocher à ce train roulant à pleine vitesse.

Ainsi, depuis 2012 j'ai consacré une grande partie de mon temps à me perfectionner sur le Framework .NET, peut-être un peu tard pour une technologie qui a vu ses premiers pas en 2001. C'est donc par le biais de la création d'applications sur Windows Phone, puis pour le Windows Store que je m'y suis réellement mis.

Le but étant au final de passer mes logiciels professionnels (http://www.sodeasoft.com) que je vends actuellement sous licences perpétuelles en SaaS (Software as a Service), et par la même occasion de tester différents types de monétisations.

Ma formation scolaire n'était pas initialement faite pour le métier de développeur.

Concrètement j'ai tous les diplômes nécessaires pour travailler dans le domaine de l'électrotechnique ; mon « vrai » diplôme qui vaut le plus à mes yeux, je l'ai reçu à 41 ans en avril 2014 par Microsoft avec le titre de MVP (Most Valuable Professional) sur l'expertise Windows Platform Development. Je peux maintenant mourir heureux.

Concernant la rémunération, et je ne focaliserai que sur la partie mobile, mon expérience n'est pas encore assez riche (j'aime le jeu de mot). A moins de trouver l'idée du siècle, l'application must have sur mobile avec soit une bannière publicitaire, en freemium ou via de l'achat intégré, être à son compte et ne vivre que d'applications mobiles n'est pas mon eldorado. Par contre créer des applications pour des tiers semble une bonne voie. J'ai discuté pour vous avec un ami, Franck N'Guessan, développeur qui a tenté cette expérience :

Bonjour Franck, tu peux te présenter ?

J'ai 32 ans et suis employé chez PSA Peugeot Citroën depuis 12 ans. Je suis développeur Windows depuis 3 ans. J'ai commencé la programmation initialement pour combler une lacune vis-à-vis de mes collègues qui occupaient la même fonction que moi. Puis je me suis décidé à réaliser du développement d'application mobile afin de « programmer dans un but ». De fil en aiguille j'ai assisté à des formations Nokia. J'ai choisi Windows Phone car l'OS était jeune avec un store peu fourni en applications. J'ai donc pris le parti pris de parier sur cet écosystème.

Connaissant tes applications, c'est d'abord des besoins personnels ?

En effet, mes premières applications ont été imaginées grâce à ma petite famille. Par exemple « Jeux pour Enfants » a été développé pour accompagner l'apprentissage des chiffres, couleurs, lettres et formes à mon fils ; « Calculer son salaire » a été développé comme outil pour ma femme afin de parfaire ses recrutements. De fil en aiguille j'ai continué à développer de nombreuses applications pour en atteindre plus d'une trentaine avec pour origine de création pour la majorité, un constat, ou un fait de tous les jours. Certaines de mes applications ont été primées à des concours. Mon travail et mon implication pour l'écosystème a été reconnu par Microsoft qui a proposé de présenter ma réussite en tant que « developer hero ».

Ce qui me fait plaisir c'est que je remarque que tout ce que tu apprends, tu le partages.

Oui, tout à fait, j'attache une grande importance à ça, et ainsi j'écris aussi des articles afin de partager des astuces, des difficultés rencontrées lors de la réalisation d'application. Ces articles sont présents sur le site http://biperra.cloudapp.net/

Je voulais que tu nous parles donc de la démarche que tu as faite pour développer une application pour une société tierce qui n'est autre qu'OverBlog. Alors comment ça s'est passé ?

Fervent défenseur de Windows Phone j'ai essayé de convertir ma femme à cet écosystème. Elle a refusé catégoriquement avec pour motif principal : il n'y a pas OverBlog sous Windows Phone. En effet, mon épouse possède un OverBlog dans lequel elle décrit le combat de mon fils atteint du cancer. N'aimant pas les échecs j'ai décidé par tous les moyens de communiquer avec Overblog. J'y suis parvenu après 8 mois de recherche via un tweet. Xavier

overblog

Hausherr de chez Overblog, avec l'appui d'Oliver Lovisa, à l'époque employée de Microsoft, a accepté de me confier la réalisation de la version Windows Phone d'Overblog. Quatre mois plus tard et après quelques phases de tests, l'application est enfin mature afin de faire ses débuts sur le store. Cette application a été une magnifique aventure humaine avec des interlocuteurs disponibles et précis dans leurs réponses. J'espère sincèrement que notre collaboration ne s'arrêtera pas là et que l'avènement de Windows 10 permettra un travail commun sur la réalisation d'une version tablette d'Overblog.

Je remercie vivement Franck N'Guessan pour le temps qu'il m'a accordé pour partager son expérience. On peut te retrouver sur twitter à @CeriBooWP

"Développeur(se), le plus beau métier du monde"

Ludwine Probst,
développeuse chez Cityzen Data, à Brest,
https://twitter.com/nivdul

Stéphanie Moallic,
développeuse chez B&B Hotels, à Brest,
https://twitter.com/steffy_29

Qu'est-ce que c'est qu'être développeur en 2015 ?

Le monde numérique évolue à vitesse grand V, le développeur doit lui aussi être en évolution constante afin de s'adapter et d'apprendre les nouvelles technologies ainsi que les nouvelles tendances.

C'est ce qui fait aussi que le métier de développeur est si varié.

Aujourd'hui, le monde numérique s'invite partout ! Nous sommes entourés de plus en plus de données et d'objets connectés : services, santé, sport, vie de tous les jours...

Quels profils émergent ?

Avec l'explosion du nombre de données et les enjeux qu'elles représentent (stockage, analyse, restitution), de nouveaux profils dans la "Data Science" voient le jour et sont de plus en plus demandés. Les données aujourd'hui représentent de vrais enjeux business, par exemple pour des sites de e-commerce avec des systèmes de recommandation, ou encore personnalisation de contenu par utilisateur.

Après avoir longtemps été négligés, les utilisateurs redeviennent aujourd'hui un centre d'attention avec un intérêt grandissant pour l'UX et l'UI design. En effet les applications proposées étaient souvent peu pratiques, ce qui était lié au peu de concurrence, mais aussi au peu d'intérêt porté à cet aspect. Une attention particulière est portée aux utilisateurs et surtout à leur expérience lors de l'utilisation d'applications ou sites, afin que celles-ci soient les plus naturelles et efficaces possible.

L'utilisateur revient au centre, il doit être séduit et fidélisé.

Et le nombre de développeuses est-il en augmentation ?

D'après plusieurs études et les derniers chiffres, la réponse est non. Il y a aujourd'hui autour de 10-15% de femmes présentes dans les filières informatiques en France.

Ce chiffre ces dernières années semble stagner, voire diminuer suivant les différentes études et sources.

Et l'on dénombre autour de 10% de femmes développeuses ou techniques (les chiffres exacts sont difficiles à obtenir).

Les initiatives à travers le monde se multiplient pour initier les filles et femmes au code, et former des communautés de développeuses pour éviter l'isolement.

En France, on peut citer Devoxx4Kids ou Coding Goûter qui proposent des ateliers de découverte au code pour les enfants et adolescents. L'association Duchess France met en avant des femmes développeuses en tant que rôles modèles, avec pour ambition d'inspirer et représenter des femmes techniques dans l'informatique.

La Web@cadémie, quant à elle, propose une formation de développeur Web en 2 ans pour des jeunes filles et garçons de 18 à 25 ans, sortis du système scolaire sans qualification.

La féminisation des promotions est une préoccupation et une ambition de cette école. Il ne s'agit pas d'atteindre nécessairement une parité parfaite dans l'informatique mais de permettre aux jeunes filles, mais aussi aux femmes, de mieux se projeter dans ce milieu, de s'y affirmer; le milieu de l'informatique souffrant encore aujourd'hui de clichés, et manque de modèles féminins.

Mon âme de développeur

Même si au fil des années, j'ai changé plusieurs fois de spécialités dans l'informatique, je n'ai jamais quitté le monde du développement, pour différentes raisons :

- Tout d'abord pour la liberté de créer suivant mon envie, même pour le FUN
- Ensuite de m'intéresser aux nouveautés pour ouvrir mes connaissances
- Et enfin de partager cette connaissance acquise

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

Mon arrivée dans l'univers de développement a commencé bien avant la démocratisation de l'internet car les seuls moyens d'apprendre étaient les livres, les magazines, et les clubs informatiques. Or pour mettre en pratique ce que je lisais, je retapais sur mon ordinateur, les lignes de codes publiés en Assembleur 68000 des différents magazines que j'achetais et je passais beaucoup de temps à corriger mes erreurs de frappes. Pour comprendre le langage, je me suis documenté à partir des livres et je me suis aussi fait assister par des connaissances des clubs. Bien sur, j'ai rapidement vu les possibilités qu'offrait le développement et cela m'a donné des idées et l'envie de faire la même chose. Toutefois, 'avoir envie' est une chose, et le faire, en est une autre.

La première étape était de trouver une idée personnelle et de la réaliser complétement même si celle-ci existe déjà (Internet, logiciels, jeux...). Le but c'est d'avoir une envie de développer et votre satisfaction sera quand vous l'aurez terminée et qu'elle fonctionne. Ensuite, il sera toujours temps de partager votre réalisation. La deuxième étape est de se documenter pour pouvoir réaliser l'étape précédente. Le Web ne fera pas tout car souvent ce sont des portions de code que vous retrouverez. C'est pourquoi la source de mes connaissances restera toujours les livres, les communautés, les fablabs, les meetups (et Programmez, NDLR)...

Enfin, il n'est jamais évidant de terminer un développement seul. Il ne faut pas hésiter à en parler autour de soi et à le montrer, car souvent quand vous en parlez, la solution aux problèmes vient naturellement.

Ma vision du futur du développeur montre que le développement fonctionne par cycle et qu'il suit l'évolution technologique du moment.

C'est pourquoi, les nouveaux moyens de consommation du numérique auront besoin d'applications et par conséquent de développeurs, et les outils/logiciels que nous utilisons aujourd'hui, il faudra les refaire pour les faire tourner sur ces nouveaux supports.

Un développeur a une âme de faire toujours quelque chose, même inutile, alors lorsque la retraite sonnera, le développement continuera à m'appeler pour continuer à avancer et à transmettre à la jeune génération...

Etude mutationelles http://www.global-contact.net/etudes/
Duchess France www.duchess-france.org
Web@cadémie http://webacademie.org/

Le mouvement "apprendre le code" à l'école peut-il aider à sensibiliser les filles à l'informatique ?

Le mouvement "apprendre le code" est une initiative intéressante mais qui semble limitée si elle se traduit par le simple apprentissage d'un langage ou outil. Ce qui nous semble le plus important dans le code et l'informatique en général, c'est le raisonnement, la capacité à poser et résoudre un problème et la compréhension de l'algorithmique. Les mathématiques sont justement enseignées à la base dans ce sens au collège et lycée : ils nous initient aux démonstrations de théorèmes, à la modélisation et résolution de problèmes.
Le monde actuel étant de plus en plus ancré dans le numérique, il semble désormais important d'y sensibiliser les enfants à l'école; ils comprendront au mieux le monde qui les entoure et cela leur donnera les armes et connaissances nécessaires pour qu'ils saisissent notre monde actuel et puissent s'y impliquer au mieux.

Pourquoi devenir développeuse ?

Stéphanie : et pourquoi pas ? J'ai toujours trouvé fascinant ce que l'on pouvait faire avec un ordinateur et quelques lignes de code : de la musique, de la génération de fractales, des sites Internet, commander la mise en route d'un chauffage dans une pièce...Les domaines d'applications du métier de développeur/développeuse sont vastes. Et j'apprécie le côté créatif du métier.
Ludwine : le métier de développeur évolue au rythme du monde qui nous entoure. L'informatique et plus globalement le numérique, de plus en plus présents dans notre quotidien, modèlent et impactent clairement notre façon de vivre. Aujourd'hui pour moi, un développeur n'est plus un simple exécutant, il devient créateur et innovateur. Et les hommes autant que les femmes peuvent s'y épanouir et y trouver leur place.

Notre dernier mot : où bosser, faut-il un diplôme, bon ou mauvais job ?

Que retenir de ce dossier ? La passion est un leitmotiv qui est souvent revenu parmi les témoignages. Et avec raison. La passion est un des moteurs du développeur. J'ai connu des développeurs qui ont rapidement voulu migrer vers le rôle de chef de projet, même s'ils savaient qu'ils n'étaient pas forcément très bons en gestion de ressources humaines. Mais leur rapport avec le code était un peu distant, il fallait faire le job mais la passion n'était pas très présente. D'autres ont purement et simplement quitté le métier pour faire autre chose et changer de vie.

François Tonic

Vous l'aurez compris, développeur, comme tout job, est exigeant, parfois épuisant, souvent stressant. Mais on peut s'y épanouir quand on aime ce que l'on fait. Mais le développement n'est pas l'eldorado où l'on va décrocher un bon poste et un bon salaire sans rien faire. Il faut aussi savoir se remettre en cause quand son profil devient obsolète ou trop standard pour faire la différence. La concurrence est féroce car des développeurs il y en a. Un bon développeur trouvera toujours du travail. Des compétences spécifiques feront aussi la différence, par exemple : Drupal, Symfony, Zend, Hadoop, etc. Mais attention, on ne devient pas expert dans une technologie X après 1 mois d'utilisation. C'est le risque du syndrome du développeur Web au tournant des années 2000 où on trouvait tout et n'importe quoi.

Paris, Province, Étranger ?

L'opposition Ile de France – province n'est pas nouvelle. Nous avons interrogé le Cinov-IT à ce sujet (M. A. Rallong) : il y a plus de sièges sociaux et un marché de l'emploi (plus important) en Île de France, mais certaines régions sont numériquement très dynamiques. Citons par exemple Bordeaux, Nantes, Grenoble, Lyon, Toulouse, ou encore la région Auvergne. Après, dans des zones très rurales, cela peut être plus difficile pour trouver des formations et des postes. Toujours selon le Cinov-IT, parmi les arguments pour s'installer en province, ou y rester, nous trouvons la qualité de vie, un coût de la vie moins élevé (excepté dans certaines villes). Sur les salaires, l'écart existe toujours entre Paris – Province mais attention, il y a aussi des différences entre les régions.
Il faudra parfois accepter une certaine mobilité dans une même région ou vers une région.
Partir à l'étranger ? Si vous avez une opportunité, saisissez-la ! Vous allez découvrir un autre univers, une autre culture, découvrir et acquérir de nouvelles compétences. Osez ! Vous êtes nul en Anglais ? Ne vous bloquez pas. L'important est de communiquer et de se faire comprendre. Et vous allez vite maîtriser. Bien entendu, un départ à l'étranger pour plusieurs mois ou plusieurs années ne s'improvise pas (fiscalité, contrat de travail, visa, logements, etc.).

Apprentissage du code à l'école : oui, mais...

L'informatique est partout. C'est un constat et une réalité. Donc il faut des applications, et donc des développeurs. La formation a bien entendu son importance. Vous avez les cursus traditionnels, les filières courtes et les écoles de type 42. Pour Nicolas Sadirac, 42 est en dehors des silos. Et l'école permet aussi de casser une certaine image du développeur et de l'informatique. Souvent, le jeune a une vision très technique de la chose, dixit N. Sadirac. Mais dès la piscine, dès les premières semaines à l'école, les étudiants voient que ce n'est pas seulement un métier technique. Dans le même temps, le directeur de 42 précise « il faut s'adapter tout le temps, les modes changent. Un bon développeur est capable de s'adapter. ».

> *« Le code n'est pas une finalité »*
> Nicolas Sadirac, 42

Diplôme ou pas diplôme ? C'est une question qui revient souvent et que je me pose de plus en plus. Si nous prenons l'exemple de 42, des dizaines de milliers de jeunes, avec le bac, parfois un peu plus, parfois sans aucun diplôme, tentent leur chance aux sélections. Mais la sélection est sévère. Au final, ce sont un millier d'élèves qui vont rentrer. Ceux qui réussissent la Piscine (4 semaines intensives, 15h par jour). « Ils savent à quoi s'attendre » précise N. Sadirac. L'étudiant doit donc apprendre rapidement et suivre la cadence. Au final, 42 affiche un très faible taux d'abandon, environ 7 %... Un dogme que 42 veut insuffler : le code c'est fun !
Mais les cursus traditionnels, les diplômes, les cursus universitaires gardent leur utilité. Ils fournissent des socles techniques et méthodologiques nécessaires. Il faut aussi pouvoir innover. Ensuite, tous les étudiants n'ont pas les moyens financiers de payer des milliers d'euros chaque année. Et encore, il faut rajouter toutes les dépenses annexes : logements, transports, matériels, nourriture, sorties, etc. La vision trop technique, trop mathématique peut aussi décourager les lycéens, et les obliger de changer de filière alors qu'ils pourraient faire de bons développeurs. Il faut encourager ces profils.

⊞

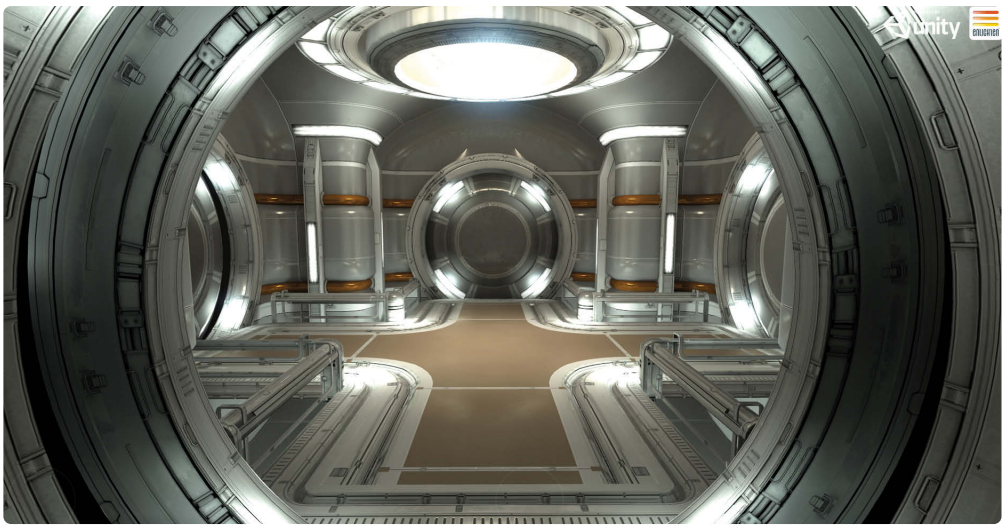
To be continued...

Nous allons poursuivre sur le métier du développeur avec :

- Développeurs & jeux vidéo : les filières, les métiers, les salaires
- Quelle école d'informatique choisir ?

A suivre dans Programmez ! n°183 de fin février

Bienvenue dans Unity



Unity est une plateforme / un écosystème complet vous permettant de créer des applications, dans la majorité des cas des jeux-vidéos, pour une multitude de plateformes (17 à ce jour). Grâce à Unity, vous pouvez développer des jeux aussi bien en 2D qu'en 3D à l'aide d'un éditeur à la fois simple et puissant qui fait la grande force du produit. Car si l'on peut bien évidemment faire ses propres scripts, beaucoup d'actions se font sans écrire la moindre ligne de code, mais simplement via l'interface de l'éditeur.

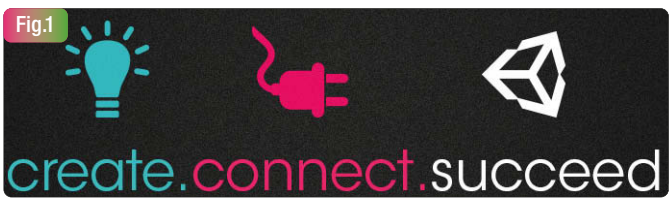
Maxime **Frappat** & Jonathan **Antoine**
développeurs passionnés - Infinite Square

Unity c'est aussi plusieurs services dans le Cloud: ils vous offrent notamment de compiler vos sources sur les serveurs d'Unity (sans avoir besoin d'un Mac par exemple), de mettre à disposition vos applications pour vos testeurs, d'obtenir et consulter des statistiques d'usage de vos jeux. Il existe aussi un store complet de plugins et d'assets (vidéos, images, shaders, textures, modèles 3D) réutilisables simplement dans vos développements Fig.1.

Plateformes supportées

Avec plus d'une quinzaine de plateformes supportées, Unity est de plus en plus utilisé par les développeurs afin de cibler une multitude de plateformes (mobiles ou non), sans avoir à faire une application par système cible.

Fig.1



Parmi ces plateformes, on retrouve : iOS, Android, Windows Phone et BlackBerry pour la partie mobile, Xbox 360, Xbox One, PS3, PS4, PS Vita, Playstation Mobile, Wii U. Unity est aussi pour les consoles et le support des différents système d'exploitation de bureau comme Windows, Windows Store Apps, Mac, Linux ainsi que la plupart des navigateurs Internet Fig.2.

Le principe est le suivant : vous développez votre jeu une fois avec l'éditeur Unity et Visual Studio (ou Mono Editor par exemple), et Unity est en mesure de vous générer les binaires pour les différentes plateformes qu'il supporte. Cela parait un peu magique. Mais cela fonctionne vraiment !

L'éditeur

Téléchargement d'Unity

Vous l'aurez compris, l'éditeur est la pièce maîtresse d'Unity et vous pouvez l'obtenir gratuitement à cette adresse :
http://unity3d.com/unity/download.

L'éditeur fonctionne aussi bien sur Windows que sur OS X et Linux. D'expérience, la génération des packages fonctionne très bien sous Windows mais peut poser quelques problèmes lors de la génération de package iOS (la génération se passe bien mais le code ne compile pas). Il est donc recommandé d'installer Unity sur Windows pour vos développements (avec Visual Studio) et d'avoir une version d'Unity sur votre Mac pour générer les packages iOS.

À noter qu'Unity produit aussi beaucoup de patches de corrections de bugs et qu'ils sont disponibles à cette adresse :
http://unity3d.com/unity/qa/patch-releases.

Licences

Unity est disponible sous la forme de deux versions « pro ou gratuite » et une page complète détaille des différences entre ces deux possibilités :

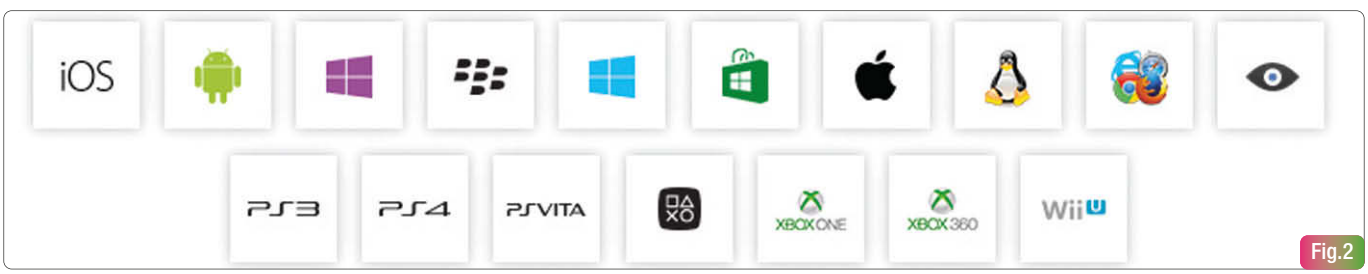


Fig.2

http://unity3d.com/unity/licenses. Voici cependant les limitations les plus visibles de la version gratuite :

- Pas de splashscreen personnalisé,
- Pas de lecture de vidéos,
- Pas d'effets avancés sur l'affichage de vos scènes,
- Pas de support des plugins natifs (C++, C, Java, Objective-C),
- Profiler et Profiler GPU pour investiguer vos problématiques de performances,
- Pas de personnalisation de l'interface de l'éditeur avec le skin « noir ».

Pour plus d'informations sur les différences entre les deux types de licences rendez-vous sur http://unity3d.com/unity/licenses.

Versions

Unity possède un système de versionning faisant que chaque nouvelle version apporte une réelle nouveauté. La version 4.5 apportait par exemple beaucoup de nouveautés autour du système de shaders et plus de 450 corrections de bugs. La version 4.6 qui vient de sortir il y a seulement quelques semaines était quant à elle très attendue car elle apporte un système de création d'interface utilisateurs (boutons, textes, label, panels, etc.) directement intégré à Unity.

Cela peut paraître un peu déroutant d'avoir autant de versions car pour chacune, Unity propose une série de versions correctives (4.5.1, 4.5.2, 4.5.3, 4.5.4, 4.5.5); il ne faut toutefois pas en avoir peur car il y a très peu de breaking changes (je n'en ai jamais rencontré) entre les différentes versions. La prochaine version d'Unity sera la version 5, et ce sera une grosse mise à jour avec beaucoup d'améliorations pour augmenter la productivité de développement et faire d'Unity un éditeur encore plus professionnel. Il serait nécessaire d'écrire un article complet sur le sujet mais voici déjà les 6 fonctionnalités principales de cette version :

- **Un nouveau système de shader** (ces petits bouts de code exécutés sur le GPU) optimisés pour le rendu de matériaux « physiques ».
- **Un nouveau système d'éclairage** permettant d'être au plus proche de la réalité, même sur les plateformes mobiles avec notamment la possibilité d'effectuer des animations sur les sources d'éclairage.
- **Un nouveau système audio.** Le précédent était déjà très puissant avec une intégration avec FMOD, mais le nouveau système permet de faire du mixage dans l'éditeur, de combiner des effets, des hiérarchies d'effet sonores et bien plus !
- **Le support de WebGL et la possibilité de déployer vos jeux dans un navigateur sans plugin.** Plus besoin d'installer un activeX sur les navigateurs compatibles WebGL pour proposer vos jeux. Le navigateur devient une plateforme supplémentaire comme les autres.
- **Le support du 64 bits à la fois pour l'éditeur** (c'est-à-dire de meilleures performances sur les gros projets) mais **aussi pour les binaires iOS,** obligatoire à partir du 1er février 2015**.**
- **L'intégration de leurs propres plateformes de publicité inter-applications.** Vous pourrez promouvoir vos propres applications ou échanger de la visibilité avec d'autres applications utilisant ce réseau.

On peut voir qu'Unity fait tout pour séduire les développeurs de jeux vidéo et n'est pas en manque d'évolutions et de propositions de nouvelles fonctionnalités pour leurs outils !

Le lien de téléchargement est toujours le même :
http://unity3d.com/unity/download

Installation

Voici les étapes qui vous seront nécessaires pour installer l'éditeur :

- Lancez l'exécutable téléchargé précédemment,
- Sur le premier écran d'introduction, cliquez sur « Next »,
- Le deuxième écran vous affiche les termes d'utilisation du logiciel, cliquez sur « I Agree » pour les accepter et passer à l'étape suivante,
- L'écran suivant vous propose un choix d'installations facultatives à installer. Par défaut, l'ensemble des éléments sont cochés ce qui vous permettra d'avoir un environnement de développement (*MonoDevelop*) et un projet de démonstration (*AngryBots*) afin de tester ce qu'il est possible de faire. Cliquez sur « Next » pour terminer l'opération,
- L'étape suivante vous propose de modifier l'emplacement d'installation du logiciel. Cliquez sur « Install » pour lancer l'installation,
- Une fois l'installation terminée, cliquez sur « Finish » pour lancer l'éditeur.

Il est intéressant de noter qu'il est possible d'installer et de faire cohabiter plusieurs versions différentes d'Unity dans des répertoires différents, sans difficulté. Vous pouvez donc sans problème travailler sur un projet en 4.5 d'un côté, et en avoir un autre en 4.6 de l'autre : très pratique !

Premier lancement

Lors du premier lancement, une boîte de dialogue vous demandera de créer un nouveau projet. Choisissez l'emplacement où vous souhaitez le sauvegarder, ainsi que la liste des packages à inclure automatiquement à la création. Sachez que vous pourrez les inclure par la suite si vous ne le faites pas via cette boite de dialogue.

Le dernier paramètre que vous pouvez modifier est le type de jeu que vous souhaitez faire, 2D ou 3D. En sélectionnant l'un ou l'autre, l'éditeur changera sa mise en forme pour s'adapter à une vue en « à plat » ou en 3 dimensions. De même que les packages, ce paramètre peut être modifié une fois le projet créé Fig.3.

Terminez le processus de création en cliquant sur « Create ». Dans le cas d'un projet 3D, vous devriez arriver sur un écran de ce type : Fig.4.

Voici quelques précisions sur les différents modules affichés :

- **Project** : la hiérarchie des fichiers de votre projet. Vous pouvez en ajouter via un glisser/déposer directement depuis l'Explorer vers Unity.,

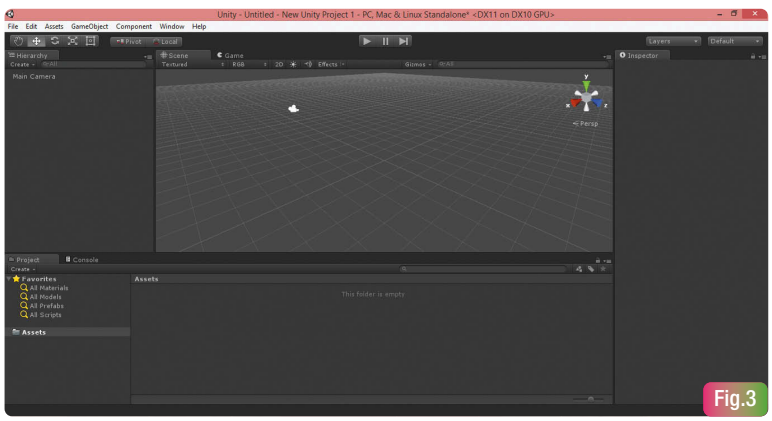
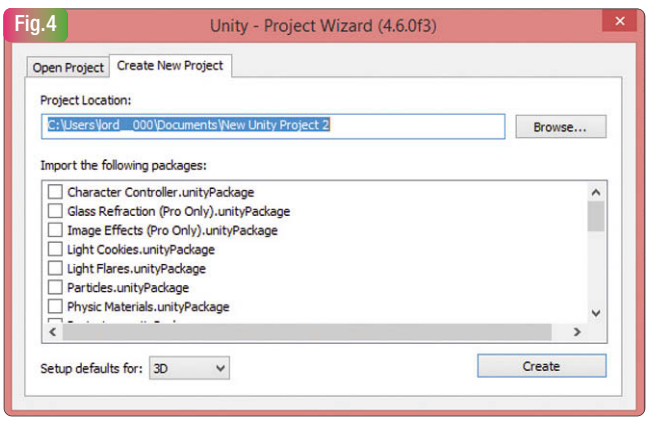


Fig.3



- **Scene** : la scène courante où s'afficheront vos objets,
- **Hierarchy** : la hiérarchie des objets que contient la scène,
- **Inspector** : l'ensemble des composants et propriétés de l'objet sélectionné de la scène courante,
- **Console** : affiche les messages de compilation, d'erreurs, d'avertissements ou de débogage,
- **Game** : le rendu de votre scène lorsque vous testez votre jeu dans l'éditeur. Cela permet d'obtenir le même rendu que sur les plateformes finales.

À noter que les différentes fenêtres sont déplaçables à volonté selon vos envies. Si par erreur vous en fermez-une, pas de crainte à avoir, elle sera toujours ré-activables en passant par le menu « Windows ». Les habitués de Visual Studio ne seront donc pas trop perdus dans cet environnement.
Sous le menu, on retrouve plusieurs boîtes à outils :

- A gauche : un ensemble de boutons permettant de zoomer ou se déplacer dans la scène ainsi que de bouger ou faire pivoter un objet,
- Au centre : très certainement les boutons les plus utilisés pendant vos développements car ils permettent de lancer le jeu pour le tester immédiatement dans l'éditeur, de le mettre en pause ou d'avancer frame par frame lors des phases de débogage.
- A droite : un bouton pour paramétrer l'affichage des couches (*Layers*) et un autre pour paramétrer la disposition des modules dans l'éditeur.

