PROGRAMMEZ.COM

PROGRAMMEZ!

Mensuel n°180 - Décembre 2014

le magazine du développeur

Facebook Parse

Développez rapidement vos applications mobiles

DevOps

= Développement + Exploitation

DevOps va-t-il tuer le développeur ?

Le développement continu

Les outils, les bonnes pratiques

Le DevOps au quotidien

SWIFT

le nouveau langage d'Apple

.Net / Visual Studio

Microsoft supporte Android, iOS, Linux, OS X et Xamarin

Carrière

Testeur : un métier d'avenir

Web Design

Utiliser le One Page

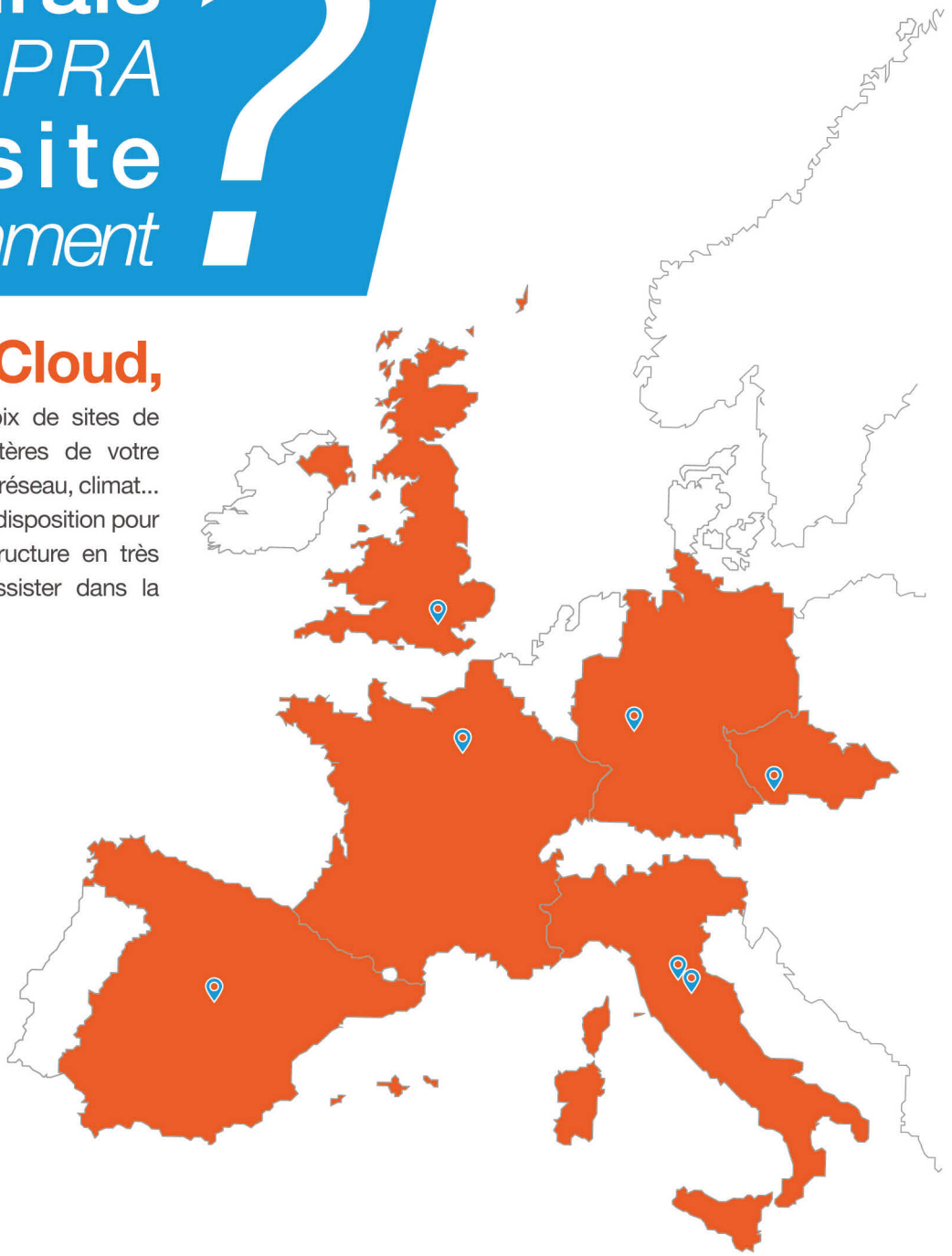
M 04319 - 180 - F: 5,95 € - RD

Printed in EU - Imprimé en UE - BELGIQUE 6,45 € SUISSE 12 FS - LUXEMBOURG 6,45 € - DOM Surf 6,90 € Canada 8,95 $ CAN - TOM 940 XPF - MAROC 50 DH

Je voudrais *bâtir un PRA* **multi-site** *Je fais comment* ?