Utilisation des scènes

La scène est la pièce principale de votre jeu. Tous les objets qui s'afficheront seront inclus dans une scène quoiqu'il arrive. Cela peut s'apparenter à un gros conteneur, aux balises <HTML></HTML> ou au papier sur lequel on va écrire : c'est notre support. Lors de l'assemblage de votre projet, le rendu de votre jeu sera une collection d'une ou plusieurs scènes, des objets que vous y avez placés et de l'ensemble des scripts que vous avez créés.
Typiquement, dans une scène on retrouve souvent une caméra, une lumière ainsi que différents éléments de décors. Tous ces objets sont placés sur la scène à certaines coordonnées, avec des paramètres spécifiques et c'est la scène qui se charge de garder ses informations. Pensez donc à sauvegarder la scène aussi souvent que possible !
Attention, la scène est donc plus représentée par le volet « hierarchy » où l'on retrouve les objets la composant que par le volet « scène », où il ne s'agit que de sa représentation.
Par défaut, au premier lancement, Unity crée une scène, mais vous pouvez en créer une nouvelle quand vous le souhaitez en allant dans le menu « File à Create a new scene ». Par convention, les scènes sont sauvegardées et rangées dans un dossier « Scenes » du dossier Assets où se trouvent tous les fichiers/dossiers.

Structure du projet

Un projet Unity s'articule principalement autour de quatre dossiers : Assets, Library, ProjectSettings et Temp.
Le dossier « Assets » contient l'intégralité des scripts, sons, images, ... de votre jeu. Si l'on regarde de plus près le module « Project », on peut voir que seul le dossier « Assets » y figure. Il est important de noter qu'il est préférable, voire obligatoire, d'ajouter un fichier via l'interface d'Unity plutôt que directement dans le dossier, car, lors de l'ajout, l'éditeur va créer un fichier de métadonnées qui sera lié à celui-ci. Dans le cas contraire, il est possible qu'Unity l'ignore tout simplement. Pour ajouter un fichier, il est possible de faire un clic droit puis « Import à New Asset », ou encore, plus simplement, de le glisser/déposer dans le bon dossier.
Le dossier « Library » sert de stockage local pour les éléments importés en maintenant leurs métadonnées. Vous pouvez sans crainte le supprimer à n'importe quel moment pour qu'Unity le régénère pour vous par exemple en cas de problème.
Le répertoire « ProjectSettings » va stocker les différentes configurations liées au projet comme les paramètres audio, la qualité des textures selon les plateformes, ... Tout ceci est accessible via le menu « Edit à Project Settings ».
Pour finir, le dossier « Temp » est utilisé par les fichiers temporaires lors des phases de compilation. Vous pouvez lui aussi le supprimer sans crainte à n'importe quel moment.

Utilisation d'un contrôleur de code source

Unity supporte très bien l'utilisation d'un contrôleur de code source tel que TFS, Git ou SVN mais il est nécessaire de configurer correctement l'éditeur.

Configuration d'Unity

Il est notamment important de rendre visibles les fichiers de métadonnées et de les sérialiser sous la forme textuelle plutôt que binaire afin de faciliter la gestion des conflits. Pour cela, il faut aller dans le menu « Edit à Project Settings à Editor » et activer l'option « Visible meta files » ainsi que « Force Text » comme mode de sérialisation Fig.5.
Cette configuration aura pour effet de créer un fichier d'extension « .meta » pour chacun de vos fichiers. Il faudra bien prendre soin de les synchroniser avec votre contrôleur de code source.

Quels dossiers et fichiers synchroniser ?

Si vous respectez la hiérarchie de dossier standard d'un projet Unity, seuls 3 dossiers sont à synchroniser : « Assets », « ProjectSettings » et « BuildSettings ». Chaque nouveau développeur pourra ouvrir le projet avec ces 3 seuls dossiers. La première ouverture sera bien sûr un peu plus longue car Unity importera les différents assets et créera le dossier « Library » de métadonnées.

La classe GameObject

Généralités

Dans Unity, chaque élément d'une scène hérite de la classe « GameObject », ce qui en fait l'équivalent de la classe « System.Object » dans le Framework .NET. Que ce soit une lumière, une caméra, un système de particule, ... c'est un GameObject à la base.
Concrètement, si vous créez un nouveau GameObject dans votre scène, ce qu'il est possible de faire via le module « Hierarchy » et son menu « Create à Create Empty », un nouvel objet vide sera automatiquement ajouté à la scène avec, dans le panneau « Inspector », la possibilité de modifier son nom, sa position, sa rotation et sa taille. Ces quatre paramètres sont les composantes de base de la classe GameObject.

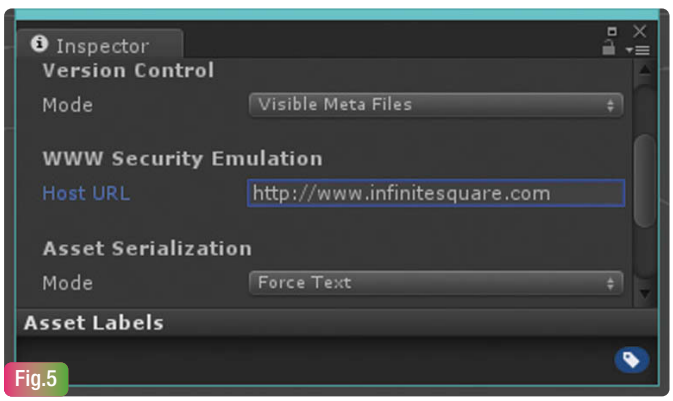


Fig.5

Dans le monde du développement de jeu-vidéo, il est courant d'utiliser des vecteurs pour définir une position. De ce fait, chaque propriété de la classe « Transform » est un vecteur comprenant trois coordonnées X, Y et Z (*Vector3*).
Il est aussi important de noter que chaque gameobject peut être activé ou désactivé, soit via l'éditeur (la case à cocher à côté du nom de l'objet sert à cela), soit via le code (via la méthode gameObject.SetActive d'un script). Lorsqu'un script est désactivé, il se comporte comme s'il n'existait pas du tout !

Composants

Comme vous avez pu le voir, un GameObject ne comprend aucune fonctionnalité autre que son positionnement et quelques métadonnées (nom, layer, etc.). Pour lui rajouter des attributs et le rendre utile, il est possible de lui ajouter ce qu'on appelle des composants (*Components*) qui apparaîtront dans la fenêtre « Inspector ».
Ils peuvent être de différents types comme un « SpriteRenderer » pour afficher une image, un « Mesh » qui est la représentation d'une forme grâce à un réseau de mailles (combinaison de triangles), un son avec une « AudioSource », de la physique grâce à un « Collider » pour gérer les collisions, un système de particule, etc.
Pour voir comment tout cela fonctionne, nous allons ajouter un cube dans notre scène via le menu «Menu GameObject à 3D Object à Cube ». Si l'on observe ce que nous retourne l'inspecteur nous pouvons voir 4 objets:

- **Transform** : hérité de la classe GameObject,
- **Cube (Mesh Filter)** : définit un « Mesh » de forme cubique. Il est présent et disponible par défaut dans Unity,
- **Box Collider** : crée une surface de collision de forme cubique,
- **Mesh Renderer** : définit la façon d'afficher le « Mesh » en appliquant une texture de matériaux.

Unity nous a ajouté automatiquement un gameobject avec ces 4 composants afin d'afficher un cube.
Voilà, nous venons de créer et d'afficher notre premier cube sur 17 plateformes.

Créer son propre script/comportement

Maintenant que l'on sait afficher un objet sur la scène, nous pouvons passer à la partie « Scripting » qui va nous permettre de définir un comportement à notre objet grâce à du code. Celui-ci pourra être écrit en plusieurs langages : C#, JavaScript ou encore Boo. Durant l'ensemble de ce dossier, nous utiliserons le C#.

Qu'est-ce que c'est qu'un comportement et un script ?

Un comportement est une pièce maîtresse de votre jeu car c'est lui qui va contenir la logique de celui-ci. C'est grâce à un script que vous définirez que le joueur gagne 100 points s'il attrape une pièce ou qu'il perd s'il tombe dans une trappe. Un script peut aussi définir des comportements lorsque l'utilisateur tape sur un bouton, lorsqu'un événement survient, etc. En fait, un script est capable de faire exactement ce que vous souhaitez et la meilleure façon de le définir est : « un bout de code attaché à un ou plusieurs Gameobject ».
Il est possible d'ajouter autant de scripts/comportements que vous le souhaitez à un objet, et rien ne vous empêche de les réutiliser dans vos différents projets.

Ajouter un script simple sur un objet

Sélectionnez votre cube et cliquez sur « Add Component > New Script ». Nommez le « TestScript » et vérifiez que le langage sélectionné soit « CSharp ». Une fois l'opération terminée, double-cliquez sur le nom du script dans l'inspecteur, cela aura pour effet d'ouvrir MonoDevelop (ou Visual Studio) qui va nous permettre de modifier le script.
Voici à quoi ressemble le code par défaut une fois le script créé :

```
using UnityEngine;
using System.Collections;

public class TestScript : MonoBehaviour
{
    // Use this for initialization
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

La classe *TestScript* hérite de la classe *MonoBehaviour*. Ceci est très important car c'est cette classe qui nous permet d'avoir accès à des fonctions importantes dans l'univers d'Unity (nous en parlons juste après J) : *Awake()*, *Start()*, ... et sans cet héritage il n'est pas possible de lier le script en tant que composant et de le faire apparaître dans l'inspecteur. Rappelez-vous donc que chaque script qui sera utilisé en tant que composant doit hériter de la classe *MonoBehaviour*.

Cycle de vie d'un script

Un script possède son propre cycle de vie sur lequel vous pouvez insérer votre propre code, et il existe plus d'une vingtaine de points d'insertion possibles pour nous autres développeurs.
Voici les moments de vie les plus importants à connaître :

- **Awake :** exécuté une fois avant que tout autre code ne le soit, et notamment avant qu'aucune autre méthode « start » ne soit déclenchée sur aucun autre objet. Attention, ce code n'est pas appelé si le gameobject est desactivé,
- **Start :** exécuté une fois au « lancement » de chaque objet, c'est-à-dire lorsqu'il est inséré, activé dans une scène avant que la méthode «Update » ne soit appelée pour la première fois. C'est un bon endroit pour initialiser vos scripts. Attention : il est fortement déconseillé par Unity d'utiliser le constructeur pour initialiser vos objets,
- **OnCollision :** appelé lorsqu'un collider est associé à votre Gameobject et qu'il rencontre un objet déclencheur. Très pratique donc si vous souhaitez ajouter une logique à ce type d'événement,
- **Update :** appelé à chaque frame d'affichage d'Unity. C'est le bon endroit pour vérifier si l'utilisateur tape avec son doigt ou la souris sur votre joueur, un ennemi ou encore pour déplacer vos différents éléments. La valeur « Time.deltaTime » vous permettra de connaître le temps écoulé depuis le dernier appel de la méthode update par le player d'Unity. Attention, ce code étant appelé à chaque frame, il est important de ne pas faire trop de « choses ici » pour ne pas diminuer les performances de votre jeu,
- **OnDestroy :** appelé lorsque l'objet est détruit soit explicitement via du code ou lorsque la scène est déchargée par Unity.

Pour avoir la liste et l'ordonnancement complet, je vous invite à lire la page suivante du manuel en ligne d'Unity :

http://docs.unity3d.com/Manual/ExecutionOrder.html

Pour ajouter votre propre bout de code il *suffit* d'ajouter une méthode (privée ou non) ayant comme nom l'événement ciblé et ne retournant rien (void en C#). Par exemple, ce bout de code C# déplace le Gameobject sur lequel il est attaché vers l'infini (et au-delà) :

```
// Update is called once per frame
void Update()
{
   //Déplacement d'un pixel sur l'axe Y
  this.gameObject.transform.Translate (0, 1, 0);
}
```

Aussi, par défaut, un script n'est pas exécuté dans l'éditeur, mais il est possible de modifier ce comportement en plaçant l'attribut « [ExecuteInEditMode] » sur la définition de la classe du script.

Ajouter des paramètres à vos scripts

Maintenant que nous savons créer des scripts, il devient vite intéressant de pouvoir passer des paramètres à ceux-ci, et de pouvoir les renseigner directement dans l'inspecteur de propriété d'Unity.

Pour cela rien de plus simple : il suffit de déclarer vos champs comme publics. Tous les champs publics apparaîtront donc automatiquement dans l'éditeur avec le bon éditeur (case à cocher pour un booléen, champ texte avec validation pour un float, etc.).

Attention, une propriété avec getter/setter ne sera pas visible dans l'éditeur !

Il est aussi possible d'ajouter un peu plus d'informations à l'éditeur en utilisant des attributs. En voici quelques exemples :

- [Tooltip("Je suis une valeur entre 0 et 100.")] : affiche un tooltip sur la propriété,
- [Range(0, 100) : définit une zone de valeurs possibles pour le champ privé,
- [Header("Des propriétés relatives à la santé")] : définit une section et son titre dans l'éditeur,
- [HideInInspector] : permet de cacher pour une raison quelconque une variable de l'éditeur.

Mise en place de Singleton/Manager

Un des concepts fondamentaux dans le développement d'un jeu est ce qu'on appelle le « Manager ». C'est une classe qui se veut globale dans notre projet et pour lequel nous ne souhaitons pas avoir plusieurs instances, même lors d'un changement de scène. Un *design pattern* bien connu des développeurs ressort donc logiquement : le Singleton.

Voici donc comment le mettre en place dans Unity :

```
using System.Diagnostics;
using UnityEngine;

namespace Assets
{
   public class Singleton<T> : MonoBehaviour where T : MonoBehaviour
   {
      private static T _instance;

      public static T Instance
      {
         get
         {
            if (_instance == null)
            {
               _instance = FindObjectOfType<T>();

               if (_instance == null)
               {
                var gameObject = new GameObject("Singleton <" + typeof(T).Name + ">");
                  _instance = gameObject.AddComponent<T>();
               }

               //Tell unity not to destroy this object when loading a new scene!
               DontDestroyOnLoad(_instance.gameObject);
            }

            return _instance;
         }
      }

      void Awake()
      {
         if (_instance == null)
         {
            //If I am the first instance, make me the Singleton
            _instance = this as T;
            DontDestroyOnLoad(this);
         }
         else
         {
            //If a Singleton already exists and you find
            //another reference in scene, destroy it!
            if (this != _instance)
               Destroy(gameObject);
         }
      }
   }
}
```

Ce code un peu compliqué au premier abord (mais fourni dans la documentation d'Unity) va exposer une propriété statique « Instance » de type « T » correspondant à l'objet que vous souhaitez définir en tant que Manager. Si cette instance existe, elle est retournée sinon, elle est créée et ajoutée à la scène courante. Cela est très astucieux car cela limite la durée de vie de celle-ci à celle de la scène.

De plus, vous constaterez que chaque singleton est visible dans la hiérarchie des objets lorsque vous jouez le jeu dans l'éditeur : très pratique pour savoir ce qui se passe.

À l'utilisation, cela est un jeu d'enfant. Voici par exemple un extrait du code du jeu HiLight définissant un manager « InAppPurchaseManager » responsable des achats in-app :

```
InAppPurchaseManager.Instance.RequestBuy();

//Définition de la classe InAppPurchaseManager
public class InAppPurchaseManager : Singleton<InAppPurchaseManager>
{
   public void RequestBuy()
   {
    //Ne pas oublier de dire merci !
   }
}
```

Exemple concret

C'est parti pour la création d'un mini-jeu qui nous permettra de montrer quelques concepts simples. Le jeu sera composé de deux scènes, une d'introduction et une de jeu. Le plateau de jeu sera composé d'une sphère rouge en son centre, en cliquant sur celle-ci on fera apparaître deux nouvelles sphères, une rouge et une noire, ayant des positions et des dimensions aléatoires. En cliquant sur une boule noire, la partie est perdue et on revient à la scène d'introduction. En cliquant sur une rouge, on recrée deux boules de couleurs différentes.

Interface utilisateur

Créez un nouveau projet 3D et ajoutez du texte via le menu « *GameObject > UI > Text* » et un bouton via « *GameObject > UI > Button* ». Vous pouvez remarquer qu'un *Canvas* a été créé automatiquement afin de contenir mes deux éléments. Amusez-vous à positionner et personnaliser le texte et le bouton comme bon vous semble (couleur du texte, alignement, ...). Sauvegardez la scène sous le nom *MainScene* Fig.6.

Création d'un manager

La prochaine étape est de créer un manager pour gérer les différents états du jeu. Voici à quoi ressemble cette classe :

```
using Assets;
using UnityEngine;

public class GameManager : Singleton<GameManager>
{
  public enum GameState
  {
    // MainScene
    Wait,

    // GameScene
    Started
  }

  public GameState CurrentState;

  private void Awake()
  {
    CurrentState = GameState.Wait;
  }

  void Start()
  {

  }

  void Update()
  {
    switch (CurrentState)
    {
      case GameState.Wait:
        // Nous n'avons aucun traitement à faire ici
        break;

      case GameState.Started:
        // Le traitement des sphères sera à faire ici
        break;
    }
  }

  // Gestion du clic sur le bouton
  public void OnStartClicked()
  {
    // Chargement de la scène de jeu
    Application.LoadLevel("GameScene");
    CurrentState = GameState.Started;
  }
}
```

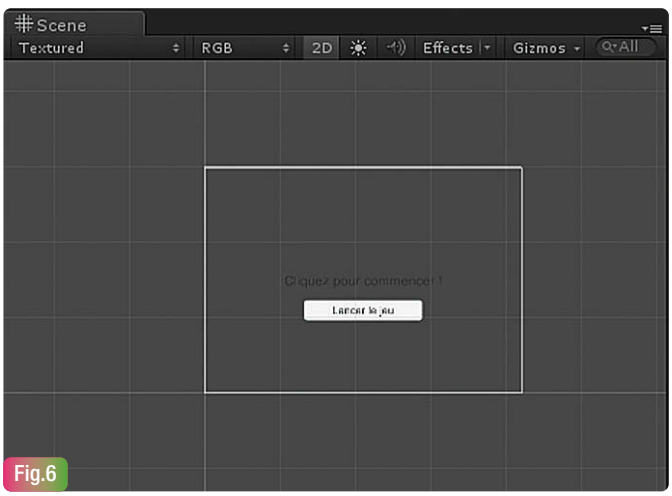


Fig.6

Nous allons créer un nouvel objet vide dans la scène que l'on appellera « Manager » qui contiendra notre script *GameManager*. Pour cela, dans l'inspecteur d'un Gameobject nouvellement créé, cliquez sur « Add Component » et recherchez « GameManager » dans les scripts.

Gestion du click bouton

Pour gérer la gestion du clic sur le bouton, nous avons créé la méthode *OnStartClicked()* qui sera appelée lors du déclenchement de l'évènement. Pour lier cette fonction au bouton, sélectionnez le bouton et glissez et déposez l'objet *Manager* dans la partie *OnClick()* puis sélectionnez la méthode *OnStartClicked()* dans la liste de droite.

Création de préfabriqués

Les préfabriqués ou plus communément appelés *prefab,* sont utilisés pour créer des objets avec différents réglages qui seront stockés afin d'être réutilisables plus facilement. Concrètement, nous allons utiliser les *prefabs* pour nous deux types de sphères.

Nous allons commencer par créer deux nouveaux matériaux qui seront utilisés par nos *MeshRenderer* afin d'afficher une sphère avec une couleur différente. Dans le panneau « Projet » cliquez sur « Create à Material », renommez-le en *RedMaterial* et changez la couleur dans l'inspecteur. Faites de même pour le matériau noir et nommez-le *BlackMaterial*.

Une fois terminé, créez deux sphères *RedSphere* et *BlackSphere* et affectez à chacun son matériau respectif en faisant glissez le matériau sur l'objet voulu. Vérifiez dans la partie *Transform*, que les positions des deux boules soient à 0 et que leurs tailles soient à 1.

Nous avons donc deux sphères avec des réglages différents, pour en faire des préfabriquées rien de plus simple : faites glissez les objets de la fenêtre *Hierarchy* à la fenêtre *Project*. Faites l'opération inverse pour ajouter un *prefab* dans votre scène.

Si vous souhaitez éditer les valeurs de vos *prefabs*, n'oubliez pas de

cliquer sur le bouton « *Apply* » en haut de l'inspecteur pour enregistrer les modifications.
Créez une nouvelle scène à partir du menu « *File à New Scene* » que vous appellerez *GameScene,* ce sera notre scène de jeu.

Réglage de la caméra

En créant la scène, l'objet « *Main Camera* » a été automatiquement créé et ajouté par Unity. Cet objet comporte un composant de type *Caméra* qui va servir à afficher le rendu de notre scène. Vous pouvez ajouter un nombre illimité de caméras qui pourront afficher tout ou partie de l'écran, un type d'objet particulier, seulement l'interface utilisateur, ... les possibilités sont très larges.
Il existe différents types d'affichages qui peuvent être changés en utilisant le paramètre « *Clear Flags* » :

- **Skybox** : c'est le type de caméra utilisée par défaut. Elle consiste à afficher votre scène comme si elle était enfermée dans une boîte avec un fond de couleur, un peu comme le ciel,
- **Solid Color** : en utilisant cette option de caméra, chaque partie de l'écran vide, c'est-à-dire sans aucun rendu d'objets s'affichera de la couleur prédéfinie,
- **Depth Only** : permet d'afficher un rendu à partir d'une certaine profondeur uniquement,
- **Don't Clear** : ce mode est un peu spécial car il va faire en sorte de ne pas effacer l'ancien buffer de rendu pour afficher le nouveau rendu, il va réécrire par-dessus.

Si vous souhaitez afficher qu'un ou plusieurs types de couches (= *layers*), vous pouvez le faire en réglant la propriété « **Culling Mask** ».
Deux types de projections sont possibles via le paramètre « **Projection** » : orthographique et perspective. La première est très utilisée pour les jeux en 2D car elle permet de supprimer l'effet de perspective qui rend les objets plus petit avec la distance. La seconde permet d'avoir un rendu réaliste pour les rendus en 3D.
Positionnez donc la caméra en X = *0, Y = 0, Z = -10 (un peu en arrière)* et mettez le type de projection en mode perspective. Positionner la caméra en négatif sur l'axe Z permet de la reculer. On pourra alors par la suite ajouter les objets en laissant leur profondeur à 0 pour qu'ils soient affichés.

Détection de collision

Ajoutez une sphère rouge au centre de la scène en glissant-déposant le préfabriqué *RedSphere* sur la scène. Le principe du jeu est de cliquer sur les sphères, nous allons donc avoir besoin de détecter quand l'utilisateur appuie sur celles-ci. Pour ce faire, nous allons devoir utiliser la technique du lancer de rayon (= *Raycasting*). Cela consiste à créer une droite fictive qui sera positionnée à la perpendiculaire par rapport à notre écran et placée au niveau de notre pointeur de souris. On cherchera alors le premier élément touché par cette droite si une collision existe.

Récupération des évènements de la souris

Pour savoir quand l'utilisateur va faire un clic gauche ou lors d'un appui sur une surface tactile, nous allons utiliser la classe *Input* qui nous fournira tous les renseignements que l'on souhaite. Voici les propriétés ou méthodes qui vont nous intéresser :

- **TouchCount** : retourne le nombre doigts détectés pour la frame courante,
- **Touches** : contient la liste de tous les doigts détectés sur l'écran. Chaque entrée de la liste étant de type *Touch*,
- **MousePosition** : obtient les coordonnées du curseur de la souris,
- **GetMouseButtonDown(int button)** : retourne vrai si le bouton de la souris est détecté comme cliqué. L'état est réévalué à chaque frame et pour pouvoir détecter le bouton comme cliqué une nouvelle fois, le bouton doit repasser dans l'état non cliqué. Le paramètre correspond au type de bouton de la souris : 0 pour la gauche, 1 pour la droite, 2 pour celui du milieu.

Nous aurons aussi besoin de connaître l'état de l'appui tactile : début, stationnaire, en mouvement, fin, ...
Grâce à toutes ses informations on peut écrire notre test qui permettra de détecter aussi bien un clic qu'un doigt.

```
if ((Input.touchCount == 1 && Input.touches[0].phase == TouchPhase.Began) || Input.Get
MouseButtonDown(0))
{
    // Tester la collision avec une sphère
}
```

Raycasting

Une fois que l'on sait que le joueur a effectué l'action souhaitée, on peut appliquer la méthode du lancer de rayon pour détecter les collisions. Voici un petit script qui permet de réaliser ce que l'on souhaite :

```
// On crée un rayon à partir de la position de la souris
var ray = Camera.main.ScreenPointToRay(Input.mousePosition);
// Pour tester le tactile on utilisera plutôt ceci
// var ray = Camera.main.ScreenPointToRay(Input.touches[0].position);
RaycastHit hit;

// On vérifie la collision avec un objet et, si elle existe, on utilise son nom
// pour déterminer s'il s'agit d'une boule rouge ou noire
if (Physics.Raycast(ray, out hit))
{
  if (hit.collider.gameObject.name == "RedSphere")
  {
    // On créé deux nouvelles sphères
  }
  else if (hit.collider.gameObject.name == "BlackSphere")
  {
    // On a perdu, on revient à la scène d'introduction
    Application.LoadLevel("MainScene");
    GameManager.Instance.CurrentState = GameState.Wait;
  }
}
```

Instancier des préfabriqués

Maintenant que l'on sait déterminer si une sphère rouge ou noire a été touchée, il ne nous reste plus qu'à savoir comment en créer directement à partir de notre script si la boule touchée est rouge. Encore une fois l'opération est extrêmement simple et tient à une seule méthode : « *Instantiate* ».
Cette fonction prend en paramètre au minimum un objet de type Object qui sera cloné et retourné en valeur de retour de la méthode. Dans notre cas, nous lui passerons en paramètre nos deux préfabriqués *RedSphere* et *BlackSphere* grâce à deux propriétés publiques que l'on va créer dans notre classe *GameManager*.

```
public GameObject RedSpherePrefab;
public GameObject BlackSpherePrefab;
```

Pour les lier à nos préfabriqués, dans la scène MainScene, sélectionnez *Manager* dans la hiérarchie des objets et faites glisser vos *prefabs* dans

les champs de l'inspecteur. Notre objectif étant de créer deux sphères aux dimensions et tailles aléatoire, voici comment avec une méthode qui permet d'obtenir ce résultat :

```
private void SpawnSpheres()
{
   // Pour créer un nouveau GameObject on utilise la méthode Instanciate()
   // plutôt que de faire new GameObject();
   var redSphere = Instantiate(RedSpherePrefab) as GameObject;
   redSphere.name = "RedSphere";
   redSphere.transform.position = GetRandomPosition();
   redSphere.transform.localScale = GetRandomSize();

   var blackSphere = Instantiate(BlackSpherePrefab) as GameObject;
   blackSphere.name = "BlackSphere";
   blackSphere.transform.position = GetRandomPosition();
   blackSphere.transform.localScale = GetRandomSize();
}

/// <summary>
/// Retourne une position aléatoire
/// </summary>
/// <returns>Positionnement sous forme de vecteur</returns>
private Vector3 GetRandomPosition()
{
   var randomX = Random.Range(-5, 5);
   var randomY = Random.Range(-5, 5);
   var randomZ = Random.Range(-5, 5);

   return new Vector3(randomX, randomY, randomZ);
}

/// <summary>
/// Retourne une taille aléatoire
/// </summary>
/// <returns>Taille sous forme de vecteur</returns>
private Vector3 GetRandomSize()
{
   var randomX = Random.Range(0.5f, 1.5f);
   var randomY = Random.Range(0.5f, 1.5f);
   var randomZ = Random.Range(0.5f, 1.5f);

   return new Vector3(randomX, randomY, randomZ);
}
```

Conclusion de l'exemple

Grâce à ce petit jeu simpliste nous avons pu aborder quelques principes de base d'Unity comme la caméra, les préfabriqués, les scènes et l'intégration de scripts personnalisés. Il nous reste à voir comment utiliser des plugins et bien entendu, comment générer un exécutable de notre jeu !

Utiliser des plugins

Qu'est-ce que l'asset store ?

Vous êtes peut être habitués à télécharger des applications sur le Store de votre smartphone : l'Asset Store est la même chose pour des composants réutilisables dans vos différents projets. Certains sont gratuits, certains sont payants mais il s'agit d'un incontournable à visiter au début de votre projet pour potentiellement gagner du temps pendant vos développements.

Parmi les choses que vous pouvez trouver, il y a notamment :

- Modèles 3D préfabriqués,
- Animations et systèmes d'animations d'objets et de propriétés,
- Sons, musiques et effets sonores,
- Projets complets (exemples et tutoriaux notamment),
- Systèmes de particules avancés,
- Scripts préfabriqués,
- Accès à des services tiers (Facebook, Twitter, In App product de chaque plateforme),
- Shaders,
- Textures et matériaux.

Souvent, l'avantage d'un plugin ou d'un asset est qu'il est disponible sur toutes les plateformes que vous ciblez. Il est par exemple commun de savoir effectuer une action sur une plateforme (Windows Phone par exemple) et de ne pas savoir la faire sur une autre (iOS par exemple); avoir un plugin complet pour les différentes plateformes est un avantage certain ! Un autre avantage est qu'Unity permet une installation simplifiée depuis son Store.

Comment installer un plugin ?

Le plus simple pour parcourir le Store Unity est peut-être d'utiliser le site Web dédié : https://www.assetstore.unity3d.com/. Utilisez le champ de recherche, parcourez les catégories, et une fois votre choix fait, il suffit de cliquer sur « Open in Unity ». Ici, on va installer le SDK Facebook : Fig.7. Cela a pour effet de m'ouvrir la même page dans Unity qui me propose alors de télécharger le package en cliquant sur le bouton « Download »; attention, le chargement peut être long et faire croire qu'Unity est parfois figé. Après m'être connecté sur le Store, ce qui me permet notamment de conserver mes achats sur mes différents projets, le téléchargement et l'installation du SDK se déclencheFig.8.

J'ai alors une fenêtre me demandant ce que je souhaite importer dans mon projet. Il n'est pas rare d'avoir des scènes d'exemples dans les plugins que vous utilisez et je vous conseille de laisser la sélection par défaut et de cliquer sur « Import ». Unity va alors travailler un petit peu pour importer les différents éléments.

L'import a alors créé une série de dossiers et ajouté les fichiers du plugin.

Fig.7

Fig.8

La hiérarchie de dossiers et le contenu exact dépendent bien sûr du plugin mais dans notre cas, trois dossiers sont créés :

- « Examples » : il contient les exemples fournis par Facebook. Un bon endroit pour commencer,
- « Facebook » : contient les scripts et ressources spécifiques à Facebook (en C#),
- « Plugins » : contient les binaires spécifiques à chaque plateforme. Ne soyez pas surpris de ne voir qu'Android dans notre cas, car l'intégration Facebook est de base dans iOS, et le plugin ne gère pas encore les plateformes Windows.

L'utilisation du plugin est alors spécifique au plugin mais consiste généralement à appeler du code C# classique. Dans notre cas il est par exemple possible de demander la connexion de l'utilisateur de cette façon :

```
FB.Login("email,user_friends", LoginCallback);
```

Génération des binaires du jeu

Il est maintenant temps de générer les différents binaires pour aller les déposer sur les Stores et commencer à devenir millionnaire ! La génération commence par l'écran de paramétrages des binaires «File à Build Settings » qui ouvre une fenêtre avec 3 zones différentes :

- **Scenes in build :** permet de sélectionner les scènes à incorporer dans le binaire final. Il pourrait par exemple être intéressant de faire une scène en fonction du format de la plateforme : tablettes, PC, consoles ou mobiles. Une façon de le faire serait de passer par des scènes. En ajoutant moins de scènes dans un package, vous en réduisez sa taille,
- **Platform :** la liste des plateformes disponibles pour la génération d'un binaire. Pour chaque plateforme, Unity recompile les différents assets et en conserve une « courante ». Vous pouvez donc soit choisir une plateforme pour générer le binaire, soit définir la plateforme sélectionnée comme courante (celle que vous utilisez le plus souvent) : « set as current ».
- **Détails de la plateforme :** pour chaque plateforme certains paramètres sont disponibles et notamment pour créer un package de « développement » avec des logs supplémentaires, tout comme la possibilité de brancher le profiler ou un package « release » plus rapide et parfait pour les Stores.

En cliquant sur « Build », vous générerez un package dans le format ciblé; en cliquant sur « Build and Run », vous déploierez en sus le package sur un éventuel device branché.

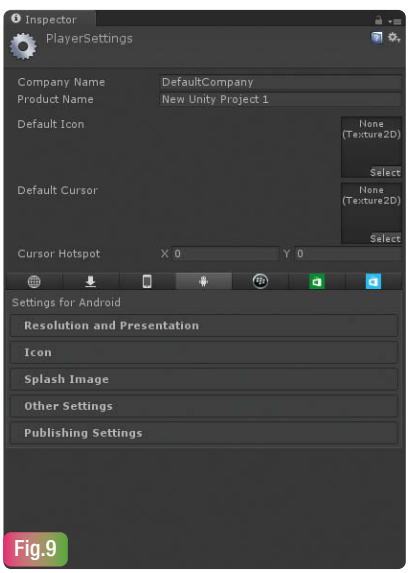


Fig.9

Paramètres du player Unity

Le player Unity est le composant qui exécute votre jeu sur chacune des plateformes; Unity vous permet de le personnaliser en fonction de chacune d'entre elles. Cela est très important pour des choses classiques comme de choisir une icône adaptée à la plateforme mais aussi pour des techniques plus avancées comme de choisir un format de compression des textures ou des sons de votre jeu en fonction des plateformes.

Lorsque vous sélectionnez une plateforme, vous pouvez cliquer sur le bouton « Player Settings » qui fera apparaître une fenêtre de configuration avec des informations génériques (en haut), et des onglets pour chaque plateforme en dessous. En cliquant sur une plateforme (par exemple Windows Phone), il est possible de définir chacun des paramètres spécifiques de celle-ci : icônes, splashscreen, noms sur les tuiles, informations de publication, etc. Fig.9.

Les binaires mobiles les plus courants

Nous allons voir ici le processus pour les trois plateformes mobiles les plus courantes.

Windows (Phone)

La génération est très simple : en cliquant sur "Build", Unity vous demande de choisir un dossier où il placera une solution Visual Studio complète. Vous n'avez alors plus qu'à ouvrir ce SLN, personnaliser éventuellement le code, et générer le package pour vos tests (déploiement sur une tablette par exemple) ou la publication sur le Store (Association avec le Store, signature et création).

Android

Unity est capable de vous produire directement l'APK final de déploiement sur le Store, ou de générer un projet de développement Android pour d'éventuelles personnalisations. Si c'est ce que vous souhaitez, il suffit de cocher la case « Google Android Project ».

La suite du déploiement est la même que pour un package Android classique.

iOS

Dans ce cas, Unity est capable de générer un projet XCode complet prêt à la compilation. Une fois le projet réalisé, XCode s'ouvre automatiquement et vous pouvez choisir de le déployer sur vos devices, ou de créer une archive pour le déploiement sur le Store. D'expérience, notamment avec le SDK Facebook, il est recommandé d'effectuer cette génération depuis la version OS X d'Unity.

Attention, comme expliqué plus tôt, Unity ne supporte pas encore la version x64 bientôt obligatoire sur le Store d'Apple.

Tailles des binaires

Une préoccupation lors de la création de jeux mobiles est la taille du binaire final : Unity est très doué pour réduire au maximum celle-ci. Il est capable de n'inclure dans le package que les éléments réellement utilisés (images, sons, etc.) et d'exclure ceux qui ne le sont pas. De même, par défaut, il sait choisir des formats de compression optimisés pour chacune des plateformes afin de réduire au maximum la taille des packages pour une qualité finale optimale.

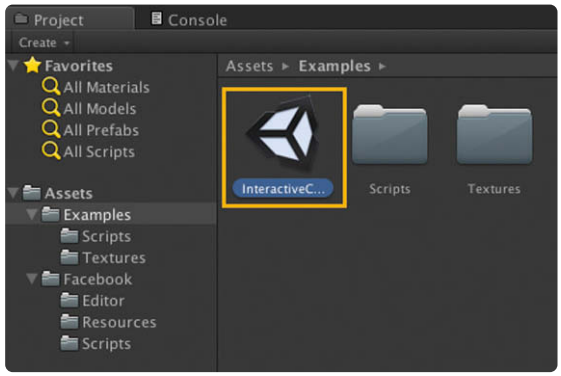
Conclusion

Dans cet article nous avons pu voir qu'Unity est vraiment la plateforme de rêve pour les créateurs de jeu vidéo : simplicité, productivité et performances sont au rendez-vous ! La possibilité d'utiliser C#/Visual Studio pour l'écriture de script est vraiment la cerise sur le gâteau.

Enjoy !

Intégration Facebook - Unity

Les réseaux sociaux, spécialement Facebook, jouent de plus en plus un rôle important au sein des applications mobiles. En effet, il permet de socialiser le jeu sans qu'il soit forcément multi-joueurs. Ils permettent l'échange d'informations relatives au jeu telles que le score, la progression. Le joueur sera capable d'inviter et/ou de challenger ses amis, ou encore de faire apparaître son score sur son mur. Ainsi, ces nouvelles fonctionnalités bonifient l'expérience du joueur.



Anthony Legiret
Développeur **Magma Mobile**

Installation

Avant de commencer, il nous faut les outils :
Unity : http://unity3d.com/unity/download
Le SDK Facebook : https://developers.facebook.com/docs/unity/downloads
Je conseille fortement de vous rendre sur le site développeur de Facebook afin de trouver un approfondissement des fonctions utilisées.
La documentation est consultable à cette adresse : https://developers.facebook.com/docs/unity.
Après avoir installé, et téléchargé le SDK Facebook, Ouvrez un projet vide avec Unity 3D et importez le package dans Unity 3D (Assets > Import Package > Custom Package).

Création d'une application Facebook

Afin de créer une application Facebook, il vous suffit de vous rendre sur le site développeur de Facebook : https://developers.facebook.com/. Créez une nouvelle application en passant les détails, qui sont obligatoires seulement lors de la mise en production. Après la création de l'application, vous serez sur le dashboard (tableau de bord) de votre application. Ce qui nous intéresse principalement, c'est l'identifiant de l'application (App ID). De retour sur Unity 3D, la configuration du SDK Facebook se trouve dans la barre de menu, à Facebook > Edit Settings. Il suffira de donner le nom et l'identifiant de l'application, précédemment configurée dans le dashboard Fig.1.

Un petit tour des fonctions principales

- FB.Login() : permet de se connecter a Facebook et de demander des permissions,
- FB.Logout() : permet de se déconnecter de Facebook,
- FB.Feed() : permet de poster un message personnalisé sur le mur de la personne connectée,
- FB.AppRequest() : permet d'envoyer une requête à des amis,
- FB.Api() : permet d'utiliser l'API REST de Facebook. vous trouverez ici toutes les requêtes possible : https://developers.facebook.com/docs/graph-api/reference.

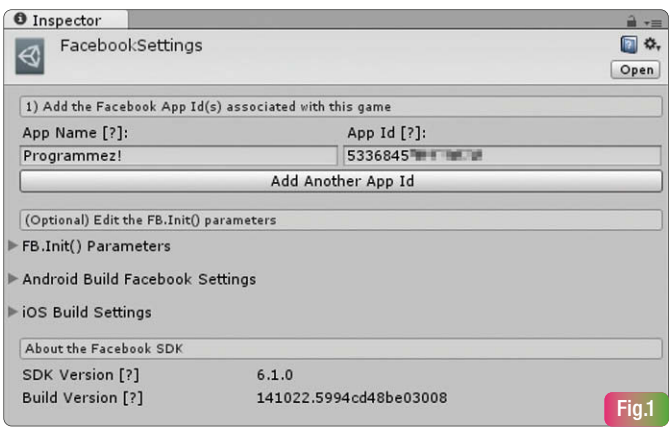


Fig.1

```
public class FacebookScript : MonoBehaviour {

    void Awake()
    {
        // Initialisation du SDK Facebook
        FB.Init(onInitCompleted);
    }

    // Callback lorsque le sdk a fini de s'initialiser
    void onInitCompleted()
    {
        Debug.LogWarning("Initialization completed");

        // Permissions permettant d'acceder aux informations de l'utilisateur
        string permissions = "email,public_profile,publish_actions";

        // On va se connecter juste après que le sdk soit initialisé
        if (!FB.IsLoggedIn)
            FB.Login(permissions, loginCallback);
    }

    // Callback d'une connection
    void loginCallback(FBResult result)
    {
        if (string.IsNullOrEmpty(result.Error))
        {
            Debug.LogWarning("success : " + result.Text);
        }
        else
        {
            Debug.LogWarning("Error : " + result.Error);
        }
    }
}
```
Fig.2

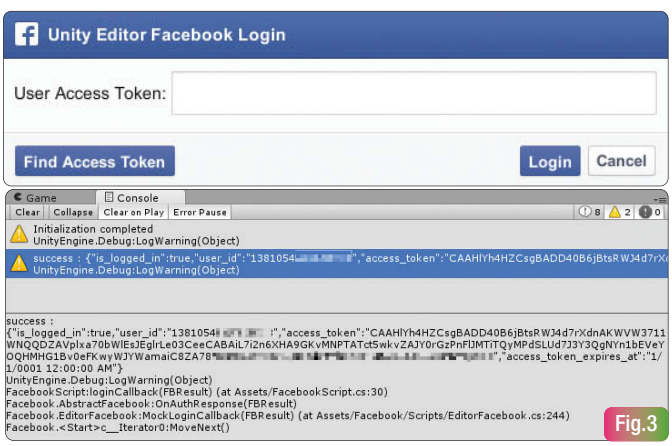


Fig.3

Résultat lors du lancement du jeu, un pop-up me demande de me connecter grâce au "user access token" que vous pourrez avoir en cliquant sur le bouton "Find Access Token".

Initialisation et login Fig.2

La fonction “Awake” est une fonction appelée par le moteur Unity lors du lancement du jeu. A ce moment-là, nous initialisons le SDK Facebook afin qu’il puisse se configurer grâce à l’App ID précédemment renseignée. Une fois que l’initialisation est terminée, la fonction callback “onInitCompleted” est appelée et se connectera seulement si nécessaire. En effet, le SDK se reconnecte automatiquement si le joueur quitte l’application en étant connecté.
Vous avez pu le remarquer, le SDK est en asynchrone et fonctionne avec des callbacks. Le callback des fonctions du SDK possède toujours un objet FBResult en paramètre, qui permet d’avoir le résultat de la requête envoyée Fig.3.

Envoyer une invitation Fig.4.

Grâce à la fonction “AppRequest”, il est possible d’envoyer des invitations, des cadeaux à ses amis. Ici, nous enverrons le message “come play with me” aux amis sélectionnés par la pop-up ouverte automatiquement ci-dessous Fig.5.

Écrire sur le mur Fig.6.

Cette méthode, à l’apparence très simple, possède de nombreux paramètres afin de personnaliser au maximum l’information à partager. En effet, il est possible d’afficher une image (comme dans l’exemple ci-dessous), mais aussi un titre, une description, une légende. D’autres paramètres permettent l’affichage d’autres médias sources tels que de l’audio ou de la vidéo Fig.7.

Partage de score Fig.8.

Principalement utilisé pour les jeux, l’API nous permet de sauvegarder un score (entier 32bits). Afin de modifier le score du joueur, il suffit d’utiliser l’API Rest de Facebook. Le premier paramètre de la fonction “FB.API” est la requête vers l’API. “me” représente l’utilisateur courant, “scores” représente le score du joueur. La structure d’une requête vers l’API est assez intuitive. Le second paramètre est le type de requête (POST, GET, DELETE). Nous avons ensuite, la fonction de callback et les paramètres qui sont envoyés grâce à un objet Dictonnary.
Il est aussi possible de récupérer ce score sauvegardé, ainsi que celui des amis grâce à l’API. Idée : il peut être sympa de créer un tableau de scores des amis, avec un classement selon le score de chacun.

Pour aller plus loin

Facebook demande aux développeurs de suivre leur “guide-line” afin de proposer une meilleure expérience de jeu. Je vous conseille de bien lire ces permissions, chaque action demande certaines permissions spécifiques et certaines requièrent une vérification par Facebook pour la publication. Eh oui, les données que vous manipulez sont privées et confidentielles. Facebook veut vérifier que vous ne faites pas n’importe quoi avec l’api. https://developers.facebook.com/docs/facebook-login/permissions

```
// Envoie une requete aux amis selectionnés
public void inviteFriends()
{
    FB.AppRequest(
        message : "come play with me",
        title: "Programmez !"
        );
}
```

Fig.4

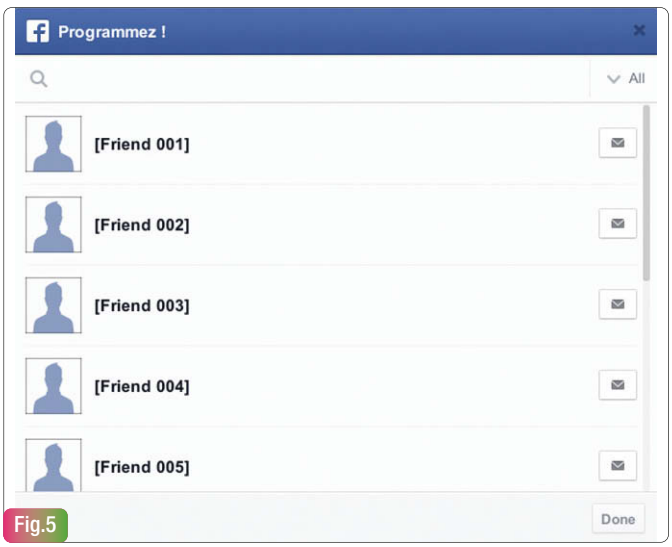


Fig.5

```
public void FeedStatus()
{
    FB.Feed("","http://www.programmez.com/","Programmez","Caption","Description",
            "http://mydomainename.com/img.jpg");
}
```

Fig.6

Fig.7

```
public void sendScore(int score)
{
    // Parametres d'envoie du score
    Dictionary<string,string> param = new Dictionary<string, string>();
    param.Add("score",score.ToString());

    // Utilisation de l'api pour envoyer le score
    FB.API("/me/scores",Facebook.HttpMethod.POST,sendScoreCallback,param);
}

void sendScoreCallback(FBResult result)
{
    if (string.IsNullOrEmpty(result.Error))
    {
        Debug.LogWarning("Score Posted : " + result.Text);
    }
    else
    {
        Debug.LogWarning("Error : " + result.Error);
    }
}

public void getScore()
{
    // Utilisation de l'api pour recevoir le score
    FB.API("/me/scores",Facebook.HttpMethod.GET,getScoreCallback);
}

void getScoreCallback(FBResult result)
{
    if (string.IsNullOrEmpty(result.Error))
    {
        // le resultat est en json
        // On parse le json pour avoir seulement le score
        JSONNode json = JSONNode.Parse(result.Text);

        currentScore = int.Parse(json["data"][0]["score"]);
        Debug.LogWarning(json["data"][0]["score"]);
    }
    else
    {
        Debug.LogWarning("Error : " + result.Error);
    }
}
```

Fig.8

Complétez votre collection PROGRAMMEZ!

le magazine du développeur

Collector N°150

Quantité limitée

Prix unitaire : 6 € (Frais postaux inclus) - **France métropolitaine uniquement.**

nouveau

Tout Programmez! sur une clé USB

Tous les numéros de Programmez! depuis le n°100.

29,90 €*

Clé USB 2 Go.
Testé sur Linux, OS X, Windows. Les magazines sont au format PDF.

* *tarif pour l'Europe uniquement. Pour les autres pays, voir la boutique en ligne*

Vous pouvez aussi commander directement sur notre site internet : www.programmez.com

☐ ***175*** : |__| exemplaire(s)
☐ ***177*** : |__| exemplaire(s)
☐ ***179*** : |__| exemplaire(s)
☐ ***180*** : |__| exemplaire(s)
☐ ***181*** : |__| exemplaire(s)
☐ ***150*** : |__| exemplaire(s)

☐ ***La Clé USB avec les numéros de Programmez depuis le n°100***
29,90 € (Tarif pour Europe uniquement)

Commande à envoyer à :
Programmez!
7, avenue Roger Chambonnet
91220 Brétigny sur Orge

soit |__|__| exemplaires x 6 € = |__|__| € + ☐ 29,90 € soit au **TOTAL** = |__|__|__|__|__| €

Prix unitaire : 6 € (Frais postaux inclus), France métropolitaine uniquement.

Tarifs France métropolitaine

PROG 182

☐ M. ☐ Mme ☐ Mlle Entreprise : |__|__|__|__|__|__|__|__| Fonction : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Prénom : |__|__|__|__|__|__|__|__|__|__|__|__| Nom : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Adresse : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Code postal : |__|__|__|__|__| Ville : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Tél : |__|__|__|__|__|__|__|__|__|__| ***(Attention, e-mail indispensable)***

E-mail : |__|__|__|__|__|__|__|__|__|__|__|__|__|__| @ |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Règlement par chèque à l'ordre de Programmez !

Valable jusqu'au 28 février 2015
Offres pouvant s'arrêter à tout moment, sans préavis

Créer des plugins et des widgets pour Wordpress

Nous avançons étape par étape, de la personnalisation, en passant par l'administration et la création de thème. Nous voilà donc arrivés à la création de plugin et de widget. Je vous invite à découvrir et à vous initier à ces deux sujets grâce à cet article.

Freddy Hebrard
Responsable Système d'information centre des ressources régional de Lorraine
MCPD/ MCITP/MCTS Microsoft

Dans Wordpress 4, un plugin est un moyen simple et efficace de rajouter de nouvelles fonctionnalités à vos sites / blogs. Un Plugin peut être réutilisable sur un autre site Wordpress.
La création d'un plugin :
Au même titre que les thèmes, le dossier du plugin ainsi que la structure des fichiers obligatoires doit répondre à une certaine nomenclature afin de le rendre fonctionnel dans notre environnement.

Etape de création du plugin

- Comme nous l'avons vu ensemble dans les précédents numéros de Programmez !, le répertoire qui accueille les thèmes se situe à cette adresse : « [racine_wordress]\ wp-content\themes\ ».
- Dans ce répertoire, ajouter un nouveau répertoire au nom de votre plugin (ex : programmez).
- Ajouter un fichier de base .php (par convention, nommez-le avec le même nom que le dossier, soit le nom de votre plugin).
- Insérer le code de définition de votre plugin dans ce fichier.

```
<?PHP
/*
Plugin Name: Programmez!
Plugin URI:  http://www.programmez.com/
Description:  Mon premier plugin grace a programmez!
Author:  Programmez Team
Author Uri: http://www.programmez.com/fhebrard
Licence: GPL2
Version:  Beta 1
*/
```

(Code : Définition de base dans le fichier de définition de notre plugin (programmez.php))

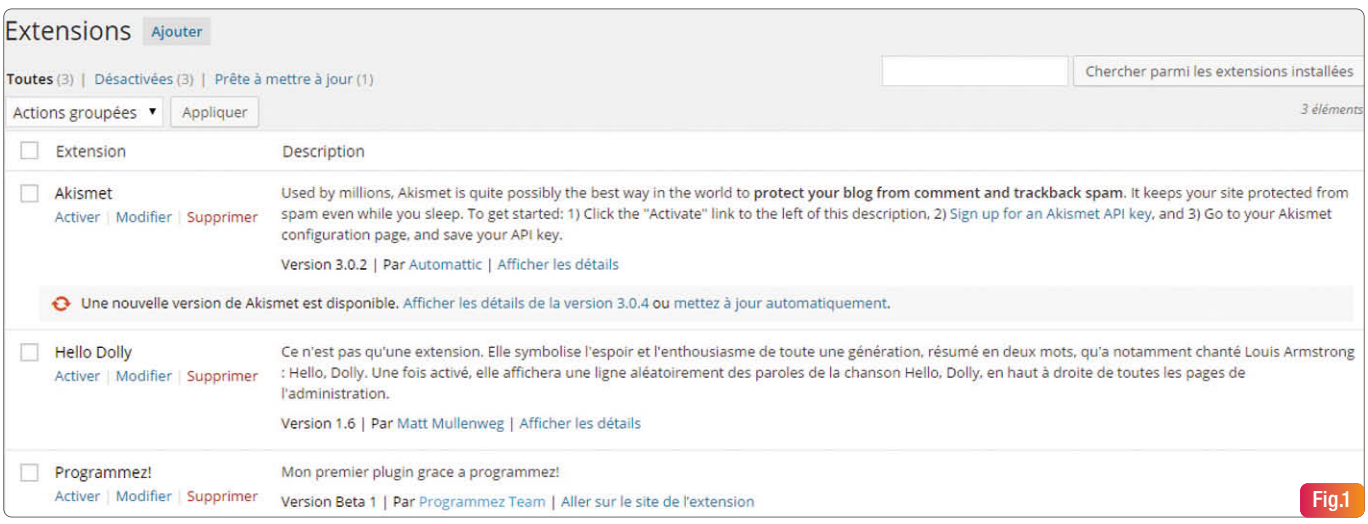
Dans la définition d'un plugin, seule la ligne « Plugin Name : » est obligatoire pour la détection de notre plugin par Wordpress.
- Plugin Name est le nom du plugin.
- Plugin URI est l'adresse du plugin pour donner plus d'informations et une description de celui-ci.
- Description : histoire de savoir à quoi il sert....
- Version : version de votre plugin, vous gérez vous-même le format.
- Author Uri : l'adresse qui parle de vous, votre blog etc....
- License : le nom de la licence utilisée pour votre plugin.

- Est-il détecté et activable ? vérifiez-le comme vous l'avez appris dans les précédents articles dans l'administration de notre Wordpress / (menu gauche) Extensions -> extensions installées
  Sur la page centrale, notre plugin doit apparaître avec les informations que vous avez renseignées. Fig.1.

Bravo! Vous venez de créer votre premier plugin.... Mais il ne sert strictement à rien !

Créons un plugin qui fait quelque chose

Ce qu'il faut savoir avant de commencer à remplir votre fichier de code, c'est que tout élément déclaré dans votre fichier de plugin et dont le plugin est activé sera accessible de n'importe où au sein de votre structure Wordpress...
C'est à ce moment qu'il faut que vous apposiez une bonne pratique avant d'aller plus loin ! Si vous écrivez un plugin et que vous faites appel directement aux fonctions de celui-ci dans un thème, une page ou dans un autre plugin de votre site, vous allez créer un couplage fort ! Vous allez générer des dépendances et votre code sera moins réutilisable.
Utilisez la puissance du moteur de Wordpress et son système modulaire pour faire appel à nos fonctions/méthodes, nous avons vu dans nos précédents articles les hooks (pour rappel ce sont des points d'entrée dans une fonction Wordpress dans lesquels nous greffons nos fonctions), il y a deux types de Hooks : les actions et les filtres.



Administration des plugin)

Voici les signatures pour rappel :

```
add_action($tag, $function_to_add, $priority, $accepted_args) ;
add_filter($tag, $function_to_add, $priority, $accepted_args);
```

Les actions : elles sont exécutées à des moments précis durant l'exécution.
Les filtres : plus ou moins pareils que les actions dans leurs fonctionnements mais servent à modifier du texte ou des types avant leur dans la BDD ou dans le navigateur.

Pour infos :

Liste des actions sur http://codex.wordpress.org/Plugin_API/Action_Reference
Liste des filtres sur http://codex.wordpress.org/Plugin_API/Filter_Reference
Exemple d'un premier code fonctionnel qui permet de modifier un titre de page :
Dans notre fichier programmez « programmez.php » ajoutez :

```
add_filter('wp_title',array($this,'fonctionChangeTitre'),20);
public function fonctionChangeTitre($titreActuel){
    return $titreActuel . ' | Avec mon premier plugin !';
}
```

(Code : ajout d'un filtre)

La fonction « fonctionChangeTitre » vous permet une fois appelée par le filtre 'wp_title' de modifier le titre de votre page, il prend en argument le titre actuel ($titreActuel) et retourne la modification du titre. La fonction add_filter() permet de rattacher notre fonction au filtre « wp_title »
Résultat, en ouvrant votre site, regardez dans le code, dans la section <head> :

```
« <title> | Avec mon premier plugin !</title> »
```

Pour les projets de plus grande envergure, faites-le avec « class ».
Vous pouvez développer simplement en utilisant du code procédural mais aussi en développant en utilisant la POO (cf 1ere partie sur Wordpress). Dans le cadre de projet de grande envergure, je vous conseille pour un code plus clair ce dernier.
Je vous donne un exemple simple pour l'utilisation avec un filtre :

```
<?PHP
/*
Plugin Name: Programmez!
Plugin URI:   http://www.programmez.com/
Description:  Mon premier plugin grace a programmez!
Author:       Programmez Team
Author Uri:   http://www.programmez.com/fhebrard
Licence:   GPL2
Version:   Beta 1
*/

class monPremierPluginProgrammez
{
  //le constructeur
  public function __construct(){
     //notez l'utilisation d'un array pour passer notre méthode en parametre
     add_filter('wp_title',array($this,'fonctionChangeTitre'),20);
  }
  public function fonctionChangeTitle($titreactuel){
     return $titreActuel . ' | Avec mon premier plugin !';
  }
}
//ne pas oublier d'instancier notre objet
new  monPremierPluginProgrammez();
```

(code : Utilisation POO dans l'ecriture d'un plugin)

- Dans ce code, vous devez être attentif à deux choses : le passage en paramètre du nom de notre méthode passe par un « array » avec l'instance de notre objet ($this) puis le nom de notre méthode.
- Deuxième élément n'oubliez pas d'instancier votre objet.

LES WIDGETS

Qu'est-ce qu'un widget ?

Un widget est une unité graphique qui permet de déplacer les blocs de fonctionnalités sur les différentes zones d'accueil de notre blog ou site Wordpress.
Ils nécessitent, comme les plugins et les thèmes, un peu de travail en amont pour les rendre fonctionnels.

Création de widget

Nul besoin de fichier ou dossier particulier disposé à un endroit précis de notre structure Wordpress. Un widget est un objet qui hérite de la "class" « WP_Widget ».
4 étapes importantes lors de l'écriture de notre "class", pour le rendre fonctionnel:

- Au moment de l'instanciation de notre widget, notre constructeur doit définir :
  - Un identifiant
  - Un titre (utile dans l'administration de Wordpress)
  - Des paramètres supplémentaires
- Notre class doit impérativement surcharger la méthode widget()
- Déclarer à Wordpress que avons un nouveau widget à utiliser via l'appel de la fonction « register_widget('[nom_de_notre_class]') »
- L'appel de register_widget doit se faire à un moment très précis dans les évènements de Wordpress, au moment de l'initialisation des widget 'widgets_init' pour cela on utilise les actions et une fonctionnalité puissante de php (fonctions anonymes) :
  - add_action('widgets_init',
    function(){register_widget('[nom_de_notre_class]') ;}) ;

```
<?PHP
class Programmez_Widget extends WP_Widget{
  public function __construct(){
     parent::__construct('Programmez_widget','mon widget', array(
        'description'=> 'mon premier widget par programmez'
     ));
  }
  public function widget($args, $instance){
     echo 'mon premier widget'
  }
}
```

(Code : votre "class" widget de base)

Le visuel de notre widget

- La méthode Widget()

Si vous utilisez votre widget tel quel, nous n'aurons que l'affichage de 'mon widget by programmez ', cela n'est pas très intéressant, dans cette section nous allons nous occuper du visuel de notre Widget.
Comme vous l'aurez compris la surcharge de la méthode widget() permet

de définir le visuel de notre widget sur nos pages .

```
public function widget($args, $instance){
      echo $args['before_widget'];
      echo $args['before_title'];
      echo apply_filters('widget_title', $instance['title']);
      echo $args['after_title'];
      ?>
      <!--
          NOTRE CODE HTML A INTEGRER AU VISUEL ex un formulaire avec des champs de saisie
      -->
          <span>Hello mon premier widget Programmez !</span>
      <?PHP
      echo $args['after_widget'];

  }
}
```

(code : Méthode type widget() surchargée)

Revenons sur quelques points de ce code :

- echo $args[‘before_widget’] : permet d’afficher une chaîne de caractères passée en paramètre avant l’affichage du contenu de votre widget.
- echo $args [‘before_title] : avant le titre
- Apply_filters(‘widget_title’, $instance[‘title’]) : affiche notre titre après application du filtre (cf hook)
- echo $args[‘after_widget’] : chaîne de caractères à afficher après le rendu de notre widget.

A quel moment passez-vous ces données ? c’est simple, souvenez-vous l’article sur les thèmes Wordpress, la méthode « register_sidebar($args) » ($args) est un tableau de paramètres où vous pouvez déterminer les éléments qui seront appliqués sur l’ensemble des widgets contenus dans la sidebar insérée, ex de code tiré du codex sur la création des paramètres :

```
$args = array(
  'name'        => __( 'Sidebar name', 'theme_text_domain' ),
  'id'          => 'unique-sidebar-id',
  'description'   => '',
    'class'       => '',
  'before_widget' => '<li id="%1$s" class="widget %2$s">',
  'after_widget'  => '</li>',
  'before_title'  => '<h2 class="widgettitle">',
  'after_title'   => '</h2>' );
```

(Code : définition des paramètres de register_sidebar)

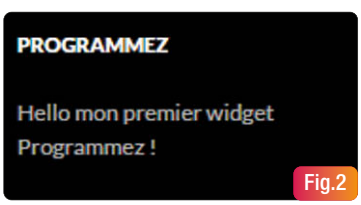


Fig.2 : *Rendu visuel sur une page de notre widget*

- la méthode form()

Il existe une autre méthode utile dans le cadre de la personnalisation de notre page, cette méthode à surcharger s’appelle form() il permet de créer un affichage lorsque que vous êtes en mode « personnalisation » de votre site, pour paramétrer par exemple le titre à afficher sur le widget.

Mon exemple va porter sur ce point : donner la possibilité à l’administrateur du site de pouvoir modifier le titre de son widget.
Voici le code :

```
< ?PHP
public function form($instance){
  $titre = isset($instance['title'])?$instance['title']:'';
  ?>
< !—ZONE HTML -->
  <p>
  <label for="<?PHP echo $this->get_field_name('title');?>"><?PHP _e('Title:');?>
</label>
<input class="widefat" id="<?PHP echo $this->get_field_id('title');?>" name="<?PHP
 echo $this->get_field_name('title');?>" type="text" value="<?PHP echo $title;?>"/>
  </p>
  < !—FIN ZONE HTML -->
<?PHP
}
```

(Code : exemple de la surcharge de form())

Revenons sur quelques points à comprendre dans ce code :

- get_field_id() et get_field_name() sont des méthodes de la "class" mère WP_Widget, elles permettent respectivement de générer un identifiant et un nom unique pour les champs utilisés lors de la sauvegarde des données dans Wordpress, il est très important de les utiliser afin que l’enregistrement des paramètres fonctionne.

Voici le résultat lors de l’administration de notre widget en mode « personnalisation » de notre site Fig.3 et 4.

Avec surcharge de form()

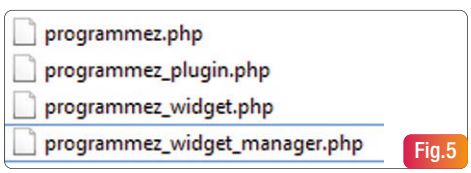
Sans la surcharge de form())

CAS PRATIQUE : Structurer notre premier projet pour intégrer nos widgets et un plugin

Il existe plusieurs méthodes pour intégrer nos widgets, je vais vous présenter une manière simple d’intégrer 1 ou x widgets en passant par un plugin et avec la POO.
L’objectif est de voir en pratique ce qu’on a vu sur l’ensemble de cet article :

- Création d’un plugin de base,
- Création d’un widget de base et son intégration à l’activation du plugin,
- Le petit plus : l’utilisation de plusieurs fichiers au sein d’un plugin.

Voici la structure de notre plugin « programmez » au sein de notre système de fichier Fig.5.



Le contenu de nos 4 fichiers et les explications :

- Programmez_plugin.php : le point d’entrée de notre plugin, il permet de faire appel au class/fichiers liés à notre plugin et widget que l’on souhaite insérer.