Avec Aruba Cloud,

vous disposez d'un large choix de sites de secours, en fonction des critères de votre stratégie de sécurité: proximité, réseau, climat... Nos équipes sont aussi à votre disposition pour vous aider à bâtir une infrastructure en très haute disponibilité et vous assister dans la définition de votre stratégie.



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.

Le boss.
ftonic@programmez.com

Les lois d'Asimov sont-elles flippantes ?

« 1 un robot ne peut porter atteinte à un être humain, ni restant passif, ni permettre qu'un être humain soit exposé au danger,

2 un robot doit obéir aux ordres que lui donne un être humain sauf si de tels ordres entrent en conflit avec la première loi,

3 un robot doit protéger son existence tant que cette protection n'entre pas en conflit avec la première ou la deuxième loi. »

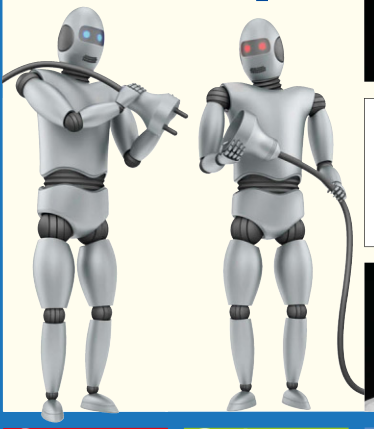
Ces Lois furent écrites par le maître de la Science-Fiction, Isaac Asimov, en 1942. Elles peuvent donner à réfléchir sur le futur rôle de la robotique et des machines évoluées et intelligentes. A notre sens, il ne faut pas s'en préoccuper après, mais dès maintenant. Comme nous l'avons abordé le mois dernier avec les réalités alternatives, de l'Homme augmenté et bionique, il y a une dimension philosophique, morale, éthique et sociétale. Ce n'est pas la première fois que j'évoque ce sujet dans Programmez ! mais je pense que désormais, le débat ne peut plus être occulté. Il doit être clairement abordé par les gouvernants, la société, les industriels, les utilisateurs (= nous).

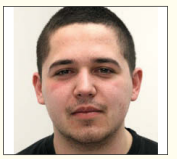
Sommes-nous prêts à ces débats ? Non. Regardez la rapide explosion et évolution de l'impression 3D. En France, on en parle encore comme d'une curiosité, souvent limitée à un cercle assez restreint (même si cette perception évolue). Mais la révolution qu'elle préfigure est colossale et va bousculer tout un pan de l'économie traditionnelle. Et si cette économie ne prend pas en compte l'impression 3D, les conséquences seront négatives (fermetures d'usines, de magasins, chômage). Plus largement le « do it yourself » et le mouvement « maker » vont contribuer à reconsidérer la manière de consommer.

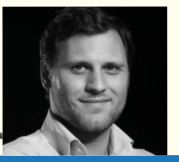
On peut aussi se poser des questions sur les conséquences d'une greffe d'un bras (ou d'une autre partie du corps) robotisé (ou plus intelligent et autonome qu'une prothèse). Augmente-t-on la capacité de l'individu ? Quelles conséquences psychologiques sur l'individu à long terme ? Quelles conséquences sur la santé et son corps ? Quelles conséquences pour les personnes augmentant ces capacités ou utilisant toute extension augmentée sans que cela soit nécessaire ? Le risque n'est-il pas d'aboutir à des êtres techno-dépendants qui ne vivraient et « fonctionneraient » que par les technologies ? Une sorte d'avilissement de l'Homme par la technologie?

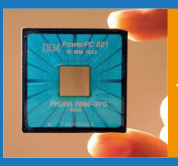
Depuis quelques semaines, on parle de robots et notamment de ces robots qui seront des employés à la place d'humains, qui seront capables de nous servir, de prendre des commandes, d'être autonomes, avec un minimum d'intelligence et de réflexion. Rappelons-nous de la définition du mot robot : robot vient de robota, mot tchèque inventé par Karel Capek signifiant « corvée », « travail », « servage ». Ce mot apparaît dans sa pièce de théâtre « R.U.R. » en 1920. Il faut comprendre « robot/robota » comme une force de travail. Dans cette première apparition, robot n'a pas le sens qu'on lui donne maintenant; il s'agit d'une personne artificielle non pas mécanique, mais chimique.