```
<?PHP
/*
Plugin Name:    Programmez!
Plugin URI:     http://www.programmez.com/
Description:    Mon premier plugin grace a programmez!
Author:         Programmez Team
Author Uri:     http://www.programmez.com/fhebrard
Licence:    GPL2
Version:    Beta 1
*/
include_once plugin_dir_path(__FILE__).'/programmez.php';
include_once plugin_dir_path(__FILE__).'/programmez_widget_manager.php';

class MonPremierPluginProgrammez
{
  public function __construct(){
     new Programmez();
     new Programmez_Widget_Manager();
  }
}
new MonPremierPluginProgrammez();
```

- Include_once plugin_dir_path(__FILE__) [....] : on inclut notre fichier comportant la class de notre plugin et le fichier qui gère les class de widgets à générer,
- Dans le constructeur on instancie nos plugins et notre gestionnaire de widget,
- On n'oublie pas d'instancier notre "class" de notre point d'entrée.
- Programmez.php : contenant la "class" de notre plugin qui instanciée dans la "class" de notre fichier « programmez_plugin.php »

```
<?PHP
class Programmez
{
  public function __construct(){
     add_filter('wp_title',array($this,'fonctionChangeTitre'),20);
  }

  public function fonctionChangeTitre($titreActuel){
     return $titreActuel . ' | Avec mon premier plugin !';
  }
}
```

- Notre plugin que nous avons présenté plus tôt dans l'article.
- Programmez_widget_manager : permet d'enregistrer les widgets à instancier sur l'évènement widget_init.

```
<?PHP
include_once plugin_dir_path(__FILE__).'/programmez_widget.php';

class Programmez_Widget_manager{
  public function __construct()
  {
        add_action('widgets_init',function(){
           register_widget('Programmez_Widget');
        });
  }
}
```

- On inclut notre fichier contenant la "class" de notre widget,
- Dans le constructeur de notre manager, nous demandons à Wordpress, d'enregistrer notre nouveau widget (noter l'utilisation de fonction anonyme PHP) lors de l'application de l'action 'widgets_init',
- Programmez_widget.php : contient une "class" exemple pour la génération d'un widget.

```
<?PHP
class Programmez_Widget extends WP_Widget{
  public function __construct(){
     parent::__construct('Programmez_widget','mon widget', array(
        'description'=> 'mon premier widget par programmez'
     ));
  }
  public function form($instance)
  {
     $titre = isset($instance['title'])?$instance['title']:'';
     ?>
     <p>
        <label for="<?PHP echo $this->get_field_name('title');?>"><?PHP _e('Title:');?
></label>
        <input class="widefat" id="<?PHP echo $this->get_field_id('title');?>" name
="<?PHP echo $this->get_field_name('title');?>"
        type="text" value="<?PHP echo $title;?>"/>
     </p>

     <?PHP
  }*/
  public function widget($args, $instance){
     echo $args['before_widget'];
     echo $args['before_title'];
     echo apply_filters('widget_title', $instance['title']);
     echo $args['after_title'];
     ?>
     <!--
        NOTRE CODE HTML A INTEGRER AU VISUEL ex un formulaire avec des champs de saisie
     -->
        <span>Hello mon premier widget Programmez !</span>
     <?PHP
     echo $args['after_widget'];

  }
```

- Notre "class" complète dont nous avons détaillé chaque partie précédemment.

CONCLUSION

Comme pour les thèmes, la construction des plugins et des widgets est simple mais demande de bien respecter une certaine nomenclature pour le bon fonctionnement, et surtout quelques règles de bonnes pratiques pour la portabilité de votre code.

Pour aller plus loin dans Wordpress, l'interaction avec la base de données, l'internationalisation et la compréhension des shortcuts devraient être les prochaines étapes d'apprentissage.

Meteor pour un développement Web rapide

Meteor [1] est un framework javascript open source pour le développement d'applications Web. Son objectif premier est de faciliter et d'accélérer le développement. Nous allons, en quelques étapes, vérifier si Meteor tient cet objectif, et quels sont ses points forts et faibles.

Jorrit Wortel,
Consultant Osaxis (http://www.osaxis.fr)

A cause de l'évolution rapide d'Internet, les applications Web doivent être de plus en plus dynamiques, performantes et distribuables. C'est dans ce contexte très exigeant qu'a été créé Meteor. Derrière ce nom assez spatial se cache un programme qui facilite le développement de sites Web à dominante JavaScript. D'après ses créateurs, Meteor permet de faire en quelques heures ce qui aurait nécessité des semaines de développement. Dans cet article, nous allons voir ce qu'il en est, et analyser comment Meteor répond aux besoins essentiels du Web moderne.

Présentation de Meteor

Meteor est un programme ayant plusieurs fonctions, ce qui le rend difficile à qualifier simplement. Il joue tout d'abord un rôle de serveur Web. En effet, c'est lui qui répond aux requêtes du navigateur et qui sert les pages. Il joue également un rôle de framework puisqu'il fournit des mécanismes de codage qui lui sont propres. Enfin, Meteor propose aussi des lignes de commande à utiliser depuis le terminal. Celles-ci servent principalement à effectuer des actions de gestion du serveur, et de gestion de projet. Meteor est, au jour de l'écriture de cet article, en version 0.8.1. Son numéro de version atteste d'une certaine jeunesse. Pourtant, il a été créé il y a environ trois ans (fin 2011). Son jeune âge lui donne un avantage sur les autres technologies poursuivant le même but, puisqu'il est né avec les contraintes du Web moderne. Concrètement, la partie serveur de Meteor est un nœud asynchrone NodeJS [2] qui répond aux évènements générés par les clients. Ce serveur transmet les données aux différents clients en JavaScript Object Notation (JSON [3]). En ce qui concerne la partie cliente, cette dernière se base sur du HTML, du JavaScript et du CSS. Enfin pour la partie base de données, Meteor propose une base MongoDB. Aucune mise en forme n'est effectuée côté serveur, ce sont les clients qui génèrent le rendu du site. Cela est fait à l'aide de fragments JavaScript qui traitent et mettent en forme les données reçues. Mais trêve de théorie, passons aux choses sérieuses et découvrons comment créer notre première application avec Meteor !

Mon premier site

Sur les distributions Linux, l'installation se fait avec la commande suivante :

```
curl https://install.meteor.com | /bin/sh
```

Une fois cela fait, il suffit d'invoquer une commande Meteor pour créer un projet :

```
meteor create monProjet
```

Cette commande vient de générer la structure basique pour une application fonctionnant avec Meteor. Il ne nous reste plus qu'à lancer le serveur pour propulser votre site (presque) vide. Pour cela, il suffit de lancer la commande suivante depuis le répertoire du projet :

```
meteor
```

Après quelques lignes de log, le serveur nous informe que le site est disponible à l'URL http://localhost:3000.
Si vous êtes du genre à vouloir obtenir un résultat rapidement, vous allez apprécier ce qui suit. En effet, Meteor propose de mettre votre site directement sur le Cloud hébergé par l'équipe qui a conçu Meteor. Il vous suffit de saisir la commande suivante et votre site est lancé :

```
meteor deploy monApplication.meteor.com
```

Bravo, votre application Web est déjà sur le net ! Malheureusement, elle n'est pas très intéressante pour l'instant. La partie suivante vous explique comment l'enrichir en y ajoutant un peu de contenu.

Gestion du contenu

A présent, nous voulons donc ajouter du contenu. Le mécanisme de base de Meteor pour créer du contenu est l'utilisation de fragments (déclarés par le mot clé « template »). Les éléments statiques ainsi que les fragments sont déclarés dans un fichier au format HTML. Tout ce qui concerne la partie dynamique du site Web est déclaré dans un fichier JavaScript. Dans une application Web Meteor, nous avons donc a minima ces deux fichiers plus une feuille de style. Si vous regardez l'application que nous venons de créer, c'est en effet ce que la commande a généré. Nous allons tout d'abord nous concentrer sur les fichiers JavaScript et HTML. Voici le code HTML de notre application :

```
<head>
  <title>AppliMeteor</title>
</head>
<body>
  {{> hello}}
</body>
<template name="hello">
  <h1>Hello World!</h1>
  {{greeting}}
  <input type="button" value="Click" />
</template>
```

Dans ce code, on peut remarquer les deux choses suivantes : les deux chaînes « {{ > hello}} » et « {{greeting}} ». Dans la syntaxe Meteor, « {{ > un_nom}} » permet de définir l'inclusion du contenu d'un fragment. Quant à la même syntaxe sans chevron ({{un_autre_nom}}), elle permet d'exécuter une fonction JavaScript. Si la fonction appelée retourne un contenu, ce dernier est intégré dans la page à l'endroit de l'appel. Dans notre page HTML, nous incluons donc le contenu du fragment « hello » dans la balise <body>. Dans ce fragment, un appel vers la fonction « greeting » est effectué. Observons maintenant son homologue JavaScript :

```
if (Meteor.isClient) {
  Template.hello.greeting = function () {
    return "Welcome to AppliMeteor.";
  };
  Template.hello.events({
    'click input': function () {
      if (typeof console !== 'undefined')
        console.log("You pressed the button");
    }
  });
}
```

La première chose à remarquer est le test effectué au début « if(Meteor.isClient) ». Cette condition n'est pas sans rappeler la célèbre fonction fork du langage C, qui, en fonction de sa valeur de retour, permet de différencier le processus père du processus fils. Ici, c'est l'appel des fonctions « Meteor.isClient » ou « Meteor.isServer » qui permet de savoir si le code est exécuté côté serveur ou côté client. Si l'on regarde la suite, nous retrouvons bien la fonction « greeting » qui est appelée dans le fragment « hello ». Notez que le nom de la fonction est précédé du nom du template qui l'appelle. Nous pouvons donc avoir plusieurs fonctions de même nom, tant qu'elles ne sont pas appelées dans le même fragment. La ligne commençant par « Template.hello.events » nous indique également que certaines actions se font sur événement provenant du fragment « hello ». Il s'agit ici d'un message affiché dans la console dans le cas où un événement « click » est lancé sur un élément de type « input ». Pour voir l'affichage de ce message, il vous faut donc lancer la console de votre navigateur. Au travers de cet exemple, nous avons vu les principaux éléments qui permettent de rendre une page dynamique et modulaire. Malgré cela, il nous manque encore un élément fondamental pour rendre notre application Web complète : les données. Découvrons donc désormais comment Meteor propose de gérer des données.

Base de données

Comme cité précédemment, Meteor met à votre disposition une API MongoDB (base de données de type NoSQL). Sans rentrer dans le détail ni insister sur ses points faibles, ce type de bases est moins contraignant que les bases relationnelles classiques, et tient mieux la charge. **Le numéro 154 de Programmez! vous propose de découvrir ce modèle de bases de données dans l'article « Bases NoSQL et MongoDB »**. Meteor nous réserve une particularité concernant la base MongoDB : l'API est présente aussi bien sur le client que sur le serveur. Mais alors que le côté serveur bénéficie d'un accès direct à la base de données, le côté client doit se contenter d'un cache. Le client « s'abonne » à des ensembles de données et le serveur les alimente dans le cache du client. S'il y a conflit sur la cohérence des données, c'est le serveur qui a le dernier mot. Par défaut, Meteor simule un accès en lecture/écriture côté client en activant les packages « insecure » et « autopublish ». Cette fonctionnalité est bien pratique dans une phase de tests ou de démo mais représente une faille de sécurité en production. En effet, cela signifie que n'importe quel client peut modifier les données et les propager instantanément vers le serveur. Il suffit toutefois d'ôter les packages pour reprendre le contrôle des données. Une fois que le cache du client est renseigné, il reste à utiliser les données disponibles. Pour afficher des données, il faut indiquer à la page où aller chercher les éléments nécessaires à son rendu. Voici un exemple de fragment qui récupère des données dans un ensemble provenant de la base :

```
<template name="color_list">
  <ul>
    {{#each colors}}
      {{> color_info}}
    {{else}}
      No color !
    {{/each}}
  </ul>
</template>
```

Cet extrait de code fait appel au fragment « color_info » pour chaque couleur dans l'ensemble de données « colors ». Si aucune couleur n'est présente, un message le signifiant est affiché. L'affichage des données provenant de la base n'est pas plus compliqué que cela. Parce que l'on est rarement la seule personne sur le Web, une application Web est par essence multi-utilisateurs. Dès lors, la problématique de mise à jour des interfaces devient fondamentale. Si un utilisateur apporte une modification à une donnée, tous les autres utilisateurs doivent la voir de manière quasi-instantanée. Comment mettre en place ce genre de comportement ? Avec Meteor, vous n'avez pas à vous en soucier car son moteur s'en occupe pour vous. La donnée modifiée est transmise depuis le client vers le serveur pour prise en compte dans la base. Le serveur propage ensuite cette donnée aux autres interfaces pour mise à jour dans leur cache. Cette mise à jour se fait grâce à des appels Ajax et ne provoque donc pas le rafraîchissement de la page.

Pour aller plus loin

Voici une application Web exemple que vous pouvez télécharger (http://www.osaxis.fr/download/application_exemple_meteor.zip) pour illustrer les principes expliqués juste avant. Il s'agit d'une application Web moins minimaliste qui permet de parier sur des chevaux. Vous pouvez, depuis la console de développement, ajouter des chevaux dans la base via des commandes MongoDB. Une fois ceux-ci ajoutés, il est possible d'en sélectionner un dans l'interface et lui ajouter ou lui retirer des paris. Il est également intéressant de faire le test en lançant deux (ou plus) navigateurs et d'observer le comportement des interfaces.

Indépendance de Meteor

Un des risques d'utiliser une application aussi jeune que Meteor concerne les doutes sur la pérennité du produit. Pour pallier cela, Meteor propose une fonctionnalité qui permet de récupérer une version de votre application Web indépendante de Meteor. Pour ce faire, il suffit de saisir la commande suivante :

```
Meteor bundle monApplication.tgz
```

Meteor produit alors une archive au format tar.gz qui peut être déployée comme une application NodeJS tout à fait valide. Vous pouvez ensuite remettre cette application en service en la déployant via une commande NodeJS analogue à la suivante :

```
NodeJS localhost:3000 monApplication.tgz
```

Attention cependant, l'archive ainsi produite est compatible uniquement avec la plateforme sur laquelle elle a été générée. L'équipe de Meteor, en attendant un développement corrigeant cela, propose une solution temporaire avec le logiciel NPM. Il faut décompresser le fichier .tgz obtenu précédemment et exécuter les commandes suivantes :

```
cd bundle/programs/server/node_modules
rm -f fibers
npm install fibers@1.0.1
```

Meteor en bref

Pour résumer, le jeune âge de Meteor se fait encore ressentir dans son mécanisme d'indépendance incomplet, et son indisponibilité sur toutes les plateformes. Conséquence directe de cette jeunesse, sa communauté est assez modeste. Malgré cela, c'est un outil qui permet de rapidement et facilement construire et déployer une application Web. On apprécie notamment sa syntaxe simple de définition de fragments, ainsi que sa gestion aisée des données. ⊞

Ressources

[1] Site officiel de Meteor : www.meteor.com
[2] Site officiel de NodeJS : www.nodejs.org
[3] Site officiel de JSON : www.json.org

Trois étapes pour intégrer facilement votre application Windows Phone à Cortana

Jean-Sébastien Dupuy
Evangéliste technique - **DX / Microsoft France**

Bonjour, je suis Cortana

C'était promis depuis son lancement à l'évènement //build/ en avril dernier, Cortana (votre assistante personnelle sous Windows Phone) a pris des cours de français et arrive sur notre vieux continent. Elle est disponible depuis le 5 décembre, en version Alpha, pour la France, l'Espagne, l'Italie et l'Allemagne. Cette mise à jour est principalement adressée aux développeurs d'une part pour participer à l'amélioration du service et nous proposer vos retours et d'autre part en profiter pour intégrer vos applications avec Cortana.

Pour découvrir la voix française de Cortana et bénéficier de ses services, il suffit simplement d'être inscrit au programme Windows Phone Preview pour les développeurs [http://dev.windows.com/fr-fr/develop/devpreview].

Version alpha ?

Pour développer Cortana, nous utilisons des techniques d'apprentissage automatique (machine learning) afin de construire des modèles de reconnaissance vocale et de compréhension de langage naturel. Pour les améliorer, nous avons besoin d'utilisateurs comme vous pour discuter avec Cortana ; qu'elle suive, en quelque sorte, des cours accélérés de français.

Cortana ne s'améliorera que si vous interagissez avec elle. En apprenant à vous connaître au fil du temps, Cortana vous fera des suggestions de plus en plus pertinentes et précises, et vous offrira une expérience personnalisée. Si vous souhaitez en découvrir davantage sur Cortana et ses services, rendez-vous sur http://www.windowsphone.com/fr-fr/how-to/wp8/cortana/meet-cortana.

Votre avis compte, n'hésitez pas à nous transmettre vos commentaires : http://cortana.uservoice.com/.

Quel rapport avec les développeurs ?

Windows Phone propose des commandes vocales. Cette fonctionnalité n'est pas récente, elle a été introduite sous Windows Phone 8.0 et permettait le lancement de votre application par reconnaissance vocale. Même si quelques applications ont joué le jeu, le concept n'a pas forcément été des plus suivis par la communauté, et la majorité des applications n'utilisait simplement pas (ou mal) cette fonctionnalité.

Pour quelles raisons ? Eh bien il faut avouer que pour l'utilisateur, la fonctionnalité manquait un peu de visibilité: si un appui long sur le bouton Windows permettait bien de lancer la reconnaissance vocale, ce n'était clairement pas très intuitif. L'activation de nos commandes vocales était donc un peu difficile à dénicher.

Cortana change la donne, elle prend désormais directement les opérations en main. Et elle ne se cache pas ! Elle est accessible par une vignette dynamique ou directement par le bouton Recherche du téléphone, elle devient donc la nouvelle interface pour accéder aux données de Bing. A noter au passage que Cortana n'est pas une assistante uniquement vocale, vous pouvez la questionner ou interagir avec elle en mode textuel si vous préférez plus d'intimité.

Mais revenons aux applications. Cortana permet d'actionner simplement, et en langage naturel, les principales fonctionnalités du téléphone (passer un appel, prendre une note, créer une alarme, ...); vous pouvez dès aujourd'hui étendre ces scénarios et lancer votre application par le biais de nos fameuses commandes vocales. Dans le principe, les commandes vocales se mettent en place très rapidement en passant par les trois étapes suivantes :

- Création d'un fichier pour définir les commandes vocales
- Enregistrer les commandes vocales au démarrage de l'application
- Gérer l'activation des commandes vocales

Etape 1. Création d'un fichier pour définir les commandes vocales

Ajouter un nouvel élément de type 'Définition des commandes vocales' à votre projet. Que vous développiez en Silverlight 8.1 ou en Windows Runtime 8.1 (Application Universelle), la structure du fichier est identique.

```
<?xml version="1.0" encoding="utf-8"?>
<VoiceCommands xmlns="http://schemas.microsoft.com/voicecommands/1.1">
 <CommandSet xml:lang="en-us">
  <CommandPrefix>tickets</CommandPrefix>
  <Example>show tonight's events at Olympia</Example>

  <Command Name="showEventsAt">
   <Example>show tonight's events at Olympia</Example>
   <ListenFor> [Show] {eventsViews} at {eventsLocations} </ListenFor>
   <Feedback> Showing {eventsViews} at {eventsLocations} </Feedback>
   <Navigate Target="HomePage.xaml"/>
  </Command>

  <Command Name="showEventsFrom">
   <Example>show tonight's events from Shaka Ponk</Example>
   <ListenFor> [Show] {eventsViews} from {eventsArtists} </ListenFor>
   <Feedback> Showing {eventsViews} from {eventsArtists} </Feedback>
   <Navigate Target="HomePage.xaml"/>
  </Command>

  <PhraseList Label="eventsViews">
   <Item> tonight's events </Item>
   <Item> best sellers </Item>
  </PhraseList>

  <PhraseList Label="eventsLocations">
   <Item> Olympia </Item>
   <Item> Bercy </Item>
  </PhraseList>

  <PhraseTopic Label="eventsArtists" Scenario="Natural Language">
   <Subject>Music</Subject>
   <Subject>Artist</Subject>
  </PhraseTopic>
 </CommandSet>

 <!-- Other CommandSets for other languages -->
</VoiceCommands>
```

Pour faire rapide, voici les principaux éléments qui composent la structure du fichier. Si vous préférez la documentation de référence http://msdn.microsoft.com/fr-fr/library/dn630431.aspx.

- CommandSet : liste les commandes vocales pour une langue spécifique. Spécifiez l'attribut xml:lang="fr-fr" pour supporter le français. Il est possible de supporter **15 langues** différentes.
- CommandPrefix : par défaut, vous pouvez lancer l'application directement par son nom. Toutefois, si le nom est trop compliqué ou trop long, vous pouvez lui attribuer un diminutif plus convivial à utiliser.
- Command : définit chaque commande vocale individuelle que l'application supporte. Vous pouvez enrichir l'application avec un maximum de 100 commandes. Ces commandes sont définies elles aussi par les éléments enfants suivants:
  - ListenFor : expression qui va être reconnue par Cortana; elle actionnera l'interaction avec l'application. La syntaxe [...] définit un terme optionnel, alors que la syntaxe {...} pointe sur un label de type PhraseList ou PhraseTopic (nouveauté en 8.1).
  - Feedback : définit comment Cortana doit répondre à l'utilisateur pendant le lancement de l'application.
  - Navigate : précise la page que l'on souhaite afficher au lancement de l'application quand cette commande est invoquée. Ce point est obligatoire en Silverlight, optionnel en Universal.
- PhraseList : une liste de termes pour aider à la compréhension. Un dictionnaire de 2000 éléments peut être renseigné. Comme c'est généralement fastidieux, je vous rassure, ces termes peuvent être dynamiquement ajoutés par code.
- PhraseTopic : c'est la nouveauté de Windows Phone 8.1. La commande vocale peut s'appuyer sur un label plus ouvert qu'un dictionnaire. Vous demandez à Cortana/Bing de reconnaître simplement le mot. Comme la reconnaissance d'un terme est souvent liée à son contexte, je vous invite à préciser l'attribut Scenario et les enfants Subject pour faciliter le travail de reconnaissance et obtenir un résultat plus pertinent.

Pour résumer, mon exemple propose donc deux commandes, **showEventAt** et **showEventFrom**, dont le but est de lister les évènements musicaux par lieu et par artiste.

- Les lieux sont définis par un dictionnaire nommé **eventsLocations**, la reconnaissance vocale va donc choisir une interprétation parmi cette liste de mots/phrases. Cette méthode est la plus fiable en termes de reconnaissance.
- Les artistes (via le label **eventsArtists**), sont quant à eux interprétés directement par le couple Cortana/Bing, sans l'usage d'un dictionnaire. Pour arriver à ces fins, l'analyse s'appuie uniquement sur la reconnaissance vocale de Bing, et sur un contexte proposé par des mots clés additionnels (comme Music ou Artist dans mon cas de figure).

Etape 2. Enregistrer les commandes vocales au démarrage de l'application

L'application doit s'exécuter au moins une fois pour enregistrer les commandes et les rendre disponibles auprès de Cortana. Les quelques lignes suivantes suffisent à enregistrer le fichier vcd.xml et les commandes associées. Projet Windows Phone Silverlight 8.1:

```
private async void RegisterVoiceCommands()
{
  Uri uriVoiceCommands = new Uri("ms-appx:///vcd.xml", UriKind.Absolute));
  await VoiceCommandService.InstallCommandSetsFromFileAsync(uriVoiceCommands);
}
```

Prérequis: n'oubliez pas d'activer les capacités **ID_CAP_SPEECH_RECOGNITION**, **ID_CAP_MICROPHONE**, et **ID_CAP_NETWORKING** dans le fichier WMAppManifest.xml.
Projet Windows Runtime 8.1 (Application Universelle) :

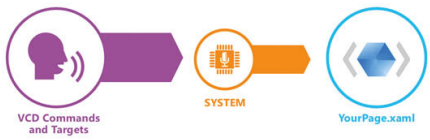
```
private async void RegisterVoiceCommands()
{
  Uri uriVoiceCommands = new Uri("ms-appx:///vcd.xml", UriKind.Absolute);
  StorageFile file = await StorageFile.GetFileFromApplicationUriAsync(uriVoiceCommands);
  await VoiceCommandManager.InstallCommandSetsFromStorageFileAsync(file);
}
```

Prérequis: n'oubliez pas d'activer la capacité **Microphone** dans le fichier Package.appxmanifest.

Etape 3. Gérer l'activation des commandes vocales

Une fois la commande reconnue et interprétée par le système. Cortana se charge de lancer l'application et en profite pour transmettre les informations vocales dictées par l'utilisateur. C'est sur ce point que l'architecture Silverlight vs Universelle est la plus différente.
Projet Windows Phone Silverlight 8.1:

- Le système lance l'application et affiche la page précisée dans l'élément Navigate du fichier de commandes vocales.



- Toutes les infos vocales sont transmises par Query String dans le contexte de navigation. Vous pouvez donc extraire ces infos dans la page et proposer un rendu approprié.

```
protected override void OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
{
 base.OnNavigatedTo(e);

 // Is this a new activation or a resurrection from tombstone?
 if (e.NavigationMode == System.Windows.Navigation.NavigationMode.New)
 {

  // Was the app launched using a voice command?
  if (NavigationContext.QueryString.ContainsKey("voiceCommandName"))
  {

   // If so, get the name of the voice command.
   string voiceCommandName
    = NavigationContext.QueryString["voiceCommandName"];

   // Define app actions for each voice command name.
   switch (voiceCommandName)
   {
    case "showWidgets":
     string viewWidget = NavigationContext.QueryString["widgetViews"];

     // Add code to display specified widgets.
     break;

    // Add cases for other voice commands.
    default:

     // There is no match for the voice command name.
     break;
   }
  }
 }
}
```

Projet Windows Runtime 8.1 (Application Universelle) :

Contrairement à Silverlight, les informations peuvent être interceptées directement dans la classe **App** via l'évènement **OnActivated** avant d'être transmises à la page (d'où le caractère optionnel de l'élément Navigate dans le fichier de commandes vocales). C'est donc à vous de rediriger l'utilisateur vers la page de votre choix!
IActivatedEventArgs permet de vérifier d'une part si l'activation de notre application provient bien de Cortana (args.Kind == ActivationKind.Voice Command) et d'autre part d'extraire toutes les infos vocales dans un bel objet tout propre (plus élégant qu'une Query String tout de même) : SpeechRecognitionResult.

Code App.xaml.cs

```
protected override void OnActivated(IActivatedEventArgs args)
{
  base.OnActivated(args);

  Frame rootFrame = Window.Current.Content as Frame;

  // Was the app activated by a voice command?
  if (args.Kind == ActivationKind.VoiceCommand)
  {
    var commandArgs = args as VoiceCommandActivatedEventArgs;
    SpeechRecognitionResult speechRecognitionResult = commandArgs.Result;

    // If so, get the name of the voice command, the actual text spoken, and the value of Command/Navigate@Target.
    string voiceCommandName = speechRecognitionResult.RulePath[0];
    string textSpoken = speechRecognitionResult.Text;
    string navigationTarget = speechRecognitionResult.SemanticInterpretation.Properties["NavigationTarget"][0];

    switch (voiceCommandName)
    {
      case "showEventsAt":
        rootFrame.Navigate(typeof(HomePage), commandArgs.Result);
        break;

      case "showEventsFrom":
        rootFrame.Navigate(typeof(HomePage), commandArgs.Result);
        break;

      // Cases for other voice commands.
      default:
        // There is no match for the voice command name.
        break;
    }
  }
}
```

Code HomePage.xaml.cs

```
/// <summary>
/// Invoked when this page is about to be displayed in a Frame.
/// </summary>
/// <param name="e">Event data that describes how this page was reached.
/// This parameter is typically used to configure the page.</param>
protected async override void OnNavigatedTo(NavigationEventArgs e)
{
  base.OnNavigatedTo(e);

  ApplicationView.GetForCurrentView().SetDesiredBoundsMode(ApplicationViewBoundsMode.UseCoreWindow);
  RegisterVoiceCommands();

  // Get recognition result from parameter passed in frame.Navigate call
  SpeechRecognitionResult speechRecognitionResult = e.Parameter as SpeechRecognitionResult;

  if (speechRecognitionResult != null)
  {
    string commandMode = speechRecognitionResult.SemanticInterpretation.Properties["commandMode"][0];

    if (commandMode == "voice") // Did the user speak or type the command?
    {
      //SpeakText(audioPlayer, String.Format("MSDN app heard you say {0}", vcResult.Text));
      HandleCommand(speechRecognitionResult);
    }
    else if (commandMode == "text")
    {
      //messageTextBox.Text = string.Format("Working on your request \"{0}\"", vcResult.Text);
      HandleCommand(speechRecognitionResult);
    }
  }

  SongkickService service = new SongkickService();
  ContentResponse response = await service.EventSearch(48.856930, 2.341200);

  lstLocations.ItemsSource = response.resultsPage.results.events;
}

private void HandleCommand(SpeechRecognitionResult speechRecognitionResult)
{
  string voiceCommandName = speechRecognitionResult.RulePath[0];
  string textSpoken = speechRecognitionResult.Text;
  string navigationTarget = speechRecognitionResult.SemanticInterpretation.Properties["NavigationTarget"][0];

  string eventsViews = speechRecognitionResult.SemanticInterpretation.Properties["eventsViews"][0];

  switch (voiceCommandName)
  {
    case "showEventsAt":
      string eventsAt = speechRecognitionResult.SemanticInterpretation.Properties["eventsLocations"][0];
      break;

    case "showEventsFrom":
      string eventsFrom = speechRecognitionResult.SemanticInterpretation.Properties["eventsArtists"][0];
      break;

    // Cases for other voice commands.
    default:
      // There is no match for the voice command name.
      break;
  }
}
```

Et voilà, le tour est joué ! Si vous avez besoin d'approfondir le sujet, je vous invite à consulter l'exemple MSDN Voice Search for Windows Phone 8.

A la découverte de Kinect pour Windows v2

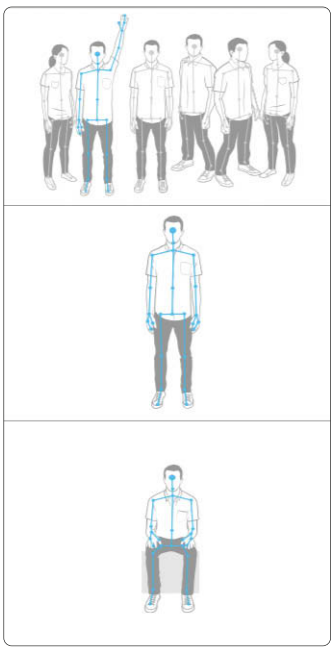
Si la technologie Kinect existe déjà depuis quelques années, une nouvelle étape importante dans son cycle de vie vient d'être franchie il y a quelques semaines avec la sortie officielle de la version 2 du capteur Kinect pour Windows. Au-delà du seul objet physique, ce lancement s'accompagne de la mise à disposition d'une nouvelle version d'un kit de développement aux capacités très largement renforcées comparé à la version antérieure. Découverte et passage en revue de cette nouvelle mouture...

Fabrice BARBIN
MVP Kinect pour Windows
Fondateur de SYNERGIZ (http://www.synergiz.com)

Fig.1

Principe général

D'un point de vue fonctionnel, le capteur Kinect dispose de plusieurs caractéristiques clés. Il incorpore tout d'abord une caméra lui permettant de filmer - avec un champ de vision large et en résolution FullHD - la scène présente devant lui. Une belle évolution par rapport à la version précédente pour laquelle la qualité vidéo était plutôt décevante. Cette capacité de vision en 2 dimensions est complétée par une caméra et des émetteurs infrarouges qui fournissent une information de profondeur. En exploitant la technique du Time-Of-Flight, c'est-à-dire l'analyse du temps de propagation et de rebond des photons, la Kinect est en effet en mesure de déterminer la distance qui la sépare des différents éléments présents dans la scène filmée : un temps de rebond court impliquera un élément proche, là où un temps long impliquera un élément plus éloigné. La finesse de cette analyse est telle que la distance peut être estimée de façon extrêmement précise. Cette approche permet d'analyser de manière fiable une scène sur une plage de distance de 50 cm à 4.5m environ. Au-delà, la finesse décroit, bien que le capteur puisse réellement voir jusqu'à 8m. Il est à noter que ces mêmes capteurs infrarouge permettent à la Kinect de voir dans des conditions de luminosité très contrastées, y compris dans le noir le plus absolu. Les éventuelles sources de lumière parasites sont par ailleurs traitées afin d'annuler tout risque d'éblouissement du capteur. Cette capacité permet d'imaginer des scénarios d'usage fonctionnant autant de jour que de nuit.



En haut, le tracking du corps. mode "full skeleton" (avec 25 points), mode assis.

En complément de ces capacités optiques, le capteur est également doté d'une série de 4 micros, répartis sur sa largeur qui lui permettent d'appréhender les sons ambiants Fig.1.

Mais où réside le principal intérêt du capteur Kinect? C'est dans l'adjonction d'intelligence qui a été apportée à l'ensemble de ces senseurs physiques. La Kinect offre en effet de réelles aptitudes à traiter et à « comprendre » ce qu'elle voit ou ce qu'elle entend.

Cela se traduit par exemple par la capacité à interpréter les signaux reçus pour détecter la présence d'individus au sein de la scène filmée, ou encore à isoler des instructions orales du bruit ambiant afin de les interpréter en mots ou phrases intelligibles.

Des capteurs

Le capteur Kinect V2 existe en deux versions : celui dédié à la console Xbox One, et celui dédié à Windows. De manière générale, les capacités matérielles des deux modèles sont quasiment identiques. Ce qui les différenciait il y a peu de temps encore, était que les deux capteurs n'étaient pas interchangeables. Cette situation a changé dernièrement puisqu'un adaptateur est commercialisé par Microsoft depuis l'automne: il permet dorénavant de brancher le capteur pour Xbox One en USB sur un PC : une ouverture vers les développeurs, heureux propriétaires d'une Xbox, qui pour une cinquantaine d'euros peuvent dorénavant s'essayer au développement sur Kinect. A noter que l'approche inverse (connexion d'un capteur Kinect pour Windows sur Xbox) demeure à ce jour impossible.

Les bases du SDK

Afin de permettre aux développeurs d'exploiter les données captées, un kit de développement « haut niveau » est fourni. Il repose sur l'exploitation de sources de données ou flux, représentations avancées des informations collectées et interprétées par le capteur :

Flux couleur : il correspond aux données captées par la caméra couleur. Rythmé par défaut à 30 images par seconde, ce flux donne accès à des images Full-HD disponibles selon différentes gammes colorimétriques.

- Flux InfraRouge : il correspond aux données captées par la caméra infrarouge. Par défaut il est rythmé à 30 images par seconde, les données sont retournées au sein d'un format réduit de 512 par 424 pixels. Ce flux est particulièrement intéressant pour disposer d'une vision en niveau de gris d'une scène, et conserver pleinement les textures.
- Flux de profondeur : il correspond aux données calculées sur la base de la technique TOF. A chaque pixel de l'image retournée est associée une valeur exprimée en millimètres et représentant la distance entre le capteur et les éléments présents dans la scène filmée. Ce flux est fourni dans la même résolution que le flux infrarouge, ce qui facilite donc leur traitement conjoint et coordonné.

- Flux Body Index : il fournit l'information relative aux individus repérés au sein de la scène filmée. A chaque pixel est associé un entier représentant le numéro d'un utilisateur. Ce flux est la concrétisation directe des capacités de Machine Learning propres à Kinect. Par apprentissage, il a en effet été « enseigné » au capteur de détecter et isoler la présence d'individus par analyse du flux de profondeur. Le modèle est conçu de manière à ce que seules des personnes soient identifiées.
- Flux Body : selon la même logique que le flux précédent, le flux Body consolide les informations relatives au repérage et à la localisation dans l'espace de 25 points singuliers identifiant une personne : main, coude épaule, etc. Ce suivi peut être opéré de manière simultanée sur 6 personnes présentes au sein de la scène filmée. A noter que si certains points ne sont pas visibles (par exemple, si une main est cachée derrière le dos), le système est en mesure de fournir sa position présumée. L'intérêt du dispositif réside dans le fait que ces informations sont fournies « en temps réel ». Il devient dès lors possible de suivre certains points précis du corps pendant un laps de temps donné et donc d'interpréter leurs évolutions comme une grammaire gestuelle : de manière simpliste, un geste de balayage de gauche à droite d'une main est par exemple reconnu d'un point de vue mathématique comme une large translation du point de la main sur l'axe X et d'une faible évolution de ce même point sur les axes Y et Z.

Architecture générale

Imaginez pour offrir une expérience de développement uniformisée, le SDK peut être utilisé sur une large gamme de plateformes : Fig.2.

Une expérience imaginée sur l'une des plateformes peut donc être déclinée à moindre coûts sur d'autres supports.

Zoom sur le squelette et les mains

Comme mentionné précédemment, le flux de données « Body » va permettre d'accéder assez simplement aux données relatives à certains points singuliers. Le code ci-dessous illustre l'initialisation de la Kinect pour pouvoir interagir avec ce flux :

```
this.kinectSensor = KinectSensor.GetDefault();
FrameDescription frameDescription = this.kinectSensor.DepthFrameSource.FrameDescription;
this.bodyFrameReader = this.kinectSensor.BodyFrameSource.OpenReader();
this.bodyFrameReader.FrameArrived += this.Reader_FrameArrived;
this.kinectSensor.Open();
```

A la réception de chaque frame, il suffit ensuite d'aller recherche la ou les jointures souhaitées et de travailler sur leur position dans l'espace.
Au-delà de la position des points singuliers, le flux « Body » fournit également d'autres éléments d'interactions avancés. Il est ainsi possible de suivre l'état des mains de 2 utilisateurs présents au sein de la scène filmée et d'identifier trois positions distinctes : main ouverte, main fermée, lasso Fig.3.
L'accès à ces 3 états se fait de manière très simple. Sur un objet de type « body » retourné par le flux, des propriétés de haut niveau donnent directement le statut :

```
If (body.HandLeftState == HandState.Lasso)
{
 // change the world
}
```

Cette reconnaissance aisée de l'état de la main ouvre des perspectives extrêmement intéressantes en termes d'interactivité, en offrant notamment la possibilité de gérer des glisser/déposer ou le défilement dans des listes d'items.

Zoom sur la partie Expression

Le SDK intègre également la possibilité de suivre différents éléments sur le visage des utilisateurs. Via l'objet « Face », il est par exemple possible de bénéficier de cinq points singuliers (Œil gauche et droit, nez, coins droite et gauche de la bouche). A défaut d'une vraie technologie d'Eye Tracking, le dispositif est utile pour identifier de manière macroscopique l'orientation de la tête de l'utilisateur. En parallèle, et à l'image de la reconnaissance de l'état des mains, il est également possible de reconnaitre certains éléments spécifiques sur le visage : détecter que la bouche de l'utilisateur est ouverte ou fermée, qu'il dispose d'une expression plutôt joyeuse ou neutre, qu'il porte des lunettes ou encore qu'il ouvre et ferme chacun des yeux.

Fig.3

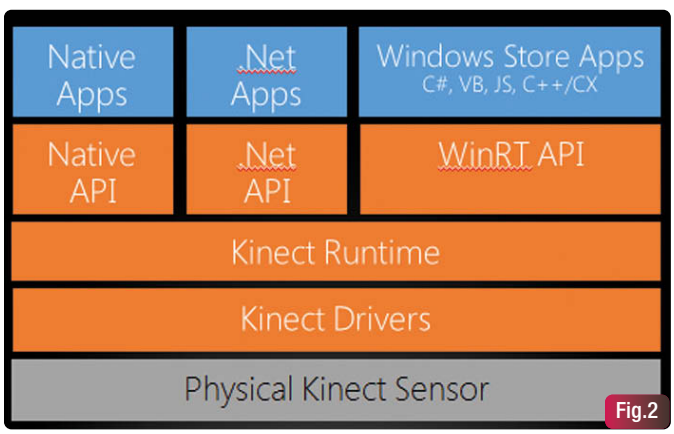


Fig.2

Mais aussi...

Au-delà des capacités mentionnées précédemment, le SDK offre encore une très large richesse fonctionnelle : usage des capacités audio pour permettre le pilotage par la voix, intégration de la reconnaissance gestuelle au sein du moteur Unity3D, intégration des sources de données au sein de plateformes libres telles que OpenFrameWork ou Cinder, extensions des contrôles WPF classiques pour les rendre utilisables par le geste, utilisation du Machine Learning pour optimiser la reconnaissance de gestes, enregistrement de séquences pour les rejouer en playback et valider ses développements.... Autant de capacités facilement accessibles au sein du SDK et permettant d'utiliser le capteur Kinect pour Windows V2 au sein de multiples scénarios d'usage.
Le kit de développement offre un ensemble d'exemples documentés permettant de rapidement prendre connaissance des concepts et de jouer de premiers exemples. Une très bonne aide pour démarrer dans la création d'applications « powered by Kinect for Windows V2 » !

Boostez vos développements Android avec Android Studio 1.0

Plateforme mobile de référence, Android manquait jusqu'à présent d'un éditeur de code intégré dédié pouvant prétendre concurrencer les niveaux de maturité atteints par ses homologues sur des plateformes rivales. Pour combler ce manque, les équipes de Google proposent désormais Android Studio tout fraîchement sorti en version 1.0. Tour du propriétaire.

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

En quelques années, la plateforme Android s'est rapidement imposée comme la solution mobile la plus répandue à travers le Monde. Pour accompagner son développement, les équipes de Google en charge d'Android ont mis à disposition des développeurs un certain nombre d'outils leur permettant de développer le plus efficacement possible des applications afin de venir garnir leur store applicatif, le fameux Google Play Store, car c'est bien là que se gagne la bataille de la mobilité au niveau OS. Pour lier l'ensemble de ces outils et proposer aux développeurs une solution intégrée la plus complète possible, Eclipse avait été choisi comme IDE de référence pour le monde Android pour lequel un plugin spécifique avait été conçu répondant au nom d'Android Development Tools (ADT). Pleinement fonctionnel, ADT aura permis à plusieurs dizaines de milliers de développeurs de concevoir des applications Android aujourd'hui disponibles sur le Google Play Store. Las, avec le temps, les équipes en charge d'ADT ont été limitées par Eclipse, le socle du plugin. En effet, l'IDE leader du monde Java n'est plus tout jeune ni exempt de tous reproches en termes de performances. Impossible dans ces conditions de proposer aux développeurs un outil rivalisant avec le niveau de productivité atteint par les IDE des plateformes rivales tels que Visual Studio ou XCode.
Afin d'adresser cette problématique, Android Studio a été lancé en preview à la conférence Google I/O 2013 avec pour objectif premier de proposer aux développeurs un IDE dédié exclusivement au monde Android améliorant de fait leur productivité. Bâti sur un socle plus léger, Android Studio propose une intégration plus poussée avec la plateforme:

Lancement de l'installation

depuis décembre 2014, il est l'IDE officiel d'Android. Avec cette sortie en version 1.0, les efforts de développement d'ADT sont désormais arrêtés; il est donc temps pour les développeurs Android d'effectuer la bascule vers ce nouvel IDE.

Installation

Basé sur la version community d'IntelliJ IDEA, Android Studio 1.0 est disponible gratuitement en téléchargement à l'adresse suivante : http://developer.android.com/sdk/index.html . Une fois le bundle téléchargé, lancez l'installation (**Fig.1**). Au cours de l'installation, il vous sera proposé de télécharger la dernière version du SDK Android ou bien de renseigner son emplacement en local sur votre ordinateur. L'installation terminée, vous pouvez lancer Android Studio qui vous propose comme premier écran un wizard facilitant votre démarrage avec l'IDE (**Fig.2**).

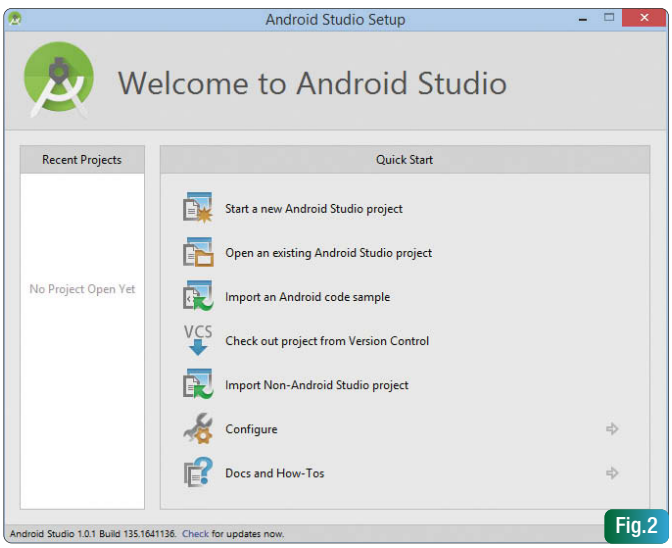


Figure 2 : Wizard de démarrage

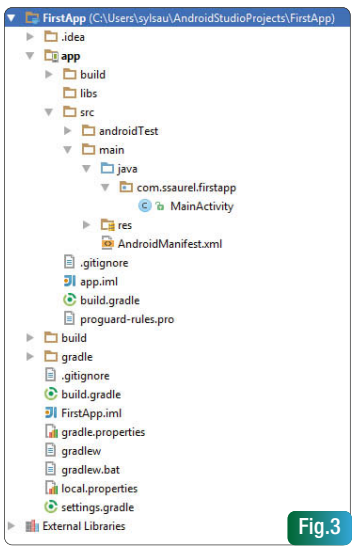


Figure 3 : Structure de projet

Structure de projet

Au sein du wizard de démarrage, cliquez sur "Start a new Android Studio project" afin de créer votre premier projet Android sous Android Studio. Durant les étapes de configuration de ce premier projet, vous pourrez vous familiariser avec les différentes options proposées par l'IDE. Le

projet créé, il est temps de s'arrêter sur la structure d'un projet Android sous Android Studio (Fig.3).
Première différence notable avec Eclipse, sous Android Studio tout se passe dans le contexte d'un projet. Ainsi, au sein de votre IDE seul le contenu du projet en cours sera affiché que ce soit le projet lui-même ou bien les modules dont il dépend par exemple. Il est bon de préciser qu'un menu déroulant permet de filtrer le contenu de cette zone de structure afin d'afficher par exemple uniquement le contenu lié au code de l'application pour gagner en visibilité.
Le répertoire .idea sert à stocker toutes les données de configuration du projet et peut se rapprocher du fichier project.properties sous Eclipse ADT. Au sein du répertoire app, on retrouve les répertoires suivants :

- build qui contient tout ce qui a été généré suite au build du projet. C'est donc ici qu'on trouvera le fichier R.java notamment,
- libs contenant les bibliothèques utilisées au sein du projet sur le même principe que ce l'on retrouve sous ADT,
- src qui va contenir les tests liés au projet sous androidTest, le code source sous main/java, les ressources (drawables, layouts, values, ...) du projet sous main/res ainsi que l'Android Manifest à la racine du répertoire main.

Au final, la structure du projet change de ce que l'on peut connaître sous Eclipse, mais l'on s'y retrouve rapidement et l'organisation permet même de gagner en clarté à terme.

Android Build System

Au sein de la structure de notre projet Android, vous aurez sans doute également remarqué un mystérieux fichier nommé build.gradle se trouvant à la racine du répertoire app. Vous l'aurez compris avec l'extension du fichier, le nouveau système de build Android se base sur Gradle qui est un puissant système de build combinant le meilleur des plus anciens Ant et Maven.
Concrètement, Android Build System consiste en un plugin Android pour Gradle qui offre flexibilité et extensibilité pour la gestion d'un projet Android. En outre, ce nouveau système de build offre l'avantage de pouvoir construire un projet de manière identique à l'intérieur d'Android Studio ou bien à l'extérieur. Il découple ainsi le build d'un projet de l'IDE lui-même ce qui le rend facilement reproductible, ce qui pouvait être un gros défaut avec Eclipse ADT.
Un système de build unifié pour les applications Android est forcément une bonne chose et ce d'autant plus que Google a eu la bonne idée de le doter de quelques unes des fonctionnalités clés suivantes :

- Support des Build Types afin de pouvoir générer différents APK pour différencier des versions debug / release par exemple.
- Support des Product Flavor permettant notamment de générer des APK différents pour une version gratuite et une version pro.
- Possibilité de combiner ces 2 variantes de builds, ce qui dans notre exemple permettra de générer 4 APKs (gratuit debug / gratuit release et pro debug / pro release).
- Gestion des APKs multiples.
- Support multi-dex.
- Gestion des dépendances vers des bibliothèques tierces.

De par la puissance de Gradle et du plugin Android pour Gradle, les possibilités sont nombreuses et vous ne regretterez pas le build lié à Eclipse ADT. Le fichier de build de notre projet a l'allure suivante :

```
apply plugin: 'com.android.application'

android {
  compileSdkVersion 21
  buildToolsVersion "19.1.0"

  defaultConfig {
    applicationId "com.ssaurel.firstapp"
    minSdkVersion 15
    targetSdkVersion 21
    versionCode 1
    versionName "1.0"
  }
  buildTypes {
    release {
      minifyEnabled false
      proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
    }
  }
}

dependencies {
  compile fileTree(dir: 'libs', include: ['*.jar'])
  compile 'com.android.support:appcompat-v7:21.0.3'
                                }
```

Fig.4

Preview rendu écrans multiples

Ici, on remarque l'appel au plugin Android pour Gradle en début de fichier. Gradle utilisant le langage Groovy pour la définition du fichier de build, le plugin Android propose un DSL facilitant la configuration du build. On remarque ainsi la définition du SDK utilisé pour la compilation au sein de la zone Android. Au sein de cette dernière, la propriété applicationId définit le package identifiant notre application. On note la zone buildTypes où l'on souhaite uniquement générer un APK de type release ici sur lequel on appliquera un traitement ProGuard tel que défini dans le fichier de paramètre passé en entrée. Enfin, les dépendances sont définies avec la possibilité de définir des règles ainsi que le numéro de version d'une bibliothèque. Là encore, c'est un énorme progrès par rapport à ce que l'on connaissait sous Eclipse ADT jusqu'ici.

Editeur IHM

Si l'éditeur de code proposé par Android Studio propose une intégration plus poussée avec la plateforme Android que ce soit au niveau de la complétion du code ou du refactoring, le vrai plus de ce nouvel IDE est à chercher du côté de l'éditeur d'IHM. L'éditeur IHM proposé par Android Studio est une source d'amélioration majeure par rapport à ce que proposait Eclipse ADT. En effet, l'éditeur proposé avec ADT était extrêmement lent voir ne fonctionnait plus depuis plusieurs versions d'Android. Bien que le nouvel éditeur IHM soit plus efficace, il est bien entendu que la plupart du temps en tant que développeur vous utiliserez en priorité le fichier XML pour construire le contenu de vos layouts. Néanmoins, les possibilités offertes offrent un gain de temps puisqu'il est possible d'avoir un rendu direct sans avoir à lancer son application au sein d'un émulateur. En outre, la vue de rendu dynamique sur de multiples écrans est un must (Fig.4).

Editeur de traduction

Avec la globalisation d'Android en particulier, et des smartphones en général, internationaliser une application est une absolue nécessité pour les développeurs. Le système de gestion de l'internationalisation mis au point par les équipes Android est puissant mais la saisie des libellés entre différentes langues peut rapidement s'avérer pénible et fastidieuse. Afin d'adresser cette problématique, Android Studio met à disposition un éditeur de traduction permettant de centraliser la gestion des libellés de chaque langue ciblée par votre application au sein d'une seule vue (Fig.5).

Raccourcis

Android Studio étant basé sur IntelliJ IDEA, il en reprend de fait les raccourcis claviers. Les aficionados d'Eclipse ADT devront ainsi revoir leurs raccourcis et mettre à jour leur aide-mémoire pour leurs tâches quotidiennes. Pour faciliter cette migration, voici une liste des raccourcis essentiels à connaître lorsque vous utilisez l'éditeur de code sous Android Studio :

- CTRL + SHIFT + A : autocomplétion,
- ALT + ENTER : correction rapide sur un projet,
- CTRL + ALT + L : reformatage de code,
- CTRL + Q : affichage de la Javadoc sur une API,
- CTRL + P : visualisation des paramètres pour une méthode donnée,
- ALT + Insert : génération de méthodes,
- F4 : aller sur une source,
- CTRL + Y : suppression d'une ligne,
- CTRL + ALT + SHIFT + N : recherche d'un symbole par nom,
- CTRL + D : duplication des lignes de codes sélectionnées.

Une petite perte de productivité les premiers temps qui sera rapidement compensée une fois que vous aurez assimilé les nouveaux raccourcis pour les opérations les plus communes que vous effectuiez sous Eclipse ADT.

Fig.5 *Editeur de traduction*

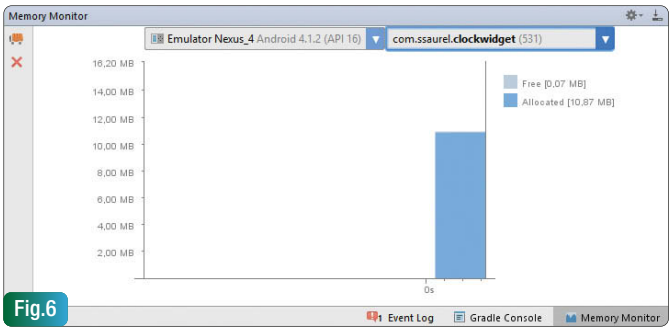


Fig.6 *Monitoring mémoire*

Exécution

L'exécution d'un projet sous Android Studio ne propose pas de réelles différences avec ce que l'on peut trouver sous Eclipse ADT puisque les outils sous-jacents utilisés sont les mêmes. Ainsi, il vous faudra comme de coutume créer un périphérique virtuel ou bien connecter votre terminal physique à votre ordinateur avant de pouvoir lancer l'exécution d'une application depuis Android Studio. Une fois cette étape franchie, il est nécessaire de lancer un "Build and Run" sur votre projet. Soit via les menus soit via le raccourci clavier SHIFT + F10. A ce niveau-là, vous remarquerez sûrement un temps de lancement plus long que ce que vous connaissiez sous Eclipse ADT puisqu'un build Gradle complet du projet est lancé.

Une fois le projet buildé et démarré, il est bien entendu possible d'accéder à une vue de débuggage qui sera opérationnelle lorsque le projet aura été lancé via le menu "Monitor". Les messages de logs sont également bien accessibles grâce à l'intégration de DDMS au sein d'une vue dédiée. Enfin, une vue permettant le monitoring mémoire de l'application en cours d'exécution est également proposée et accessible dans le coin droit en bas de l'IDE (Fig.6).

Migration depuis Eclipse

Si ce n'était déjà fait avant, je suppose que cet article vous aura fait prendre conscience que l'avenir du développement Android passera par Android Studio et qu'il est temps de migrer vos anciens projets Eclipse ADT. Pour ce faire, rien de bien compliqué. L'approche recommandée est d'importer directement votre projet Android au format ADT dans Android Studio. Il faut sélectionner le menu Fichier puis Import et puis choisir le répertoire contenant le projet Eclipse ADT. Un Wizard apparaît alors sous Android Studio pour vous guider dans la procédure d'import (Fig.7). Une fois l'import terminé, un fichier import-summary.txt est disponible afin de valider que tout a bien été importé, et, le cas échéant lister les fichiers non importés. Avec cette approche, Android Studio va générer automatiquement le fichier de build Gradle en y ajoutant les dépendances détectées dans votre projet original. Désormais, vous n'aurez plus à gérer manuellement celles-ci.

Conclusion

Avec Android Studio, Google propose enfin aux développeurs un outil sur mesure pour le développement d'applications Android. La nouvelle structure de projet adoptant le système de build Gradle est également un plus nécessaire pour industrialiser les développements Android. Eclipse ADT aura été une solution de choix depuis la création de la plateforme mais devenait vieillissant et il était temps de passer à une solution plus intégrée. Cet article aura mis en avant les grandes possibilités d'Android Studio tout en sachant que la liste n'est pas exhaustive et que les futures roadmaps promettent également beaucoup.

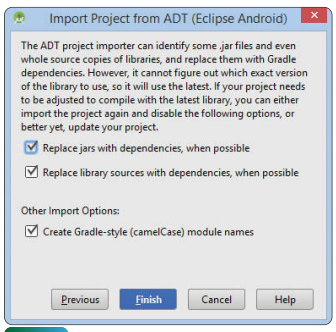


Fig.7 *Import projet Eclipse ADT*

Tests unitaires et intégration continue de projets WordPress

L'objectif de cet article est de montrer comment implémenter l'intégration continue et les tests unitaires sur des projets Wordpress.


Yvain **Giffoni**,
Ingénieur étude et développement - pôle CMS
Netapsys Conseil

1 LES OUTILS POUR LES TESTS UNITAIRES

PHPUnit

Nous allons dans un premier temps nous concentrer sur les tests unitaires. Pour rappel, les tests unitaires visent à s'assurer que chacune des fonctions ou méthodes de votre code PHP répond correctement à la spécification fonctionnelle de celle-ci. Nous allons donc écrire des tests qui pourront être rejoués à chaque fois que le besoin s'en fera sentir. Pour écrire ces tests, il est commode d'utiliser PHPUnit (https://phpunit.de/). Les tests écrits en PHPUnit sont organisés en classes PHP dont les méthodes servent de cas de tests. PHPUnit est, à la base, utilisé en ligne de commande, mais s'intègre très bien dans le cadre de processus automatisés tels que l'intégration continue. Pour ce faire et grâce aux instructions du site officiel, installez PHPUnit sur votre environnement d'intégration continue (qui est peut être celui sur lequel vous codez) via Composer, PEAR, apt-get ou un outil similaire. Une fois installé, la commande « phpunit » va exécuter nos classes de tests et afficher un rapport. Un exemple basique de test avec PHPUnit peut se résumer à cela :
La classe dont on veut tester les méthodes (MaClasse.php) :

```
class MaClasse {

   public function helloWorld() {

      return 'Hello World!';
   }
}
```

La classe de test PHPUnit (MaClasseTest.php) :

```
require_once <le chemin vers votre fichier MaClasse.php>

class MaClasseTest extends PHPUnit_Framework_TestCase {

   private $instance;

   public function setUp() {

      $this->instance = new MaClasse();
   }

   public function test_helloWorld() {

      $this->assertTrue($this->instance->helloWorld() === 'Hello World!');
   }
}
```

Notez le préfixe « test » devant la méthode de notre classe de test, c'est grâce à cela que PHPUnit saura qu'il s'agit d'un cas de test. Une différence doit en effet être faite avec les hooks tel que setUp(). Les hooks disponibles sont les suivants :

```
public function setUp: exécuté avant chaque cas de test
public function tearDown : exécuté après chaque cas de test
public static function setUpBeforeClass : exécuté à l'initialisation de la classe de test
public static function tearDownAfterClass : exécuté à la destruction de la classe de test
```

Oui mais voilà notre projet est en WordPress, on peut donc se demander ce qu'il est intéressant de tester. A l'évidence, écrire des tests pour le noyau du CMS serait une perte de temps puisque celui-ci a déjà été testé. Il en va de même pour tous les plugins officiels qu'on a pu installer. On va donc se tourner vers nos propres plugins et thèmes.
Avant de commencer à tester, une préoccupation demeure : comment faire en sorte que tous les outils WordPress soient bien chargés pour que notre plugin puisse fonctionner en dehors du site le temps du test ?

Wordpress Test

C'est là qu'intervient un petit utilitaire bien pratique - *WordPress Tests* - à récupérer depuis https://github.com/nb/WordPress-tests.
Placez le dossier « wordpress-tests » dans le dossier « plugins » de votre projet WP. Le plugin ne sera pas visible depuis le back-office de votre projet WP, mais servira d'intermédiaire entre PHPUnit et vos plugins lors de l'exécution des tests.
Entre autres, WP Tests va également nous permettre d'avoir une configuration de base de données dédiée aux tests. Ainsi l'exécution des tests se fera sur une base différente de manière à épargner la « vraie » base, qui ne doit pas subir les altérations propres aux tests.

2 CONFIGURATION

Nous allons maintenant devoir rajouter un peu de configuration à PHPUnit pour que ce dernier utilise WP Tests comme intermédiaire. Pour cela il faut rajouter un fichier « phpunit.xml » à la racine de votre projet WP, soit au même niveau que « wp-content » :

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit backupGlobals="false"
      backupStaticAttributes="false"
      colors="true"
      convertErrorsToExceptions="true"
      convertNoticesToExceptions="true"
      convertWarningsToExceptions="true"
      processIsolation="false"
      stopOnFailure="false"
      syntaxCheck="false"
      bootstrap="wp-content/plugins/wordpress-tests/bootstrap.php"
>
   <testsuites>
```

```
    <testsuite name="Plugins">
      <directory>tests-unitaires</directory>
    </testsuite>
  </testsuites>
</phpunit>
```

Le fichier XML que nous venons de créer sera lu par PHPUnit à chaque fois que la commande sera lancée, ainsi l'utilitaire saura où se trouvent les classes de tests (élément <directory> de la section <testsuite>). Consultez la documentation de PHPUnit pour une liste exhaustive des configurations possibles (https://phpunit.de/manual/current/en/index.html).
Nous allons donc modifier la valeur contenue entre les balises <directory> pour indiquer le répertoire qui contiendra nos tests. Pour commencer on pourra par exemple créer un répertoire « tests-unitaires » à la racine du projet WP. Notez que la configuration proposée l'est à titre d'exemple, rien ne vous empêche de mieux organiser vos dossiers et classes de tests pour coller au plus près de vos besoins. On prêtera aussi attention à la valeur de l'attribut « bootstrap » du « phpunit.xml ». Cette valeur sert à indiquer un fichier PHP qui sera chargé avant l'exécution des tests par PHPUnit. Typiquement ce fichier va nous permettre de faire le lien entre PHPUnit et notre utilitaire WP Tests. WP Tests fournit ce fameux fichier de bootstrap, il n'y a donc qu'à mettre le bon chemin dans le phpunit.xml.
Il faut ensuite configurer WP Tests notamment pour lui spécifier la base qui devra être utilisée pour les tests. Dans le dossier de WP Tests se trouve un fichier « unitests-config-sample.php », faites en une copie et renommez-la en supprimant le « -sample ». C'est dans ce nouveau fichier que vous allez renseigner toutes les informations de la base de tests. Il va vous falloir changer les cinq premières constantes :

```
define('ABSPATH', dirname(__FILE__) . '/../../../');

define('DB_NAME', '<nom de la base de test>');
define('DB_USER', '<user pour la base de test>');
define('DB_PASSWORD', '<password pour la base de test>');
define('DB_HOST', '<host pour la base de test>');
```

ABSPATH devrait convenir si vous avez bien installé WP Tests dans le répertoire « plugins » de votre projet WP. En revanche, à vous d'adapter les valeurs suivantes.