Aujourd'hui, le robot a une intelligence peu ou pas autonome et une réflexion réduite ou inexistante. De plus, elle dépend de son programme qui reste le plus souvent constant. Demain, la création des vrais cerveaux artificiels, ou la possibilité de transférer le « contenu » de son cerveau sous une forme numérique, a le potentiel de fournir aux robots une intelligence humaine. Les robots auront-ils une âme et une conscience ?





À LIRE DANS LE PROCHAIN NUMÉRO n°181 en kiosque le 29 Décembre 2014

Xamarin
Développer des applications mobiles multiplateforme Android, iOS, Windows Phone

Parallèle / HPC
- Comment et pourquoi utiliser la programmation parallèle ? Les outils, les frameworks, les bonnes pratiques
- Exascale : comment exploiter 100 millions de coeurs processeurs ?

Scala
La puissance de l'objet et du fonctionnel au coeur de Java

Internet : familles et métiers de l'Internet

Petite infographie du Ministère de l'économie, de l'industrie et du numérique.

OpenStack Summit Paris :

+5000 personnes

PHP Forum 2014

+400 personnes

Apple Watch

réveil prévu pour le printemps 2015...

Microsoft Band :

disponible aux USA, aucune date pour la France.

Fire Phone

le flop smartphone 2014 mais Amazon dit : "encore"

Le Jedisme

est-il une religion ? L'Empire ne fait aucun commentaire

OS Mobile :

Android, iOS écrasent tout

Bon ok, Windows Phone est la 3e plate-forme mobile. Mais les rapports et chiffres sont parfois cruels. Windows Phone perd des points de marché ou stagne ailleurs, pendant que Android et iOS continuent à dominer le monde mobile.

L'un des rares marchés où WP dépasse iOS est en Italie. Sur le marché américain, le constat est amer et même inquiétant : à peine 4 % !

Preuve que malgré tous les efforts, Microsoft n'a pas su imposer sa plate-forme. Et les nouveaux Android et iOS risquent de faire très mal.

Le marché WP stagne en France et régresse en Angleterre.

En Chine, marché clé du mobile, Windows Phone devient inexistant et chute largement en un an.

	Android	iOS	Windows Phone
USA	61,8	32,6	4,3
Chine	83,4	15,2	0,4
Japon	64,5	31,3	0,9
France	72,7	15,4	10,6
Allemagne	79,2	11,8	7,1
Angleterre	58,2	31	9,6
Espagne	90,4	6,3	3
Italie	71,8	10,4	15,2

Chiffres Kantar, pour septembre 2014. En %

Pour Strategy Analytics, au niveau mondial (au 3e trimestre 2014), Android était 1er avec 83,6 % du marché, iOS suit à 12,3 % et Microsoft 3,3 %.

Microsoft

ferme son laboratoire de recherche, Microsoft Research Silicon Valley

A quand de vrais ordinateurs sur processeurs ARM ?

LibreSSL,

alternative à OpenSSL

HTML 5

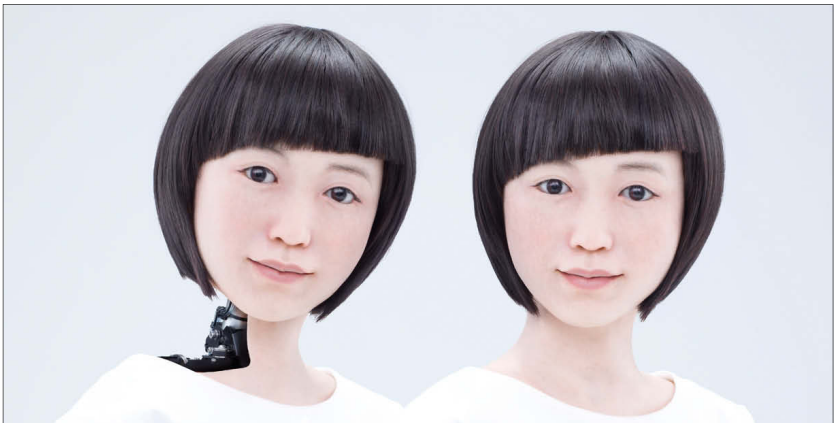
devient officiellement une recommandation du W3C.