3 TESTER UN PLUGIN WP

Une fois nos deux outils liés, nous allons pouvoir nous mettre dans la situation où nous avons un plugin maison à tester. Tout d'abord, prenons comme exemple un plugin qui crée un utilisateur à chaque fois que le hook « custom_action » est déclenché (wp-content/plugins/monplugin/MonPlugin.php) :

```
class MonPlugin {

  public function __construct() {

    add_action('custom_action', array(&$this, 'customAction'));
  }

  public function customAction($user_login) {

    wp_create_user($user_login, '123456');
  }
}
```

Et voilà la classe de test PHPUnit associée (MonPluginTest.php) :

```
require_once dirname(__FILE__) . '/../wp-content/plugins/monplugin/MonPlugin.php';

class MonPluginTest extends WP_UnitTestCase {

  private $plugin;

  public function setUp() {

    parent::setUp();
    $this->plugin = new MonPlugin();
  }

public function test_addAction() {

    $user_login = 'dummy_user';

    do_action('custom_action', $user_login);

    $this->assertNotNull(username_exists($user_login), 'L\'utilisateur "'.$user_login.'"
devrait exister');
  }
}
```

On note ici une clause « extends » différente de notre première classe de test. Ici la classe va dériver de « WP_UnitTestCase » qui elle-même dérive de « PHPUnit_Framework_TestCase ». L'intérêt de « WP_UnitTestCase » est qu'elle va nous préparer le terrain. A titre d'exemple, chaque cas de test sera exécuté au sein d'une nouvelle transaction avec la base de données, ce qui signifie que les altérations faites par le test sur la base de tests seront effacées (rollback) à la fin de celui-ci laissant place nette au test suivant. On retrouve la méthode setUp comme dans notre première classe, à la différence qu'ici, on a rajouté un appel à la méthode parente de WP_UnitTestCase qui se charge de démarrer la transaction. Il est important de toujours appeler la méthode parente à chaque fois que votre classe de test redéfinit un hook (parent::tearDown() dans tearDown() si implémentée etc...).
Vient ensuite notre fameux cas de test, qui va pouvoir utiliser les fonctions WordPress pour s'exécuter.
Nous sommes ici dans un cas de plugin assez simple. Cependant dans le cas d'un plugin nécessitant une installation particulière sous WordPress comme des tables dans la base de données par exemple, il sera nécessaire d'utiliser les hooks « setUpBeforeClass » et « tearDownAfterClass ». En effet, vous devez garder à l'esprit que les tests unitaires ne s'exécutent pas sur la même base que votre projet WP (utilisation de la base de test grâce à WP Tests). Ainsi si vous souhaitez tester un tel plugin, deux choix s'offrent à vous :
- Soit vous vous assurez au préalable que votre base de test contient déjà les tables et données nécessaires au fonctionnement du plugin testé,
- Soit vous réinstallez le plugin en base à chaque exécution des tests et le désinstallez à la fin.

Personnellement je trouve cette deuxième solution plus « propre ». Pour ce faire, il vous suffit d'implémenter le hook « setUpBeforeClass » qui sera dévolu à la création et au remplissage des tables du plugin. Le hook « tearDownAfterClass » se chargera quant à lui de supprimer ces tables et données. Voici un exemple :

```
class MonPluginTest extends WP_UnitTestCase {
```

```
    private $plugin;

    public static function setUpBeforeClass() {

        global $wpdb;

        # Installation du plugin en base (création et remplissage des tables associées)
    }

    public static function tearDownAfterClass() {

        # Désinstaller le plugin, supprimer tout ce qui lui est attaché en base
    }

    # La suite est similaire à l'exemple précédent
}
```

Voilà, vous savez à présent tester un plugin WordPress avec PHPUnit. Il ne vous reste plus qu'à jouer vos tests via la console en saisissant la commande « phpunit » à la racine de votre projet WP.

4 BRANCHER LES TESTS À UN SERVEUR D'INTÉGRATION CONTINUE

Nous allons voir maintenant comment brancher nos tests à un serveur d'intégration continue comme Jenkins.

Avant tout, assurez-vous que votre serveur Jenkins est bien démarré sur votre environnement d'intégration continue, et que les plugins Subversion et Ant sont installés et activés sur celui-ci. Il faut également que l'utilitaire Ant soit installé dans l'environnement, le plugin Jenkins se reposant dessus. Si vous êtes sous une distribution Linux, un simple apt-get ou yum install ant devrait faire l'affaire.

On va tout d'abord créer un fichier « build.xml » à la racine de votre projet WP qui indiquera les étapes que Jenkins devra suivre lors de la construction de votre projet suite à une modification des sources :

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="<mon projet>" default="build" basedir=".">
    <property name="PHPUNIT_REPORT" value="rapport-tests-unitaires.xml"/>

    <target name="build" depends="unit-tests"/>

    <target name="unit-tests" description="Tests unitaires">
      <exec executable="phpunit" failonerror="true">
        <arg value="--log-junit"/>
        <arg value="${PHPUNIT_REPORT}"/>
      </exec>
    </target>

</project>
```

Remplacez la valeur de l'attribut « name » par le nom de votre projet. Le reste du fichier build.xml peut être recopié tel quel. Pour détailler ce qui est fait, la cible « unit-tests » sera exécutée à chaque construction. Cette cible appelle la commande « phpunit » à la racine du projet WP, l'argument « –log-junit » passé à la commande va faire en sorte que PHPUnit enregistre le résultat des tests dans un format XML compréhensible par Jenkins, le fichier sera généré à la racine sous le nom « rapport-tests-unitaires.xml ». L'attribut « failonerror » stipule que si une quelconque erreur survient pendant les tests, la construction ne doit pas être considérée comme valide par Jenkins, signe de régression.

Le fichier « build.xml » prêt, il ne vous reste plus qu'à configurer un nouveau job Jenkins comme vous savez si bien le faire. N'oubliez pas de rajouter une étape de build « Appeler Ant », et une action post-build (« Actions à la suite du build ») « Publier le rapport des résultats de tests JUnit »; entrez « rapport-tests-unitaires.xml » dans le champ (ou la valeur que vous avez choisie dans le build.xml). Cette action post-build va permettre à Jenkins de récupérer les résultats de tests des constructions successives et de les afficher sous forme de graphes sur le tableau de bord du projet.

Maintenant, vous savez désormais comment brancher vos projets WordPress à Jenkins. N'hésitez pas à rajouter une dimension plus qualitative à l'analyse de votre projet avec SonarQube, ou encore, à explorer toutes les possibilités offertes par Jenkins, par exemple, en termes d'automatisation de déploiement.

Créer une application Web offline avec HTML 5

L'arrivée de HTML 5 a profondément fait évoluer le Web. Il est maintenant possible, entre autres, de développer des applications Web fonctionnant hors ligne, c'est-à-dire sans connexion à Internet. Cela permet d'envisager des scénarios où un utilisateur pourra travailler hors-connexion puis synchroniser son travail une fois une connexion retrouvée.

Sébastien **OLLIVIER,**
Développeur Infinite Square

Ce type de scénario convient parfaitement aux applications Web mobiles, à destination de devices de type tablette ou smartphone, où l'utilisateur peut vouloir utiliser une application sans avoir de connexion à Internet, par exemple lorsqu'il est dans les transports en commun, sur un chantier ne disposant pas de connexion, etc.
Pour rendre une application utilisable sans connexion, il faut s'appuyer sur deux briques HTML 5 : le cache et le storage. On va voir dans cet article comment fonctionnent ces briques, et comment on peut en tirer profit pour réaliser une application Web offline.

CACHE HTML 5

La fonctionnalité de cache HTML 5 permet de définir quelles sont les ressources de l'application Web qui doivent être mises en cache par le navigateur.

> *Avant HTML 5, certains navigateurs possédaient déjà un mécanisme de cache pour sauvegarder certains types de ressources, notamment les images, afin d'optimiser le chargement des pages en n'allant récupérer qu'une seule fois ces ressources. Chaque navigateur possédait alors sa propre logique de cache. HTML 5 apporte une spécification permettant de définir coté serveur la stratégie de cache à adopter, quel que soit le navigateur.*

Lorsque le navigateur devra récupérer une ressource ayant été mise en cache, au lieu de la récupérer depuis le serveur, il utilisera la version en cache. De cette manière, même sans connexion Internet, le navigateur pourra afficher une page si les ressources la composant sont en cache.

> La fonctionnalité de cache HTML 5 est implémentée s*ur tous les navigateurs récents et à partir de la version 10 d'Internet* Explorer.

Mettre en cache des ressources

La mise en cache de ressources (pages HTML, fichiers CSS, scripts JavaScript, polices, images, etc.) d'une application Web se fait en créant un fichier manifest. Ce fichier a pour rôle de décrire les règles de mise en cache afin de définir quelles ressources doivent être mises en cache et quelles ressources doivent être systématiquement récupérées du serveur.
Le fichier manifest contient trois sections. La section CACHE liste toutes les ressources devant être mises en cache :

```
CACHE:
index.html
cache.html
app.js
app.css
```

La section CACHE précédente définit que les fichiers index.html, cache.html, app.js et app.css doivent être mis en cache. La section FALLBACK permet de définir quelle ressource doit être récupérée du cache lorsque le navigateur n'arrive pas à récupérer une ressource du serveur.

```
FALLBACK:
details.html cache.html
```

La section FALLBACK précédente définit que le fichier cache.html sera chargé si la récupération du fichier details.html échoue, peu importe la cause de l'échec. La section NETWORK contient la liste des ressources qui ne doivent pas être mises en cache et qui nécessitent d'être systématiquement récupérées du serveur.

```
NETWORK:
payment.html
```

La section NETWORK précédente définit que le fichier payment.html doit être systématiquement récupéré du serveur.
Dans cette section, il est possible d'utiliser le symbole * pour définir toutes les ressources n'ayant pas été listées dans la section CACHE.

```
NETWORK:
*
```

Il est fréquent de rajouter à ces trois sections un entête permettant de définir quelques informations supplémentaires, comme un titre, une date de création ou une version.

```
CACHE MANIFEST
# v1 25/08/2014
# Règles de mise en cache du site
```

L'entête précédent contient la version du manifest, sa date de création ainsi qu'un commentaire. Toutes ces informations sont ignorées par les navigateurs. Toutes les lignes du manifest commençant par un # sont considérées comme des commentaires. Voici un exemple de fichier manifest complet :

```
CACHE MANIFEST
# v1 25/08/2014
# Règles de mise en cache du site

CACHE:
index.html
cache.html
app.js
app.css

FALLBACK:
details.html cache.html

NETWORK:
*
```

Une fois créé, pour qu'il soit pris en compte par les navigateurs, le manifest doit être référencé depuis l'attribut manifest de la balise html :

```
<html manifest="cache.manifest">
…
</html>
```

Le fichier manifest doit être retourné avec le type MIME text/cache-manifest.

> *Il est important de ne pas déclarer le fichier manifest en tant que ressource à mettre en cache. Si c'est le cas, le navigateur ne pourra plus expirer son cache et récupérer les nouvelles versions des ressources, à moins que l'utilisateur ne vide manuellement le cache du navigateur.*

Comment fonctionne le cache ?

Première mise en cache d'une application Web

Lorsque le navigateur va accéder pour la première fois à une application Web, il va télécharger la page HTML ainsi que ses ressources liées de manière classique.
Si la page possède l'attribut manifest sur sa balise HTML, le navigateur va télécharger le manifest à l'adresse renseignée.
Une fois ce fichier téléchargé, le navigateur va effectuer une requête par ressource listée dans la section CACHE afin de récupérer tous les éléments à mettre en cache.
Lorsque tous ces éléments auront été correctement récupérés, le navigateur les mettra en cache, avec le fichier manifest associé. Si au moins un des éléments n'a pas pu être récupéré, la mise en cache sera annulée et sera retentée au prochain chargement de la page.

Accès à une application Web mise en cache

Lorsque le navigateur va accéder à une application Web déjà mise en cache, les ressources définies dans la section CACHE du manifest ne seront pas téléchargées du serveur mais récupérées du cache. Les autres ressources seront téléchargées normalement.
Le navigateur va ensuite comparer le fichier manifest, lié aux ressources du cache, au fichier manifest renvoyé par le serveur. Si ce fichier a été modifié, ne serait-ce que d'un caractère, le navigateur commencera une mise à jour du cache.
De manière identique au premier accès à l'application Web, le navigateur va télécharger toutes les ressources spécifiées dans la section CACHE dans le fichier manifest.
Une fois toutes les ressources récupérées, l'ancien cache sera invalidé et les ressources téléchargées composeront le nouveau cache. Au prochain chargement de la page, la nouvelle version du site sera accessible.

Interagir avec la mise en cache

Il est possible d'interagir avec le mécanisme de mise en cache en utilisant l'objet JavaScript applicationCache.
Cet objet expose une propriété status représentant le statut courant de la mise en cache et plusieurs évènements onchecking, onerror, onnoupdate, ondownloading, onprogress, onupdateready, oncached et onobsolete permettant de s'abonner aux changements de statut.
Cet objet expose en plus les méthodes update permettant de forcer le processus de mise en cache et swapCache permettant de changer l'ancien cache par le nouveau.
Cette API JavaScript permet par exemple d'afficher une barre de progression indiquant l'avancement de la mise en cache puis de proposer à l'utilisateur de rafraîchir son navigateur pour profiter de la nouvelle version.

STORAGE

Une fois que l'application est accessible hors-connexion, il reste à sauvegarder localement les données à afficher dans le navigateur, pour que l'utilisateur puisse les consulter. Pour cela, il faut s'appuyer sur la brique storage de HTML 5. Ainsi, lorsque l'utilisateur accédera à l'application Web sans réseau, elle sera affichée depuis le cache HTML 5 et les données de la page seront récupérées depuis le storage HTML 5. Pour donner la possibilité à l'utilisateur de travailler hors-connexion, et de ne pas uniquement consulter les données, il est possible de sauvegarder les actions effectuées par l'utilisateur dans le storage HTML 5 puis de les synchroniser une fois une connexion à Internet retrouvée.
HTML5 propose trois manières différentes de stocker des données dans le navigateur.

WebSQL

WebSQL est une base de données relationnelle intégrée au navigateur. Il est possible d'y créer des tables, des relations entre tables, d'effectuer des requêtes, etc. de manière similaire à une base de données classique.
Chaque base de données est contextualisée par domaine et par navigateur, c'est-à-dire qu'un site ne pourra accéder qu'à sa base de données et pas aux bases de données des autres sites.

> *Les spécifications HTML 5 ont défini la fonctionnalité WebSQL comme obsolète. Cependant, cette fonctionnalité a déjà été implémentée par la majorité des navigateurs récents, elle reste donc accessible et utilisable. Mais la spécification n'évoluera plus et son support risque de décroitre donc il est conseillé de de ne pas l'utiliser pour de nouveaux développements.*

Web Storage

Le second type de stockage est un stockage clef / valeur, où la clef et la valeur sont des chaînes de caractères.
Ce type de stockage est généralement utilisé pour stocker des données simples, comme les préférences d'un utilisateur, une liste d'éléments à afficher ou des données de configuration.
Chaque navigateur impose une limitation de taille, allant de 5 Mo à 10 Mo. Au-delà de cette taille, le navigateur demandera à l'utilisateur s'il souhaite étendre cette limite.
Les données stockées dans cet espace de stockage ne sont pas cryptées et sont accessibles directement depuis les outils de développements de navigateur. Il est donc indispensable de ne pas y stocker des données sensibles. Ce type de stockage est présent sous deux modes différents : le LocalStorage et le SessionStorage.

LocalStorage

Le LocalStorage est un stockage persistant, il n'est pas effacé à la fermeture du navigateur ou de l'onglet. Les données stockées sont contextualisées par domaine, c'est-à-dire qu'un site ne pourra accéder qu'à ses données et pas à celles des autres sites. L'interaction avec ce stockage se fait via l'objet JavaScript localStorage. Le code suivant sauvegarde les préférences d'un utilisateur dans le LocalStorage.

```
var userPreferences = {
   theme: 'black',
   language: 'fr-FR'
};

var serializedPreferences = JSON.stringify(userPreferences);
localStorage.setItem("userPreferences", serializedPreferences);
```

L'objet userPreferences contient les préférences d'un utilisateur. L'appel à la méthode JSON.stringify permet de sérialiser les préférences sous forme d'une chaîne de caractères. Cette chaîne de caractères est ensuite sauvegardée dans le localStorage via la méthode setItem, avec la clef "userPreferences". Le code suivant récupère les préférences d'un utilisateur depuis le LocalStorage.

```
var serializedPreferences = localStorage.getItem("userPreferences");

var userPreferences = JSON.parse(serializedPreferences);
```

La méthode getItem est utilisée pour récupérer la donnée stockée via la clef "userPreferences". La chaîne de caractères récupérée est dé-sérialisée sous la forme d'un objet JavaScript via la méthode JSON.parse.

SessionStorage

Le SessionStorage, contrairement au LocalStorage, est un stockage éphémère. Les données de ce stockage sont contextualisées par onglet ou fenêtre du navigateur et par domaine, et non plus uniquement par domaine. Lorsque l'onglet ou la fenêtre seront fermés, les données du SessionStorage associées seront supprimées. L'interaction avec ce stockage se fait via l'objet JavaScript sessionStorage, de la même manière que vu pour le LocalStorage.

IndexedDB

Le troisième type de stockage, IndexedDB, est une base de données NoSQL. Il permet de stocker des objets indexés par une clef.
IndexedDB autorise à stocker des données plus lourdes que le Web Storage, puisque la limitation de taille imposée par les navigateurs est généralement de 50Mo.

> *La spécification HTML 5 concernant IndexedDB est en cours d'écriture. Cela implique que des modifications y seront apportées. Cependant, son support est aujourd'hui disponible sur toutes les dernières versions des navigateurs.*

L'utilisation d'IndexedDB se fait via l'objet JavaScript indexedDB. Le code suivant permet d'ouvrir une base de données.

```
var openRequest = indexedDB.open("maDB");
```

La méthode open permet d'ouvrir une base de données en spécifiant un nom de base de données. IndexedDB étant construit sur des mécanismes asynchrones, la méthode open renvoie un objet exposant des callbacks de succès et d'erreurs. Le code suivant permet de créer un ObjectStore, permettant d'y stocker des objets indexés.

```
var store = db.createObjectStore("todo", {keyPath: "timeStamp"});
```

L'appel précédent à la méthode createObjectStore crée un ObjectStore nommé todo. Le second paramètre indique que l'indexation se fera sur le champ timeStamp des objets stockés.
Le code suivant permet de stocker un nouvel objet dans l'ObjectStore créé précédemment :

```
var trans = db.transaction(["todo"], "readwrite");
var store = trans.objectStore("todo");
var request = store.put({
   "text": "Ma nouvelle tâche à faire",
   "timeStamp" : new Date().getTime()
});

request.onsuccess = function(e) {
};

request.onerror = function(e) {
};
```

L'interaction avec un ObjectStore se fait via des transactions. L'objet store de l'exemple précédent correspond à une transaction en mode lecture / écriture sur l'ObjectStore todo. La méthode "put" de cet objet permet l'ajout d'un nouvel objet.
De manière identique à l'ouverture d'une base, l'ajout d'un élément se fait de manière asynchrone, et les callback onsuccess et onerror permettent respectivement de réagir en cas de succès ou d'erreur.
Le code suivant permet de récupérer tous les éléments d'un ObjectStore :

```
var trans = db.transaction(["todo"]);
var store = trans.objectStore("todo");

var cursorRequest = store.openCursor();
```

La méthode openCursor permet d'ouvrir un curseur sur un intervalle de clef.

CONCLUSION

Comme vu précédemment, le cache et le storage HTML 5 permettent d'envisager de nouveaux scénarios d'applications Web, où l'utilisateur pourra travailler sans connexion et pourra ensuite synchroniser son travail. Pour avoir plus d'informations concernant les briques HTML 5 évoquées dans cet article, vous pouvez vous rendre sur le site de W3C et accéder aux différentes spécifications :
http://www.w3.org/TR/webdatabase/
http://www.w3.org/TR/IndexedDB/
http://www.w3.org/TR/webstorage/
http://www.w3.org/TR/2011/WD-html5-20110525/offline.html

Programmation fonctionnelle en JS : (Ré)Inventons ensemble la Monade !

2e partie

Sebastien Nichele
SQLi

Dans une première partie, nous avions passé en revue quelques principes 'fondamentaux' de la programmation fonctionnelle :

- N'utiliser que des fonctions **pures, sans effets de bord,**
- **Écrire ses programmes** à partir de fonctions et, si possible, **uniquement de fonctions**,
- Les fonctions peuvent prendre **en paramètres et renvoyer d'autres fonctions**,
- 'Fonction' est un type de donnée comme les autres (int, string...),
- Ne **pas utiliser de variables** et réaffecter des valeurs, utiliser **uniquement des constantes.**

Appliquer en permanence ces principes n'est malheureusement pas toujours simple.

Comme souvent en programmation, le recours à des **designs patterns** devient vite **indispensable** pour ne pas réinventer en permanence la roue, ou produire du code aux qualités discutables. De même, on voit dans chaque langage se développer des 'tournures' de code que l'on qualifiera d'*idiomatiques*. Celles-ci synthétisent un savoir-faire et une exploitation élégante d'une syntaxe et des axiomes portés par un paradigme - le fonctionnel, dans notre cas - et sont révélatrices d'une certaine expérience et pratique de la part du développeur qui les mets en œuvre. Des "designs patterns" fonctionnels et des tournures idiomatiques, nous en avons déjà découverts quelques-un(e)s ; souvenez-vous :

- **La récursivité** : version fonctionnelle de la boucle for, servant à 'parcourir' un ensemble de valeurs

```
iterationSuivante(pageInitiale);

function iterationSuivante(page){
 effectueCollageSur(cahier[page]);

 if( (page+1) < cahier.length){
  iterationSuivante(page +1);
 }
}
```

- **La curryfication** : une fonction prenant *n* paramètres peut être *curryfiée* et être ensuite invoquée via *n* appels chaînés à un seul paramètre, chacune prenant le paramètre suivant de la liste initiale :

```
// Prenons une fonction à 3 paramètres
function foo(a, b, c) { ... }
// Pour l'utiliser, je dois fournir les trois paramètres lors de l'appel
foo(1, 2, 3);

// Je peux 'curryfier' la fonction
var fooCurry = _.curry(foo);
// _.curry => fonction fournie par la librairie 'lodash'
// Pour ensuite pouvoir l'appeler d'une façon un peu différente
fooCurry(1)(2)(3);

// Chaque appel intermédiaire renvoie une fonction qui
// prend le paramètre suivant ;
// au 3ème appel, j'ai bien le résultat initialement attendu

// En programmation fonctionnelle, ces appels sont donc équivalents :
foo(1, 2, 3) ≡ fooCurry(1)(2)(3) ≡ _.curry(foo)(1)(2)(3)
```

SAIN(T)S RAPPELS JAVASCRIPT

Avant de trancher dans le vif, rappelons quelques caractéristiques de notre langage préféré (en mode strict, évidemment):

- *On peut déclarer des fonctions dans des fonctions,*
- *Portée des variables* : la fonction englobante la plus proche ; les blocs if / for ... sont totalement poreux, ceci inclut la variable de bouclage par exemple.
- *Hoisting* : toute variable déclarée est considérée comme déclarée au début de la fonction avec la valeur 'undefined' ; l'assignation de valeur, elle, à bien lieu au moment spécifié.
- *Hoisting de fonction* : toute fonction déclarée avec un nom est automatiquement considérée comme déclarée au début de la fonction englobante et donc accessible dès le début, même si déclarée à la fin.
- *Fonctions d'ordre supérieur* : une fonction peut recevoir une fonction en paramètre et l'exécuter ultérieurement au moment opportun ; une fonction peut aussi créer une fonction et la renvoyer à l'appelant en guise de retour.
- *Closure* : une fonction qui est renvoyée avec un return capture les variables de son environnement d'exécution avec elle : elle peut alors y accéder ultérieurement, même longtemps après l'appel original ayant généré cette fonction (c'est l'astuce utilisée pour gérer des variables privées en JS).
- *Null et undefined* : en JS, null est utilisé par le programmeur pour dénoter une absence de valeur suite à un calcul - va de pair avec le mot clé return. Undefined signifie que la fonction appelée ne renvoie pas de valeur - absence de return, équivalent de 'void' en java par exemple. Les deux cas doivent théoriquement être traités dans le code, en l'absence de typage et de compilateur susceptible de vous avertir d'un stockage de valeur forcément void...

Consulter le fichier rappelJS.js pour quelques illustrations simples.
Prêtez bien attention à la notion de closure, que nous allons exploiter par la suite.

On peut, sans prendre trop de risques, classer ces deux 'patterns' / 'tournures' comme des basiques de la PF (Programmation Fonctionnelle). Malheureusement, ils ne vont pas suffire. Currifycation, récursivité et fonctions d'ordre supérieur permettent de résoudre de nombreuses problématiques de façon élégante, certes ; mais il faudra, à un moment, faire appel à quelque chose de plus puissant, un 'pattern' dont le nom semble tiré d'un livre de biologie, auréolé de mystère et ayant fait couler beaucoup d'encre...
Je ne vais pas essayer de vous l'expliquer, ou faire de jolis dessins. Je vais plutôt essayer de me retrouver dans une situation de coding périlleuse, pour le 'réinventer' de toutes pièces.

Prêt à attaquer ? Je ne saurais trop vous conseiller de vous rafraîchir la mémoire sur quelques subtilités du JS dans l'annexe 'sain(t)s rappels Javascript' avant de commencer.
Au fil de l'eau, je m'appuierai sur une librairie JS, lodash, entrevue ci-dessus avec 'curry', qui facilite grandement la programmation fonctionnelle sans chercher à 'révolutionner' la syntaxe du JS.

Null, ordre supérieur et contexte d'exécution sont dans un bateau

Partons d'un innocent cas d'utilisation et d'un bout de code procédural. Essayons de le transformer pour en faire quelque chose d'idiomatiquement fonctionnel et élégant.

Un use case 'null'

*Je dois créer une fonction qui utilise une autre fonction pouvant me renvoyer **null**, et je dois utiliser la **valeur de retour** pour **effectuer un traitement**, uniquement si la **valeur est non null**. Ma fonction englobante ne renvoie rien.*
Mise en situation concrète :

```
function genererRapportSiNecessaire() {
  var donneesDuJour = recupererDonneesDuJourSiExisteSinonNull();

  if ( donneesDuJour !== null) {
    imprimeRapport(donneesDuJour); // Implémentation omise car
                  // sans intéret pour le propos
  }
}
```

Court, simple ; mais discutable !
Tout d'abord, genererRapportSiNecessaire est une fonction générant un effet de bord (création d'un fichier, écriture dans une base de données, peu importe) et valant 'undefined'.
En programmation fonctionnelle, on n'aime pas trop ça.
Mais soit ! On la fera évoluer par la suite pour qu'elle renvoie quelque chose. Plus grave dans l'immédiat : **une variable**, et **un if**. Plusieurs problèmes avec cette variable :

- C'est une variable : on devrait pouvoir s'en passer !
- Elle est copiée-collée trois fois (oui oui !)

Plusieurs problèmes avec ce if :

- Il n'est pas 'scalable' (complexité cyclomatique qui explose très vite si on en abuse) ,
- Il est assez faible sémantiquement ; *si vrai, je fais le traitement dans le corps du if* => je dois lire le corps du if pour savoir ce qui va se passer (ici, c'est simple et rapide, mais ce n'est pas toujours le cas ...),
- Il incite souvent à des tests logiques à la complexité de lecture bien trop élevée à mon goût,
- Le corps du if n'est pas isolé du reste du contexte de la fonction en JS (effets de bords possibles...).

On aimerait tant composer imprimeRapport et recupererDonneesDuJourSiExisteSinonNull !

```
imprimeRapport(recupererDonneesDuJourSiExisteSinonNull())
```

Oui, mais quid du if ? Le corps de la fonction genererRapportSiNecessaire, le contexte d'exécution de la variable donneesDuJour, semble tout entier être dédié à gérer cette composition conditionnelle de fonction et **la verticalité du procédural, succession d'étapes 'de haut en bas'**, est ici bien présente et visible. Essayons de progressivement arranger tout ça, comme pour le for de l'article précédent.

De la force d'un code légèrement plus sémantique
Commençons par simplifier l'expression booléenne (avec lodash) :

```
function genererRapportSiNecessaire() {
  var donneesDuJour = recupererDonneesDuJourSiExisteSinonNull();

  if ( ! _.isNull(donneesDuJour) ) {
    imprimeRapport(donneesDuJour);
  }
}
```

Bof, la lisibilité est très moyenne, dommage que lodash ne propose pas isNotNull ... Qu'à cela ne tienne, créons la fonction !

```
function genererRapportSiNecessaire() {
  var donneesDuJour = recupererDonneesDuJourSiExisteSinonNull();

  if ( isNotNull(donneesDuJour) ) {
    imprimeRapport(donneesDuJour);
  }
}

function isNotNull(x){
  return ! _.isNull(x);
}
```

Gain : une petite abstraction et une **meilleure lisibilité**.
Ce n'est pas grand-chose, mais c'est une 'transcription' de moins à effectuer par le cerveau à la lecture du code. **Plus de temps de cerveau disponible pour détecter d'autres abstractions** !

PF = Fonction, fonction, et fonction

Et justement - grand jeu de la programmation fonctionnelle, trouver des structures récurrentes pour créer des abstractions - en avons-nous une ici ?

```
var x ...

if(isNotNull(x)){
  callFunction(x);
}
```

Si une valeur est définie, j'appelle une fonction avec, sinon, je ne fais rien, ou *Si j'ai une valeur, je lui applique une fonction, si j'ai zéro valeur, je n'applique rien*.
Observez bien cette structure ; repérez spatialement le x... le if... ne pourrait-on pas améliorer ça ?

```
//Avant
if (isNotNull(x)) {
  callFunction(x);
}

//Après
ifIsNotNull(x, callFunction); //Implémentation omise pour l'instant
```

Ha HA ! Voilà une abstraction particulièrement élégante ! En exploitant la possibilité de passer des fonctions en argument, nous venons de :

- Faire porter la notion de if par une fonction (souvenez-vous, tout doit être fonction... ou tout du moins, le plus possible !),

- Créer une fonction réutilisable et paramétrable (strategy, command ? => /dev/null),

Eviter un copier-coller de la variable x à plusieurs endroits du code.
Si je décline cette idée en d'autres exemples divers :

```
executeFonctionSur(x, uneFonction);
executeFonctionSiEntier(x, uneFonction);
executeFonction3Fois(x, uneFonction);
```

Et même pourquoi pas :

```
aiguillageBoolean(x, fonctionSiVrai, fonctionSiFaux); // remplace un if / else
```

Appliquons ce principe à notre exemple de génération de rapport :

```
// Avant
function genererRapportSiNecessaire() {
 var donneesDuJour = recupererDonneesDuJourSiExisteSinonNull();

 if ( isNotNull(donneesDuJour) ) {
  imprimeRapport(donneesDuJour);
 }
}

// Après
function genererRapportSiNecessaire() {
 var donneesDuJour = recupererDonneesDuJourSiExisteSinonNull();
 ifIsNotNull (donneesDuJour, imprimeRapport);
}

// Voir même !
function genererRapportSiNecessaire() {
 ifIsNotNull (recupererDonneesDuJourSiExisteSinonNull(), imprimeRapport);
}
```

Je ne sais pas pour vous, mais je suis plutôt emballé par cette tournure ! C'est la complexité cyclomatique qui va être contente.
J'ai diminué le 'bruit' en **supprimant une variable qui s'est finalement révélée inutile,** et le code se lit presque comme une spécification fonctionnelle. En sus, j'ai "horizontalisé" mon code, qui semble bien plus fonctionnel qu'avant.

La fonction comme contexte d'exécution paramétrable et sémantique

En supprimant le bloc de code piloté par le if, et en faisant porter la notion de condition par la fonction, nous venons de **créer un contexte d'évaluation de code isolé, paramétrable et sémantique** qui :

- Embarque notre variable - passée en premier argument,
- Embarque une opération - passée en second argument,;
- Applique l'opération à la variable sous certaines conditions - ici, si la variable n'est pas nulle.

© Chris Williams

En programmation impérative, les opérateurs de contrôle de flux et les blocs de codes servent à créer des *contextes temporaires locaux de calcul*. La programmation fonctionnelle nous incite à faire porter le rôle de ces 'blocs de code' par les fonctions elles-mêmes, favorisant la création d'abstractions très fines dotées d'une sémantique plus précise, au service de l'horizontalité du code.

Les blocs 'if' ne permettent pas de raisonner 'localement' : on pense sans cesse au voisinage, aux variables proches à partir desquelles on va travailler dans le if... à chaque imbrication de if / else, vous faites un nœud au cerveau du développeur qui devra maintenir le code dans le futur (peut-être vous !). Il faut sans cesse relire le bloc, relire le code avoisinant, reconstruire mentalement les étapes de l'algorithme 'vertical' et s'en souvenir durant toute la douloureuse séance de déboggage. Le **contexte d'execution du code** s'allonge **verticalement**, **empilant les mini contextes les uns sur les autres**.
Les fonctions sont autant de *mini programmes indépendants*, plus facilement **composables** et testables unitairement ; autant de **contextes d'exécution s'enchainant horizontalement**.

Indispensable if

Avant de continuer, parlons de l'implémentation de la fonction ifIsNotNull :

```
function ifIsNotNull(x,doSomething){
 if (isNotNull(x)) {
  doSomething(x);
 }
}
```

Comme vous pouvez le voir, pas de magie là-dedans : notre if est toujours bien présent, contrairement au for que nous avions réussi à faire disparaître. Mais il est masqué dans une petite fonction utilitaire, sémantiquement riche et réutilisable, capable de réduire plusieurs lignes de codes en une seule. Une telle fonction utilitaire se doit de rester très courte pour favoriser la compréhension locale et minimiser la surface du contexte d'exécution; et l'on doit autant que possible construire ses programmes en s'appuyant sur et en composant de telles fonctions.

Fin de la 2e partie. Suite et fin de notre aventure JavaScript dans Programmez! 183.

PROJET R&D RASBERRY PI

2e partie

Rentrons maintenant dans le vif du sujet.

Patrick Guillerm
Architecte Java JEE chez **Palo IT**

Java > L'API

L'API Java Pi4J permet d'interagir avec le PiFace. Cette librairie est disponible sur le dépôt Maven :

- groupId : com.pi4j
- artifactId : pi4j-device
- version : 0.0.5.

Le code source est disponible sur GitHub (https://github.com/Pi4J/pi4j/). Cette librairie ressemble beaucoup à celle disponible en Python. On y retrouve les mêmes fonctionnalités avec le coté plus structuré de Java. Il existe cependant quelques subtilités qui diffèrent par rapport à l'API Python. Pour instancier le connecteur PiFace, il est nécessaire d'initialiser deux objets : PiFace et GpioController. L'objet PiFace correspond au module PiFace et GpioController permet de remonter les informations du connecteur entre le PiFace et le Raspberry. Bien que l'objet GpioController ne soit pas forcément utilisé, il est utile qu'il soit initialisé pour le module PiFace :

```
import com.pi4j.device.piface.PiFace;
import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.wiringpi.Spi;

public class FoobarExample {

  public static void main(String[] args) throws Exception{

    piface = new PiFaceDevice(PiFace.DEFAULT_ADDRESS, Spi.CHANNEL_0);
    gpio = GpioFactory.getInstance();

  }
}
```

L'API Java nous fournit les variables par défaut pour s'interfacer avec le PiFace. Dans le cas de l'utilisation, il suffit simplement de remplacer l'adresse du PiFace et son canal pour se connecter sur le bon module :

```
import com.pi4j.device.piface.PiFace;
import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.wiringpi.Spi;
import com.pi4j.component.switches.SwitchListener;
import com.pi4j.component.switches.SwitchState;
import com.pi4j.component.switches.SwitchStateChangeEvent;
import com.pi4j.device.piface.PiFace;
import com.pi4j.device.piface.PiFaceLed;
import com.pi4j.device.piface.PiFaceRelay;
import com.pi4j.device.piface.PiFaceSwitch;
import com.pi4j.device.piface.impl.PiFaceDevice;
import com.pi4j.gpio.extension.piface.PiFacePin;
import com.pi4j.device.piface.impl.PiFaceDevice;
import com.pi4j.wiringpi.Spi;

public class FoobarExample {

    public static void main(String[] args) throws Exception{

        piface = new PiFaceDevice(PiFace.DEFAULT_ADDRESS, Spi.
CHANNEL_0);
        gpio = GpioFactory.getInstance();

        //------------------------ LED
        // turn on LED 1
        piface.getLed(PiFaceLed.LED1).on();

        // turn off LED 1
        piface.getLed(PiFaceLed.LED1).off();

        // blink LED 8 time (1000/8 -> 125ms)
        piface.getLed(PiFaceLed.LED2).blink(125);

        // stop blink
        piface.getLed(PiFaceLed.LED2).blink(0);

        //------------------------ INPUT
        final int input0 =PiFacePin.INPUT_00.getAddress();

        piface.getInputPin(input0).isHigh();
        piface.getInputPin(input0).isLow();

        //------------------------ OUPUT
        final int output0= PiFacePin.OUTPUT_00.getAddress();

        piface.getOutputPin(output0).blink(125);
        if(piface.getOutputPin(output0).isHigh()){
          piface.getOutputPin(output0).low();
        }else{
          piface.getOutputPin(output0).high();
        }
        //------------------------ SWITCH
        piface.getSwitch(PiFaceSwitch.S1).isOn();
        piface.getSwitch(PiFaceSwitch.S1).isOff()

        //------------------------ RELAY
        // close relay 0
        piface.getRelay(PiFaceRelay.K0).close();

        // open relay 1
        piface.getRelay(PiFaceRelay.K0).open();

        // toggle relay 1 (open if close, close if open)
        piface.getRelay(PiFaceRelay.K1).toggle();
    }
}
```

De la même façon que pour l'API Python, il est également possible d'avoir des méthodes qui vont se déclencher à la suite d'une modification sur une entrée :

```
import com.pi4j.device.piface.PiFace;
import com.pi4j.io.gpio.GpioController;
import com.pi4j.io.gpio.GpioFactory;
import com.pi4j.wiringpi.Spi;
import com.pi4j.component.switches.SwitchListener;
import com.pi4j.component.switches.SwitchState;
import com.pi4j.component.switches.SwitchStateChangeEvent;
import com.pi4j.device.piface.PiFace;
import com.pi4j.device.piface.PiFaceLed;
import com.pi4j.device.piface.PiFaceRelay;
import com.pi4j.device.piface.PiFaceSwitch;
import com.pi4j.device.piface.impl.PiFaceDevice;
import com.pi4j.wiringpi.Spi;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class FoobarExample {
    private static final Logger LOGGER = LoggerFactory.getLogger(FoobarExample.class);

    public static void main(String[] args) throws Exception{

        piface = new PiFaceDevice(PiFace.DEFAULT_ADDRESS, Spi.
CHANNEL_0);
        gpio = GpioFactory.getInstance();

        piface.getSwitch(PiFaceSwitch.S1).addListener( new Switch
Listener() {
            @Override
            public void onStateChange(final SwitchStateChangeEvent event) {
                if(event.getNewState() == SwitchState.ON){
                    LOGGER.info("switch S1 - ON");
                }else{
                    LOGGER.info("switch S1 - OFF");
                }

            }
        });
}
```

> Embedded JRE

Pour faire fonctionner correctement l'API sur le Raspberry, il est nécessaire d'utiliser une version allégée du JDK d'Oracle. Cette version est conçue spécifiquement pour le processeur ARM qu'embarque le Raspberry. Suivant les cas, il peut être utile de changer le ratio entre mémoire disponible et mémoire vidéo. Cela est configurable via l'application **raspi-config**. Une fois la mémoire configurée et le Raspberry Pi redémarré, on peut télécharger la version du JRE **ARMv6/7 Linux - Headless - Client Compiler EABI, VFP, HardFP ABI, Little Endian**. Cette version s'installe de la même façon qu'une JVM classique :

```
$ sudo mkdir /usr/java
$ sudo tar xzfC ejre-7u10-fcs-b18-linux-arm-vfp-client_head
less-28_nov_2012.tar.gz /usr/java
```

Une fois la JRE décompressée, il ne reste plus qu'à configurer les variables d'environnement dans le fichier **/etc/profile** :

```
export JAVA_HOME=/usr/java/ejre1.7.0_10
export PATH=$PATH:$JAVA_HOME
```

Dès lors, il est possible de vérifier le bon fonctionnement via la commande :

```
$ java -version
java version "1.7.0_10"
Java(TM) SE Embedded Runtime Environment (build 1.7.0_10-b18, headless)
Java HotSpot(TM) Embedded Client VM (build 23.6-b04, mixed mode)
```

> Coté serveur

Bien que le Raspberry Pi ait peu de puissance, il est possible d'utiliser un conteneur de Servlet ou un serveur d'application pour concevoir une application Web. Cela permet d'interagir avec le module PiFace via le protocole HTTP. Il est donc possible d'installer un serveur tel que Wildfly. Pour cela, il est nécessaire d'apporter quelques configurations :

- Dans le fichier **wildfly\bin\standalone.conf** il est nécessaire de définir l'emplacement de la JRE ainsi que la configuration mémoire :

```
JAVA_HOME="/usr/java/ejre1.7.0_10"
...
JAVA_OPTS="-Xms64m -Xmx196m -XX:MaxPermSize=64m -Djava.net.preferIPv4Stack=true"
```

- Dans le fichier de configuration **wildfly\standalone\configuration\standalone.xml**, il est nécessaire d'autoriser l'accès extérieur sur le serveur. Cette configuration est dans la partie **interfaces** du fichier :

```
<interfaces>
<interface name="management">
<inet-address value="0.0.0.0"/>
</interface>
<interface name="public">
<inet-address value="0.0.0.0"/>
</interface>
<interface name="unsecure">
<inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
</interface>
</interfaces>
```

Une fois l'utilisateur d'administration pour Wildfly créé (via le script **wildfly\bin\add-user.sh**), le serveur est prêt à recevoir votre application Web. Il faut cependant faire attention à certaines particularités de la JRE Embedded. En effet, certaines extensions CDI (Contexts Dependency Injection) se basant sur des pools de threads ne fonctionnent pas correctement.

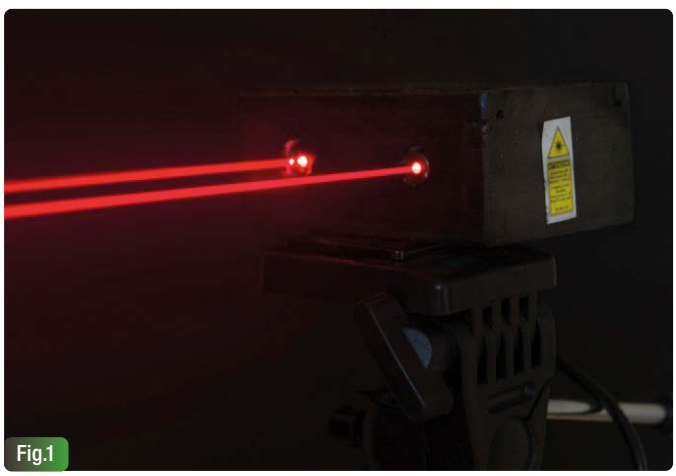


Fig.1

Projet « Stalker »

L'ELECTRONIQUE

L'électronique est un domaine qui peut effrayer. Heureusement, grâce à Internet, on trouve énormément de sites et de chaînes YouTube fournissant de bons tutoriels.

On peut citer par exemple le site http://www.elektronique.fr/ qui contient divers cours sur les principes fondamentaux de l'électronique ou la chaîne Youtube Afrotechmods.

Pour concevoir un système complexe et ainsi obtenir un comportement bien particulier, on utilise divers composants électroniques. Il n'est pas utile de tous les connaître, mais certains sont indispensables :

- Les résistances : c'est le composant de base de tout circuit. Elles ont pour rôle d'éviter les courts-circuits, de modifier la tension et le courant. Certaines d'entre elles réagissent à la température, permettant ainsi de concevoir des dispositifs comme des thermomètres, climatisation automatique, etc.
- Les diodes : elles ont la particularité de laisser passer le courant uniquement dans un sens. Cela pallie entre autres l'inversion des piles par exemple. Dans la même famille, on retrouve les LED qui, en plus de cette spécificité, ont la faculté d'émettre de la lumière.
- Les transistors : ils s'apparentent à des interrupteurs contrôlés. Ils permettent aussi bien de concevoir différents capteurs (de lumière, de pression, etc.) que d'apporter de la logique ou de rediriger le courant. Les transistors sont à la base de beaucoup de circuits imprimés et processeurs.
- Les condensateurs : qu'ils soient chimiques ou capacitifs, ils permettent d'emmagasiner de l'énergie et de la restituer. Ils sont souvent utilisés pour lisser les tensions ou comme filtres dans des circuits à courant alternatif. Soyez vigilants avec les condensateurs chimiques : ces derniers sont conçus pour fonctionner dans un certain sens du courant. S'ils sont inversés, ils explosent ! Privilégiez donc les condensateurs capacitifs si vous êtes débutant.
- L'AOP (Amplificateur Opérationnel) : ce petit composant est très utile et permet énormément de choses comme par exemple la génération de signaux et le calcul mathématique.
- Le NE555 (circuit intégré) : il permet de générer une tension carrée avec une certaine fréquence. Vous pourrez grâce à lui concevoir une télécommande infrarouge par exemple.

RETOUR D'EXPERIENCE PALO IT

Dans le cadre d'un projet interne de R&D chez Palo IT baptisé "Stalker", nous avons eu l'occasion d'éprouver le Raspberry Pi et PiFace. Le but du projet ? **Concevoir un capteur capable de confirmer des prédictions d'affluence**. Le projet était découpé en différentes parties : un frontal en Dart, un middleware en NodeJS et Java accédant à une base Cassandra et le capteur en JavaEE 7 sur le Raspberry Fig.1.

Le capteur

Le capteur est constitué de deux lasers pointant sur deux photo-transistors. Ces derniers agissent comme des interrupteurs. Lorsqu'une personne passe devant l'un des lasers, la résistance du photo-transistor en face augmente considérablement.

Les deux lasers sont espacés de quelques centimètres, permettant ainsi de déterminer le sens de passage des personnes.

Montage

Les phototransistors sont reliés au module PiFace sur ses entrées. Ces photo-transistors ont la particularité d'avoir uniquement deux pattes (au lieu de trois pour des transistors classiques). Ils se comportent exactement comme des interrupteurs classiques. Le montage électronique est donc extrêmement simple.

Les boîtiers ont été conçus à base de boîtes en bois vendues dans des boutiques d'art plastique. Les capteurs devant être à mi-hauteur, ils ont été placés sur des trépieds d'appareil photo. Une simple vis et un boulon ont permis de fixer les capteurs sur ce support. La principale difficulté de ce système est l'alignement des lasers. Il est nécessaire qu'ils soient bien parallèles et placés face aux photo-transistors. Une fois alignés et afin de s'assurer qu'ils ne bougeront pas, ils sont coulés dans de la colle, qui peut être refondue en cas de besoin.

Astuce : Dans le cas de la conception d'un système embarqué, le fait de couler l'électronique dans de la colle ou résine permet une plus grande fiabilité et résistance des circuits. Attention cependant de bien vérifier les soudures. Ce type de technique est idéal pour des montages devant résister à l'humidité, aux variations de températures ou aux contraintes physiques (vibrations, chocs, etc.).

Développement

Afin de tester les capacités du Raspberry et du PiFace, le choix a été fait de réaliser l'application du capteur à l'aide de JavaEE 7. Bien que cette technologie soit « surdimensionnée » pour les besoins d'un petit capteur, cela permet de montrer les possibilités et limites du Raspberry. Je vous invite à découvrir le code source sur le GitHub de Palo IT : https://github.com/Palo-IT/StalkerRaspberry.

Découpage de l'application

Le découpage est assez classique :

- actions : pour les vues JSF, permettant par exemple d'afficher la courbe d'affluence.
- cdi : pour les extensions CDI, comme instanciation au démarrage des beans CDI.
- commons : cette partie contient les classes utilitaires ainsi que les exceptions.
- model : les modèles d'objets utilisés par l'application. c'est ici que l'on

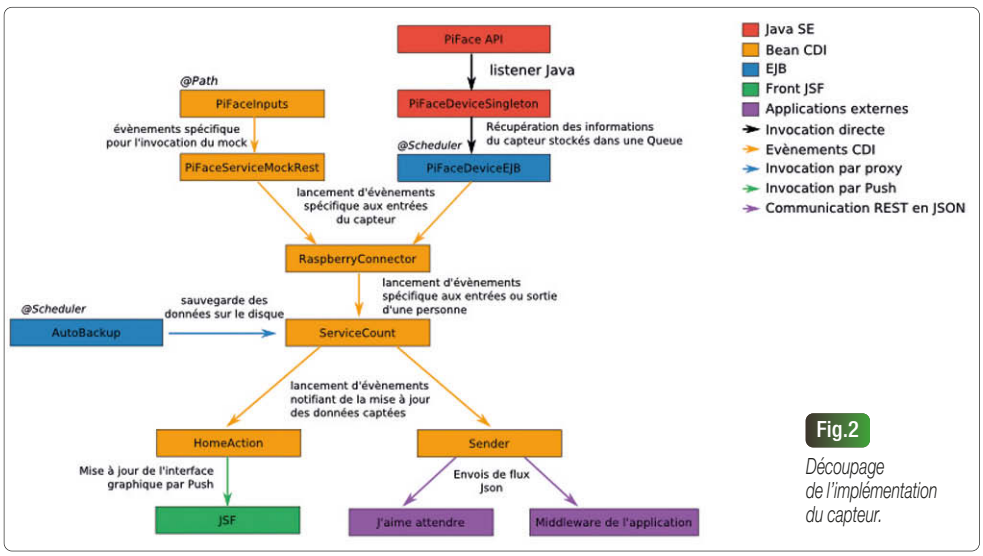


Fig.2
Découpage de l'implémentation du capteur.

retrouve des objets qui servent de retour.

- rest : ce sont les contrôleurs pour les actions avec JAX-RS.
- service : c'est cette partie qui va contenir les implémentations interagissant avec le PiFace.

L'application a la particularité d'être conçue en grande partie grâce à l'utilisation d'événements CDI.

Ce mode de développement permet une grande souplesse au niveau de l'implémentation. Le capteur produit donc des événements qui sont par la suite traités par différents services applicatifs.

Ces derniers en font de même avec d'autres événements affinant l'information **Fig.2**.

Ce type de développement permet également de simuler le capteur en amont du projet via une simple interface REST ou des tests unitaires via Arquillian. La logique de l'application répond à des événements CDI et non à l'information brute fournie par l'API de PiFace. Ces deux parties sont totalement dé-corrélées.

L'utilisation de JavaEE7 a permis de vérifier le bon fonctionnement des technologies de JavaEE 7 sur Raspberry Pi. Dans l'ensemble, ces technologies fonctionnent très bien, mis à part les "pools de thread". Le JDK présent sur Rapberry est spécifique à son processeur ARM et comporte donc des limitations par rapport à un JDK classique. Pour une utilisation ordinaire, cela n'a pas d'impact.

Cependant, si on souhaite mettre en place des événements asynchrones, cette limite nous posera problème.

Les événements CDI sont par défaut synchrones. Les informations du capteur sont donc stockées en mémoire (via une Queue) pour être lues par un EJB "scheduler". On récupère les informations à intervalle régulier pour les transformer en événements CDI :

```
@Stateless
public class PiFaceDeviceEJB implements Serializable {
    private static final long  serialVersionUID = -592353
6347577389767L;
    @Inject
    private Event<Input0Event>  input0Event;
    @Inject
    private Event<Input1Event>  input1Event;

    @Lock(LockType.READ)
    @Schedule(second = "*", minute = "*", hour = "*", persistent = false)
    public void inputObserver() throws InterruptedException {
        final int size = PiFaceDeviceSingleton.getInstance().queueSize();
        if (size >= 4) {
            for (int index = 0; index < size; index++) {
                final Input inputEvent = PiFaceDeviceSingleton.getInstance().poll();

                if (inputEvent != null) {
                    if (inputEvent.getIndex() == 0) {
                        input0Event.fire(new Input0Event(new PinData(inputEvent.getState() == 1)));
                    } else {
                        input1Event.fire(new Input1Event(new PinData(inputEvent.getState() == 1)));
                    }
                }
            }

        }
    }

}
```

La suite de la logique de l'application peut donc observer ces événements et poursuivre la suite du traitement :

```
@Startup
@ApplicationScoped
@Named
public class RaspberryConnector implements Serializable {
    //
================================================
    // INPUT
    //
================================================
    public void observeInput0(@Observes final Input0Event event) {
        LOGGER.info("\tobserve input0 : {}", event.getData().isOn());
        SENSORS[0].setFront(event.getData());
        fireEvent(SENSORS[0]);
    }

    public void observeInput1(@Observes final Input1Event event) {
        LOGGER.info("\tobserve input1 : {}", event.getData().isOn());
        SENSORS[0].setBack(event.getData());
        fireEvent(SENSORS[0]);
    }

    //
================================================
    // FIRE EVENT
    //
================================================
    protected synchronized void fireEvent(final PifaceSensor sensor) {
        if (sensor.isNewState()) {
            if (sensor.isEnter()) {
                enterEvent.fire(new EnterEvent(sensor.getUid()));
            } else if (sensor.isExit()) {
                exitEvent.fire(new ExitEvent(sensor.getUid()));
            }
        }
    }
}
```

CONCLUSION

Le Raspberry Pi se montre très polyvalent. Il est capable d'être utilisé comme station multimédia, serveur et contrôleur pour des capteurs. Bien que sa puissance soit limitée, il est capable d'exécuter bon nombre d'applications. Sa taille et sa consommation énergétique en font un candidat idéal pour réaliser des projets de domotique ou pour de l'embarqué.

Cependant, la conception d'applications Java se montre un peu plus complexe que sur une plateforme classique et plus limité au niveau de la gestion des threads.

Les API pour l'interface PiFace se montrent moins fiables également. Il existe des anomalies dans la version 0.0.5 au niveau des Listeners des entrées. Ce problème est en grande partie corrigé dans la version en cours de développement.

Le développement en Python reste la solution la plus efficace. Les librairies Python du PiFace sont plus matures et performantes.

Dans le cadre de l'utilisation de cette interface, ce langage est une très bonne solution.

Pattern Acteur : décomplexer la complexité de la concurrence et de la tolérance aux fautes

2e partie

Nous avons exposé dans la première partie de l'article, le framework Akka qui implémente le pattern acteur. Ce framework fournit une API de haut niveau pour faire de la programmation concurrente, et permet aux développeurs de s'abstraire des contraintes la programmation multi-thread (gestion des verrous, mélange de code technique et de code métier, etc ...). Avec Akka, les acteurs sont des entités légères, communiquant entre eux, par message, uniquement de façon asynchrone. Ainsi ce framework permet en outre de créer des acteurs, d'envoyer des messages à des acteurs et de recevoir des messages provenant d'un acteur.

Un autre point important avec Akka est la notion de supervision, pour gérer le cycle de vie des acteurs en cas d'erreur. Un acteur est crée et supervisé par son parent. Celui-ci décidera du comportement de son fils lors d'une erreur (arrêt, relance, ...). Cette supervision permet ainsi la création de systèmes qui s'auto réparent et donc de définir une architecture tolérante aux fautes.

Nous allons nous focaliser sur ce point, dans la deuxième partie de cet article.

David Hassoun
Lien Github : https://github.com/agassite/Acteur

Notion de supervision

Nous avons vu que le framework Akka permettait de définir une architecture tolérante aux fautes via la notion de supervision. Pour cela Akka permet de créer un acteur superviseur. Un acteur de type superviseur, va ainsi créer un ou plusieurs acteurs dits « métier », et sera responsable de leurs comportement en cas d'erreur, donc si un des acteurs fils du superviseur déclenche une exception.

Stratégie de supervision

Pour décider du comportement que le superviseur va appliquer aux acteurs, Akka définit deux types de stratégies :

- La stratégie de type **OneForOneStrategy** qui applique le comportement à l'acteur fils du superviseur déclenchant l'exception,
- La stratégie de type **OneForAllStrategy** qui applique le comportement à tous les acteurs fils du superviseur.

Comportement de supervision

Akka dispose de quatre méthodes pour définir le comportement à appliquer lors d'une erreur d'un acteur :

- La méthode resume() qui continue l'exécution de l'acteur fils,
- La méthode restart() qui redémarre l'acteur fils,
- La méthode stop() qui arrête l'acteur fils,
- La méthode escalate() qui remonte l'exception à l'acteur parent.

Remarque : Akka permet de définir notre propre comportement, à appliquer lorsqu'un acteur tombe en erreur.

Comportement par défaut

Si aucun superviseur n'est spécifié, le comportement appliqué par défaut sera le suivant :

Exception	Méthode appliquée
ActorInitializationException	stop()
ActorKilledException	stop()
Exception	restart()
Throwable	escalate()

Etude de cas

Nous allons écrire un acteur superviseur, qui va dans un premier temps créer un acteur de type worker, puis ensuite lui envoyer différents nombres entiers contenus dans un message **IntegerMessage**.
L'acteur worker contient une variable d'instance *state* de type entier (initialisée à 4 par exemple), et va diviser cet entier par le nombre provenant du message qu'il a reçu de son père.

Ecriture du worker

```
public class WorkerActor extends UntypedActor {
    LoggingAdapter log = Logging.getLogger(getContext().system(), this);
    private int state = 4;

    …

    public void onReceive(Object o) throws Exception {
        if (o instanceof IntegerMessage) {
            IntegerMessage message = (IntegerMessage) o;
            //récupère l'entier à traiter
            int value = message.getNombre().intValue();
            state = state/value;
        } else if (o instanceof ResultMessageRequest) {
            getSender().tell(state, ActorRef.noSender());
        }
    }
    …
```

```
}
```

Nous remarquons dans le code ci-dessus, que plusieurs erreurs peuvent se produire dans la méthode *onReceive* :

- Une division par 0 peut être effectuée par le worker ce qui déclenchera une **ArithmeticException,**
- Une valeur null peut être contenue dans le message, ce qui provoquera une **NullPointer Exception.**

Nous voulons par exemple que l'acteur worker continue son exécution en cas de division par 0, et qu'il soit redémarré s'il déclenche une **NullPointerException**.

Ecriture du superviseur

```
package com.poc.acteur.supervisor;

import static akka.actor.SupervisorStrategy.escalate;
import static akka.actor.SupervisorStrategy.restart;
import static akka.actor.SupervisorStrategy.resume;
import scala.concurrent.duration.Duration;
import akka.actor.ActorRef;
import akka.actor.OneForOneStrategy;
import akka.actor.Props;
import akka.actor.SupervisorStrategy;
import akka.actor.SupervisorStrategy.Directive;
import akka.actor.UntypedActor;
import akka.japi.Function;


public class SupervisorActor extends UntypedActor {

   private final  ActorRef childActor;

   public SupervisorActor() {
      childActor = getContext().actorOf(Props.create(WorkerActor.class),
            "workerActor");
   }

   private static SupervisorStrategy strategy = new OneForOne
Strategy(10,
         Duration.create("10          second"),          new          Function<Throwable,
Directive>() {
            public Directive apply(Throwable t) {
               if (t instanceof ArithmeticException) {
                  return resume();
               } else if (t instanceof NullPointerException) {
                  return restart();
               } else {
                  return escalate();
               }
            }
         });

   @Override
   public SupervisorStrategy supervisorStrategy() {
      return strategy;
   }

   public void onReceive(Object o) throws Exception {
      if (o instanceof ResultMessageRequest) {
         childActor.tell(o, getSender());
      } else
         childActor.tell(o,ActorRef.noSender());
   }

   public ActorRef getChildActor() {
      return childActor;
   }


}
```

Ecriture du programme principal

```
package com.poc.acteur.supervisor;

import java.util.concurrent.TimeUnit;

import scala.concurrent.Await;
import scala.concurrent.Future;
import scala.concurrent.duration.Duration;
import akka.actor.ActorRef;
import akka.actor.ActorSystem;
import akka.actor.Props;
import akka.event.Logging;
import akka.event.LoggingAdapter;
import akka.pattern.Patterns;

/**
 * Programme principal
 *
 */
public class Main {

   public static void main(String[] args) throws Exception {

      ActorSystem system = ActorSystem.create("faultTolerance");
      LoggingAdapter log = Logging.getLogger(system, system);

      //creation de l'acteur superviseur
      ActorRef supervisor = system.actorOf(
            Props.create(SupervisorActor.class), "supervisor");


      ResultMessageRequest request = new ResultMessageRequest();

      //Envoi d'un entier au superviseur
      log.info("Envoi de  l'entier 2 ...");
      supervisor.tell(new IntegerMessage (2),ActorRef.noSender());

      //Future pour récupérer  réponse provenant du superviseur
      Future<Object> future = Patterns
            .ask(supervisor, request, 5000);
      Integer result = (Integer) Await.result(
            future, Duration.create(5000, TimeUnit.MILLISECONDS));

      //La valeur attendue est 4/2 -> 2
```

```
        log.info("Valeur recue-> {}", result);
        assert result.equals(Integer.valueOf(2));

        log.info("Envoi de  l'entier 0 ...");
        supervisor.tell(new IntegerMessage (0),ActorRef.noSender());

        //Future pour récupérer  réponse provenant du superviseur
         future = Patterns
                .ask(supervisor, request, 5000);
         result = (Integer) Await.result(
                future, Duration.create(5000, TimeUnit.MILLISECONDS));

        log.info("Valeur recue-> {}", result);

        //L'exception ArithmetiqueException a été déclenchée par le worker
        //Le worker continue, donc la valeur renvoyée est toujours 2
        assert result.equals(Integer.valueOf(2));

        log.info("Envoi de l'entier wrappe avec une valeur null ...");
        supervisor.tell(new IntegerMessage (null),ActorRef.noSender());

        //Future pour récupérer  réponse provenant du superviseur
         future = Patterns
                .ask(supervisor, request, 5000);
         result = (Integer) Await.result(
                future, Duration.create(5000, TimeUnit.MILLISECONDS));

        //L'exception NullPointerException a été déclenchée par le worker
        //Le worker est redémarrée, la valeur renvoyée est la valeur initiale 4
        log.info("Valeur recue-> {}", result);
        assert result.equals(Integer.valueOf(4));


        log.info("Worker Actor shutdown !");
        system.shutdown();

    }
}
```

Exécution

La trace d'exécution ci-après montre que lorsque le worker reçoit la valeur 2, la valeur 2 (4/2) est bien reçue par la Future.
Ensuite lorsque le worker reçoit la valeur 0, une **ArithmeticException** est déclenchée. Son exécution continue et la valeur reçue est donc toujours 2.
Puis, enfin, lorsque le worker déclenche une **NullPointerException** , il est redémarré, et la valeur reçue par la Future est donc la valeur initiale 4.

```
[INFO] [10/05/2014 17:54:02.212] [main] [ActorSystemImpl(akka://faultTolerance)] Envoi de  l'entier 2 ...
[INFO] [10/05/2014 17:54:02.221] [main] [ActorSystemImpl(akka://faultTolerance)] Valeur reçue-> 2
[INFO] [10/05/2014 17:54:02.221] [main] [ActorSystemImpl(akka://faultTolerance)] Envoi de  l'entier 0 ...
[WARN] [10/05/2014 17:54:02.227] [faultTolerance-akka.actor.default-dispatcher-3] [akka://faultTolerance/user/supervisor/workerActor] / by zero
[INFO] [10/05/2014 17:54:02.228] [main] [ActorSystemImpl(akka://faultTolerance)] Valeur reçue-> 2
[INFO] [10/05/2014 17:54:02.228] [main] [ActorSystemImpl(akka://faultTolerance)] Envoi de  l'entier wrappe avec une valeur null
...
[ERROR] [10/05/2014 17:54:02.230] [faultTolerance-akka.actor.default-dispatcher-4] [akka://faultTolerance/user/supervisor/workerActor] null
java.lang.NullPointerException
...
[INFO] [10/05/2014 17:54:02.234] [main] [ActorSystemImpl(akka://faultTolerance)] Valeur reçue-> 4
[INFO] [10/05/2014 17:54:02.234] [main] [ActorSystemImpl(akka://faultTolerance)] Worker Actor shutdown !
```

CONCLUSION

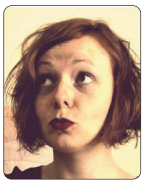
Nous avons vu à travers cet article que le framework Akka fournit une API de haut niveau, pour faire de la programmation concurrente, sans se préoccuper des problématiques récurrentes de la programmation multi-threadés. Akka est d'ailleurs utilisé dans l'outil Gathling permettant d'effectuer des tests de performances.
D'autre part avec la notion de supervision, Akka permet de mettre en œuvre la tolérance aux fautes.
Avec son système de messages aynchrones, Akka est une solution intéressante pour implémenter des architectures pilotées par des événements (EDA), distribuées, performantes et hautement disponibles.

SOURCES

Vous trouverez les sources des différents exemples de cet article sur mon github, à l'URL suivante : https://github.com/agassite/Acteur .

Conception d'interface tactile 1ère partie : la conception et processus projet

C'est incontournable, aujourd'hui notre monde devient « naturel »; il faut comprendre par là qu'il est de plus en plus constitué d'interfaces naturelles. Que ce soit par les gestes (Kinect, Leap Motion), par la voix (Siri, Cortana), les yeux avec l'eye tracking ou encore le toucher avec le tactile, nous sommes devenus des communicants épris de simplicité.



Johanna Rowe **Calvi**
Designer d'interaction et d'expérience numérique chez Expertime
MVP Hardware Interaction Design & Development

Nicolas **Calvi**
Consultant Informatique, Expert NUI chez Expertime
MVP Hardware Interaction Design & Development

Nous allons aborder ici l'interface la plus répandue actuellement, l'interface tactile. Ecrans de toutes les tailles tels que les smartphones, tablettes, bornes ou encore électroménager et tables, nous avons tous aujourd'hui, utilisé au moins une fois une interface tactile.
C'est même devenu un enjeu de taille pour les entreprises, pour le BtoC dans le but par exemple d'améliorer la visibilité d'une marque, le BtoB pour améliorer la productivité des salariés en situation de mobilité et le BtoE pour le bien-être des employés.
Devant cet enjeu grandissant, il est important de connaître ce qu'inclut le développement d'une interface tactile, comment s'y prendre, quels métiers et surtout ce que cela change par rapport au développement d'autrefois. Car oui, développer une interface tactile dite naturelle ne s'improvise pas, pour toucher au but il faut savoir mettre en place les bonnes équipes et les bonnes méthodologies.

C'est quoi une interface tactile ?

Une interface tactile, comme son nom l'indique est une surface manipulable avec les doigts. Que ce soient des applications smartphones, tablettes, des tables, des bornes ou même des interfaces projetées. Ce qu'il faut comprendre c'est qu'à l'heure actuelle de nombreuses technologies existent et sont améliorées chaque jour par des départements de R&D. On peut dorénavant rendre n'importe quelle surface tactile. Que ce soit une feuille de plastique souple qui deviendra un journal quotidien se mettant à jour tous les matins, le miroir de votre salle de bain qui servira à la projection et la manipulation de vos données de santé ou encore vos interactions tactiles avec vos plantes grâce au projet de R&D du laboratoire Disney Research permettant d'obtenir par exemple des notifications en caressant les feuilles de votre Yucca (cf Botanicus).
Les interfaces tactiles ont été créées dans le but de simplifier l'utilisation des technologies de plus en plus complexes à comprendre et à manipuler. On parle de plus en plus de fracture numérique, cette fracture est due à l'évolution rapide de la digitalisation de notre monde. De plus en plus d'actions obligatoires, professionnelles ou personnelles telles que des tâches administratives, nécessitent un minimum d'acquis.
L'objectif des interfaces tactiles, et plus largement des interfaces naturelles, a pour but de réduire cette fracture numérique en facilitant l'utilisation des « outils informatiques » afin que la navigation et les actions deviennent intuitives pour les utilisateurs ciblés.
L'objectif est double : faciliter le quotidien et faire disparaitre visuellement la complexité technologique des actions utilisateurs. On parle même à terme de faire disparaitre la technologie (comme avec le projet Botanicus).

Les 10 étapes clés de la phase de conception centrée utilisateur

Lorsque l'on commence à travailler sur un nouveau projet, il convient d'appliquer un processus particulier afin de minimiser les coûts et la perte de temps. Ce processus de design peut être appliqué à tout type de projet. Bien qu'il soit évident dans certains types de domaines comme les projets de conception de mobilier ou de voitures depuis des années, il est rarement appliqué dans le monde de l'informatique. Ce processus est un mélange de connaissance de l'Homme, d'empathie, de créativité, de bricolage, de curiosité et d'analyse. La créativité est un des paramètres les plus importants quand on souhaite trouver des solutions innovantes. Les dix étapes ci-dessous sont un mélange entre des processus existants et les modifications que j'ai apportées grâce aux 5 dernières années passées sur le terrain avec des utilisateurs de types très variés.

La problématique

A quelle problématique devez-vous répondre en termes de besoins ? La première phase est de réussir à formuler par écrit et de manière simple la problématique à laquelle votre interface doit répondre. Quand vous travaillez sur une interface métier les mots clés qui ressortiront seront en général « augmenter la productivité », « réduire le nombre d'étapes », « simplifier l'interface », « faire en sorte que le nouvel outil soit accepté », etc. Si vous travaillez sur une interface BtoC les mots clés seront plutôt « augmenter le nombre de commandes », « faire une interface attractive », « augmenter la rétention des utilisateurs » etc.

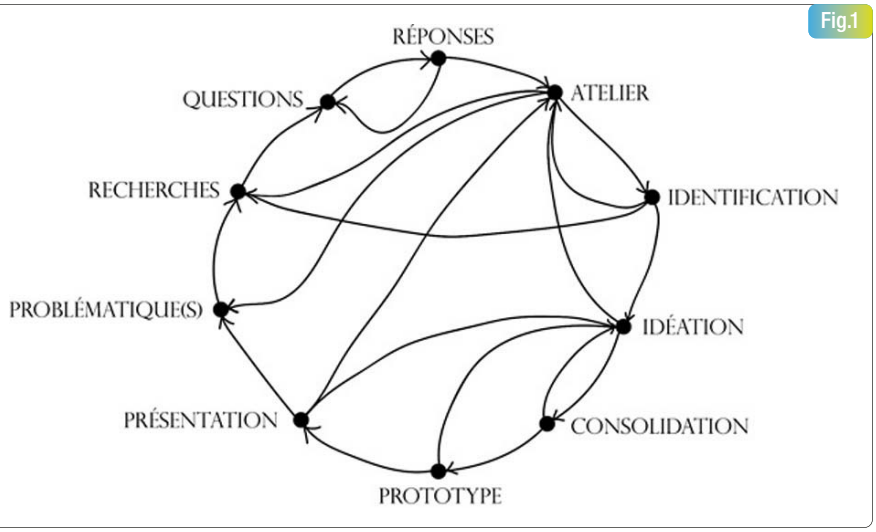
Les recherches

Cette phase semble assez évidente mais est souvent bâclée. Elle est pourtant indispensable afin de faire une application tactile pertinente. Souvent les gens pensent avoir des idées incroyablement innovantes, pourtant en général ces mêmes idées ont déjà été trouvées par d'autres. La base de l'innovation est la connaissance de ce qui existe, des limites et des critiques de ces outils. La phase de recherche doit donc regrouper les connaissances de l'existant (concurrent ou non), les solutions même très éloignées dans le même domaine, qu'elles soient par exemple interfaces tactiles, Web ou objets physiques.

Les questions

Se poser des questions aide à clarifier le contexte du projet. Où, quand, qui, comment, pourquoi, pour qui, quand ? Cela parait basique mais quand on prend par exemple « pour qui », voici les informations qui peuvent en découler : le type d'utilisateurs, leur âge, leur appétence à

Fig.1



la technologie, le nombre d'utilisateurs simultanés. On peut ainsi déterminer le niveau d'intuitivité nécessaire, un besoin éventuel de formation au nouvel outil, la nécessité de faire un plus ou moins grand nombre d'ateliers avec les utilisateurs.

Les réponses

A cet instant vous n'avez pas encore confronté vos recherches et analyses au client ou représentant des utilisateurs, c'est donc le moment de le faire. Vous allez donc prendre toute votre analyse et l'exposer de manière à ne pas influencer votre interlocuteur. Le but est de faire une première validation du contexte projet et des problématiques à résoudre.

Les ateliers

Organiser des ateliers vous permet à la fois d'obtenir des informations précieuses mais aussi de vous créer des soutiens pour le projet. Pensez à former un groupe homogène et à ne pas mettre un profil problématique dans ce groupe (des utilisateurs + un supérieur hiérarchique par exemple).
Ce que l'on cherche ici à éviter sont des phrases du type « c'est moi le chef donc c'est moi qui décide » que j'ai personnellement déjà vécu au début de ma carrière lors d'un atelier.
N'oubliez pas que la clé du succès d'un atelier est sa légèreté, car c'est cette légèreté qui apportera la créativité.

L'identification

Vous pouvez maintenant identifier l'ensemble des problématiques et des besoins des utilisateurs en analysant les résultats du ou des ateliers. C'est aussi le moment où vous pouvez identifier les personnes qui sont engagées et seront vos soutiens et vos points de contact quand vous aurez besoin d'échanger et de tester vos idées.

L'idéation

Il est important de prendre conscience qu'à cette étape et depuis le début vous êtes toujours en phase d'ouverture. Les contraintes techniques ne doivent pas bloquer vos idées. C'est le moment de créativité avec un grand C. Le designer à cette étape se retrouve seul et fait marcher son imagination pour trouver des concepts d'interfaces, d'interactions, des gestuelles, des scénarios différents afin d'élargir les possibilités et d'aller trouver ce qui fera réellement la différence.

Consolidation

Il s'agit ensuite d'observer son travail de manière critique, de trier les idées, de modifier des étapes pour ensuite les confronter aux autres membres de l'équipe. L'équipe technique peut par exemple à ce moment déterminer la faisabilité et les modifications à apporter.

Prototyper

Pour un designer, concevoir pour une interface tactile ne change pas son processus, sauf lors de cette étape ou les tests de ses idées pourront se faire grâce à des morceaux de papier découpés à échelle 1 afin de tester leur utilisabilité avec les doigts. Dans les autres domaines du design on fera des maquettes en mousse par exemple, ou en clay pour les designers transport. Le prototype pourra dans un second temps être réalisé sur un outil informatique de prototypage comme Pentotype ou Invision.

Présenter

Le but de l'étape de présentation est de mettre dans les mains des utilisateurs une première ébauche, ce qui leur permettra de pouvoir manipuler de manière basique l'interface sans coût de développement ni graphisme. Le but ici est de présenter, d'échanger, d'expliquer et surtout d'écouter sans jamais essayer d'influencer pour être sûr de la qualité des feedbacks.

Le workflow

Bien que le processus tel que décrit ci-dessus soit linéaire, la vérité est qu'il ressemble en général plutôt au schéma de la **figure 1**.
Cette phase n'est pas linéaire et il ne faut jamais hésiter à retravailler une de ces phase car c'est ainsi que votre solution sera différente, aboutie, innovante. Elle vous permet d'avoir une base solide sur laquelle construire.

Du client à l'application finale : schéma méthodologique

Si l'on résume tout ce que l'on vient de dire, un processus projet tel que nous l'appliquons de façon macro peut se présenter à la manière de la **figure 2**. On remarquera les différentes phases qui se chevauchent et l'entrée des différents métiers dans le processus projet.
Il est important que durant toutes ces phases, la communication soit primordiale entre les équipes afin de détecter et éliminer tout ce qui pourrait nuire au bon déroulement du projet.

Développer une interface tactile, de nouvelles contraintes et de nouveaux mots pour les développeurs

Créer une interface tactile, c'est repenser le paradigme du développement. En effet, comme nous l'avons dit précédemment, l'expérience utilisateur étant au centre de vos développements, il faut savoir travailler avec d'autres profils. De manière générale, un Designer d'interaction et un Graphiste sont les deux profils nécessaires pour créer la meilleure interface, mais surtout pour qu'elle soit innovante.

Deux mondes, deux visions, mais un même but

La première chose à comprendre, c'est qu'il ne faut pas brider la créativité des métiers créatifs; trop souvent les storyboards et wireframes complexes du designer ou les visuels aboutis du graphiste ont tendance à faire botter le développeur en touche. Il va donc trouver plein d'excuses techniques pour ne pas faire ce qui est demandé.
Il faut savoir prendre des risques et essayer de développer toutes les idées; ce n'est qu'après une réelle complexité rencontrée que l'on demandera au Designer ou au Graphiste d'adapter son travail pour contourner cette

difficulté. C'est là qu'est le nerf de la guerre, la communication entre le développeur et les créatifs. Il faut savoir les comprendre et se faire comprendre, car ici on parle de la confrontation de deux mondes qui ne se comprennent pas toujours. La contrainte est de savoir tirer parti de l'ensemble des compétences de chacun, et surtout quand votre graphiste vous dit que ses couleurs sont les bonnes, il a raison, c'est son métier pas le vôtre, ayez confiance dans vos équipes créatives.

Chaque mot a son importance

Ce qui change aussi, c'est le vocabulaire employé par chacun. Tous les métiers ont leur jargon technique, que ce soit le développeur, le designer ou le graphiste. Apprendre le jargon des autres métiers est souvent la clé pour une meilleure communication. Cela va dans les deux sens, il faut que le développeur apprenne à parler créatif, comme les créatifs doivent faire un effort pour comprendre le développeur.
Il n'est pas question ici de connaître tout le spectre de vocabulaire de chaque métier, mais de connaître a minima une base qui facilite les échanges et donc améliore la productivité de toute l'équipe.

Travailler en équipe, plus que jamais

Impliquer chaque métier dans toutes les phases de développement est crucial. Le designer doit montrer son travail en amont au développeur pour qu'il analyse et détecte les potentiels problèmes. De même le développeur doit informer le designer des problématiques techniques inhérentes aux technologies qui vont être utilisées pour qu'il sache adapter l'expérience utilisateur.

C'est une contrainte bien connue, mais dans le développement d'une interface tactile qui fait intervenir autant de métiers, ne pas être soudés mène souvent à un développement encore plus chaotique et toujours à une interface non aboutie.

Etre précis, même quand il s'agit de graphisme ou d'interaction

L'un des vieux réflexes des développeurs, c'est leur légèreté à intégrer avec précision les éléments graphiques ou les différentes interactions conçues par les métiers créatifs. Un rectangle de 50 par 50 pixels n'est pas un rectangle 47 par 47 pixels, un texte de 20 points n'est pas un texte de 19 points. La précision de l'intégration sera la clé pour que ce qui a été conçu ressorte avec le rendu le plus optimal.
Il en va de même avec les interactions, un clavier qui doit s'afficher en dessous d'une TextBox ne doit pas s'afficher sur le côté ou en haut; si cela a été pensé ainsi par le designer, c'est qu'il y a une raison. Il faut bien garder à l'esprit que c'est souvent de tous petits détails qui ruinent l'expérience d'une application, il en va donc de la responsabilité du développeur de respecter le travail du designer et du graphiste et donc de faire les choses comme elles ont été prévues. Cela ne vous viendrait pas à l'idée de changer une règle de gestion de votre code (par exemple accepter une lettre dans un champ numérique), il en va de même pour eux, leur travail ont des spécifications, il faut donc les traiter comme telles.

Evoluer pour mieux innover

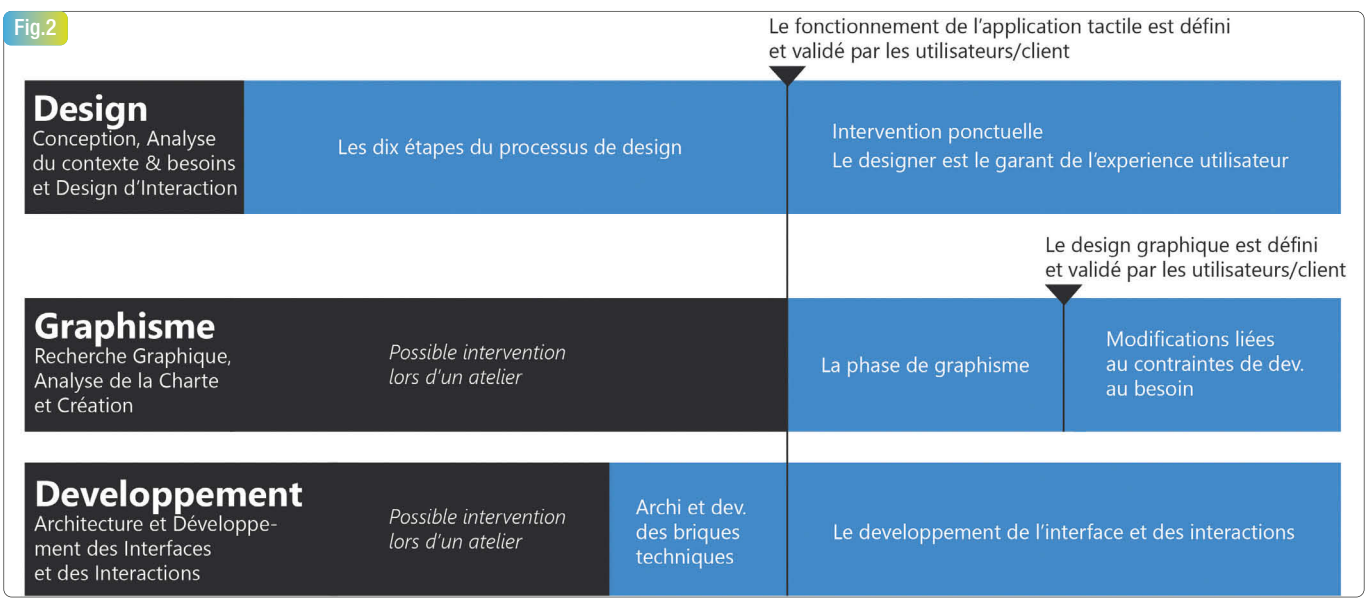
Au final, trop de développeurs restent dans leur zone de confort et ne s'ouvrent pas vers la nouveauté. Ces développeurs ne donneront pas de bons résultats sur des applications tactiles, il faut savoir se remettre en question et écouter les autres, sans pour autant se laisser marcher sur les pieds.
Il faut aussi savoir aller vers de nouvelles technologies, de nouveaux médias, étendre le champ d'action et souvent l'apprentissage d'un nouveau Framework où l'utilisation d'une nouvelle librairie permet de changer sa vision des choses, d'apprendre, et donc de fournir un code plus efficace et adapté.
C'est une façon différente de faire les choses, mais comme dans tous les domaines, c'est par ce genre de changements et d'ouvertures qu'on innove et que les applications que vous allez produire seront différenciantes, mais surtout utiles pour les utilisateurs.

Nous vous avons donc présenté nos conseils pour créer une expérience tactile aboutie et utile. En quelques mots, cela peut se synthétiser par une bonne équipe pluridisciplinaire et une bonne communication de groupe et le fait de prendre son temps lors de la phase de design.
Cependant, les problématiques de support et les demandes des clients peuvent complexifier les choses. Dans une seconde partie, nous aborderons les problématiques des périphériques, les règles à connaître pour un bon développement réussi et comment contourner les contraintes de vos clients et utilisateurs.

Liens :

Pentotype => http://www.pentotype.com/
Vidéo Botanicus => http://bit.ly/1eYRqwr

Fig.2

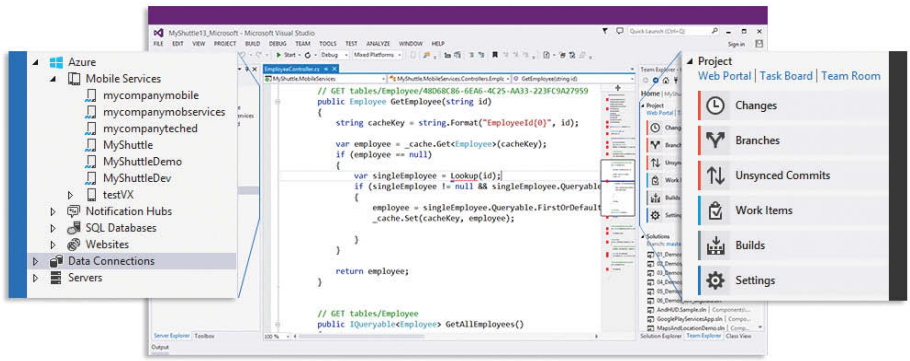


Visual Studio Community 2013

Lors de la conférence Connect() de novembre dernier à New-York, de nombreuses annonces ont été faites (ouverture du Framework .Net, Visual Studio 2015, ...). Une information un peu moins médiatisée mais tout aussi importante pour la sphère des développeurs est la sortie d'une nouvelle édition de Visual Studio appelée « Community ».

Teddy DESMAS
et Jonathan ANTOINE
Consultants
Infinite Square

INFINITE SQUARE



Qu'est-ce que c'est ?

Visual Studio Community 2013 est la nouvelle version du célèbre IDE de la firme de Redmond. Il existait jusqu'à lors 4 grandes versions, de la plus complète (Ultimate), aux plus allégées (Express) en passant par les versions Premium et Pro. La version Ultimate est et restera celle qui propose le maximum de fonctionnalités. En contrepartie, elle nécessite de souscrire un abonnement parfois onéreux selon le contexte. C'est pourquoi Microsoft proposait les versions Express de son IDE. Ces versions gratuites étaient mises à disposition des jeunes ou de toutes autres personnes passionnées souhaitant réaliser des applications et n'ayant pas de besoins forts. Ces versions étaient très limitées en termes de fonctionnalités, ne permettaient pas de jouir des nombreuses extensions de Visual Studio et étaient bridées à un seul type de langage et un seul type de développement (Web, Windows, Windows Phone, etc.). Typiquement les personnes souhaitant réaliser un site Web devaient installer Visual Studio Express for Web et s'ils souhaitaient également réaliser une application Windows 8, ils devaient installer la version for Windows Desktop. Visual Studio Community a pour vocation de remplacer les versions Express et de ne proposer qu'une seule et même version gratuite de Visual Studio, l'édition Community. Les versions Express déjà en place continueront d'être disponibles mais aucune nouvelle édition ne devrait être proposée. Visual Studio Community est basée sur la version Pro de Visual Studio et propose de nombreuses fonctionnalités permettant de réaliser des applications diverses.

Présentation des fonctionnalités

"Puissante", "Extensible" et "Polyvalente", ce sont les trois mots choisis par Microsoft pour décrire cette version de Visual Studio.

Community est exactement l'équivalent de la version professionnelle de Visual Studio 2013. La matrice de comparaison des versions de Visual Studio 2013 est disponible en ligne (http://www.visualstudio.com/en-us/products/compare-visual-studio-products-vs.aspx) mais pour simplifier : toute la partie développement est présente et il vous manquera peut-être des fonctionnalités de tests (Coded UI Tests, test de charges Web, couverture de code, etc.), la partie architecture (diagramme, etc.) ou encore quelques fonctionnalités de travail collaboratif (Code review, etc.).

Puissante

Visual Studio Community 2013 permet d'utiliser de nombreux langages de programmation (C#, Visual Basic, C++, F#, JavaScript, Pyhton, HTML5) pour créer des applications cross-plateformes de l'application native (desktop, store) au Web, en passant par le Cloud.
Les développeurs souhaitant partager leur travail avec d'autres ne seront pas en peine puisqu'ils pourront également utiliser Git et VSO nativement supportés. Il est également bon de rappeler que Microsoft propose aussi un accès gratuit à Visual Studio Online (jusqu'à 5 utilisateurs gratuits) pour pouvoir profiter pleinement des services de Visual Studio Online.

Extensible

Un des grands reproches faits aux versions Express était notamment l'impossibilité d'utiliser les extensions de Visual Studio. C'est maintenant terminé car Visual Studio Community 2013 propose à tous ses utilisateurs la possibilité d'installer toutes les extensions présentes dans la galerie de Visual Studio (http://aka.ms/vsgallery) pour ainsi accroître leur efficacité. vLes utilisateurs pourront donc avoir accès gratuitement aux incontournables comme les Productivity Power Tools, Web Essentials, Xamarin, Tools for Unity, Tools for Python, Resharper .... Il en existe déjà plusieurs milliers à ce jour.

Polyvalente

Une autre fonctionnalité proposée par Community 2013 est la possibilité de réaliser des applications cross-plateformes (iOS, Android, Windows, Linux, etc.) en utilisant par exemple Apache Cordova (HTML5, CSS, JS), Unity (C#, JS combiné avec un player natif) ou encore Xamarin (C# et interfaces graphiques natives). Pour cela il vous faudra télécharger des extensions qui vous apporteront le support nécessaire dans Visual Studio. Comme vous l'aurez compris, Microsoft à travers Visual Studio Community 2013, souhaite mettre en avant le développement multi-plateformes. Nous allons donc expliquer plus en détails ces trois extensions qui raviront les développeurs souhaitant se lancer dans la création d'applications et souhaitant cibler un maximum d'utilisateurs finaux.

Apache Cordova

Apache Cordova est un projet Open Source qui permet, en utilisant toutes vos compétences dans les standards du Web (HTML, CSS, et Javascript), de créer une seule et même application pour toutes les plateformes supportées. D'Android (2.3.3 et plus), à iOS (6, 7, 8), en passant par Windows (8, 8.1) et Windows Phone (8, 8.1) ou encore Blackberry, vous pourrez créer des applications et accéder aux nombreuses API natives présentes sur les

appareils (appareil photo, géolocalisation, barcode scanner, ...) tout en continuant à utiliser vos compétences de développeur Web.
L'extension « Visual Studio Tools for Apache Cordova », permet également de tester et de débugger vos applications grâce aux différents simulateurs disponibles, mais aussi débugger sur votre propre appareil (Windows, Android) pour ainsi s'assurer que l'application fonctionne comme vous l'aviez imaginé (visuellement et fonctionnellement). Ce type de projet permet bien sûr de bénéficier de tous les outils proposés par Visual Studio pour faciliter vos développements : débugger, IntelliSense, coloration syntaxique...

Unity

Unity est l'un des logiciels et moteurs 2D/3D, le plus utilisé pour créer des jeux, de l'animation ou représenter des scènes 3D/2D. Unity est basé sur l'utilisation de la version « Mono » du framework .NET.
L'extension « UnityVS » permet d'intégrer cette technologie dans Visual Studio et ainsi réaliser des jeux en ayant d'une part notre environnement de développement préféré, mais aussi en utilisant nos compétences en C#. En combinant le développement sur Visual Studio et l'utilisation du logiciel Unity pour tester le rendu visuel de l'application et notamment de ses animations, vous avez tout le confort nécessaire pour réaliser un jeu, le tester et le publier sur toutes les plateformes que vous souhaitez cibler.
L'extension vous permettra notamment de profiter de l'expérience de débogage de code de Visual Studio (pas à pas, fenêtre d'exécution immédiate, modification de variables, fenêtre de log) tout en exécutant votre scène 2D/3D dans l'éditeur d'Unity. L'extension est aussi parfaitement intégrée dans Unity : Visual Studio devient l'éditeur de code par défaut et Unity sait ouvrir vos fichiers sur la bonne ligne de code au besoin.
L'extension, développée et supportée par Microsoft, se met à jour régulièrement pour apporter des corrections de bugs et s'adapter aux montées de version d'Unity.

Xamarin

Xamarin est une autre des extensions présentes qui permet de réaliser des applications natives à partir d'un seul code. Xamarin permet aux développeurs C# de réaliser une seule et même logique de code pour cibler toutes les plateformes mobiles (Android, iOS et Windows Phone).
Contrairement à Apache Cordova, les applications réalisées sont transformées par Xamarin en applications natives, et sont donc plus performantes.
Le développeur devra cependant réaliser les IHM pour chacune des plateformes cibles.

Bien d'autres extensions disponibles

Bien sûr le développement cross-plateformes n'est pas la seule possibilité et vous pourrez utiliser des SDKs de tous bords pour vos projets. Ces nouvelles versions ont notamment été annoncées :

- Azure SDK for .NET 2.5
- Visual Studio Tools for Unity (VSTU) 2.0 Preview
- Kinect for Windows 2.0 SDK RTW
- Visual Studio Tools for Apache Cordova CTP3
- TypeScript 1.3

Dans chacun des cas, vous serez dans un environnement de travail bénéficiant de tous les avantages de Visual Studio : prise en charge de l'IntelliSense, coloration syntaxique, inspection du DOM, debugger temps réel, etc.
Les développeurs XAML apprécieront aussi la présence de Blend dans cette édition Community de Visual Studio. Pas de compromis pour cette version donc.

À qui est-elle destinée ? Combien ça coûte ? Qui doit payer ?

Visual Studio Community 2013 est gratuit dans la plupart des cas. Vous avez bien lu : **gratuit !** Évidemment, cela est un peu simplifié et cela reste payant pour les grosses entreprises.
Visual Studio Community 2013 est gratuit dans ces cas :

- Vous êtes un développeur indépendant et vous réalisez des applications gratuites ou payantes,
- Vous êtes un étudiant ou une école,
- Vous travaillez sur un projet Open-Source,
- Vous êtes une entreprise de moins de 250 PC ou de moins d'un million de dollars de revenus annuels.

Si vous n'êtes pas dans un des cas, il vous sera demandé de passer par le système de licence traditionnel de Visual Studio.
Je vous invite cependant à lire le document de licence complet :
http://www.visualstudio.com/support/legal/dn877550

Pourquoi ?

Depuis l'arrivée de Satya Nadella à la tête de Microsoft, une transformation vers beaucoup plus d'ouverture est en train de se produire. L'arrivée de cette nouvelle mouture gratuite en est un des signes forts.
Visual Studio représente plus de 1,8 milliard d'installations et 7 millions de téléchargements uniquement sur l'année passée. C'est un outil majeur de Microsoft qui souhaite avec cette édition gratuite attirer encore plus de développeurs sur sa plateforme.
Avec cette nouvelle édition, l'annonce de l'entrée en Open-Source de toute la partie .NET Server et .Net Core ainsi que la possibilité d'héberger une application ASP.Net sur Linux ou OS X, Microsoft a un message clair : **développeurs, venez sur notre plateforme, nous vous attendons !**

Conclusion

Avec cette nouvelle mouture de Visual Studio 2013, équivalente de la version PRO, Microsoft affirme son message d'ouverture. Ce n'est pas une version édulcorée difficile d'accès ou limitée par des restrictions difficilement compréhensibles comme dans les versions express, mais une seule édition, facile à prendre en main et à comprendre. Il ne nous reste plus... qu'à développer de belles applications !

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 :
F. Bordage, J. Chatard, S. Saurel

Secrétaire de rédaction : Olivier Pavie
Experts : J. Thiriet, T. Solignas, F. Bensusan, M. Frappat, J. Antoine, A. Legiret, F. Hebrard, J. Wortel, J-S Dupuy, F. Barbin, Y. Giffoni, S. Ollivier, S. Nichele, P. Guillerm, D. Hassoun, J. Rowe Calvi, N. Calvi, T. Desmas, S. Sibué, K. Sibué, M. Audouin, M. Nebra, C. Peugnet, L. Probst, S. Moallic, C. Villeneuve.

Crédits couverture : © 11-05-14 © Intense_Media / iStock

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

Microsoft

tech·days 2015

10, 11, 12 février
au Palais des Congrès de Paris

Le rendez-vous de l'innovation numérique

Ambient Intelligence*

Mobilité, big data, machine learning, cloud, objets connectés :
l'Ambient Intelligence sera au cœur des sessions,
des zones d'exposition et de la liveTV

Inscrivez-vous sur
www.mstechdays.fr

#mstechdays