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.

Octobre 2014	Octobre 2013	Tendances	Langage	%	Evolution (en %)
1	1	↑	C	17,655	+0,41
2	2	↓	Java	13,506	-2,6
3	3	↑	Objective-C	10,096	+1,1
4	4	↓	C+ +	4,868	-3,8
5	6	↓	C#	4,748	-0,97
6	7	↓	basic	3,507	-1,31
7	5	↓	PHP	2,942	-3,15
8	8	↓	Python	2,333	-0,77
9	12	↑	Perl	2,116	+0,51
10	9	↓	Transact-SQL	2,102	-0,52

Pas de changement pour le trio de tête : Le trio reste immuable, C, Java et Objective-C. Plus loin, Dart apparaît désormais à la 17e place et Swift à la 19e. JavaScript sort du top10.



Les robots, les nouveaux salariés ?

Les commentaires et études se multiplient pour dire que les robots remplaceront les salariés humains dans de nombreux domaines. Info ou intox ? Il faut rester très prudents sur ces études.

Les éléments

1 Une étude de l'université d'Oxford pointe du doigt que + 10 millions de postes salariés pourraient être remplacés par des robots, ordinateurs, machines, d'ici 2034 en Angleterre.

2 Une étude du cabinet Roland Berger évoque la possibilité que 20 % des tâches soient réalisées par des robots, des machines, soit l'équivalent de 3 millions d'emplois. Cela concerne la classe moyenne et les conséquences de la transformation numérique.

Le cabinet Roland Berger évoque plusieurs impacts numériques et technologiques sur l'emploi :

	cabinet	notre commentaire
Big Data	Impact très fort sur l'emploi. Automatisation des tâches.	Le Big Data n'a pas pour nous un impact sur l'emploi. Pour le moment cela concerne une poignée de postes dont les Data Scientist. En revanche, cela aura des conséquences sur les équipes marketing (organisation, optimisation)
Robotique avancée	Très fort sur les tâches physiques	C'est une évolution de la robotique actuelle notamment en industrie. Ce sera l'automatisation des chaînes futures et actuelles.
Véhicule autonome	Fort sur les transports.	L'impact est difficile à déterminer. Les métiers vont changer, de nouveaux vont apparaître. L'impact sera-t-il sur le transport en commun ou sur l'industrie automobile ?
Objets connectés	Modéré	L'objet connecté peut remplacer des emplois. Actuellement cependant, il s'agit surtout d'un complément aux utilisateurs, à la vie quotidienne. Peut créer de nouvelles sociétés (construction, maintenance, etc.).
Internet mobile	Limité. distributeur	Le eCommerce mobile a déjà un impact fort. Les magasins et distributeurs qui n'apporteront pas de valeurs ajoutées ou de vrais services joueront leur survie.
Cloud	Très limité / SI	Peut créer des emplois spécialisés mais attention aux employés des départements informatiques et prestataires extérieurs qui sont / seront impactés par des mutations et licenciements.

Lien étude : http://www.rolandberger.fr/media/pdf/Roland_Berger_TAB_Transformation_Digitale-20141030.pdf

Dès les années 1980, la polémique avait eu lieu en France, et ailleurs, sur les chaînes robotisées dans l'industrie automobile. Le mouvement s'accélère et concerne d'autres industries mais ce n'est pas spécifique à la robotique.

Cependant, il y a la robotique qui va aider le salarié, la personne et la « robotique salariée » remplaçant l'humain, purement et simplement. Et là nous ne parlons plus de l'industrie mais dans des magasins, des restaurants, etc. Bref, une nouvelle robotique autonome et capable de remplir des fonctions de salariées. Ces postes ne sont pas forcément tenus par ce que l'on appelle la classe moyenne. Ainsi, des robots serveurs (apparence et fonctions encore basiques) sont déployés dans quelques restaurants en Chine et Japon. Nous sommes dans une catégorie très particulière de robots : les androïdes. Exemple : des robots guides dans des musées (modèle : Kodomoroid, Japon).

agenda

Décembre

Tour de France PC Soft 2014

PC Soft dévoilera les versions 20 de WinDev, WebDev et WinDev Mobile durant son tour de France annuel. L'événement se déroule l'après midi, à partir de 13h45. Les dates de décembre : Strasbourg (2), Genève (3). Toutes les informations sur http://windev20tdf.fr

Sessions de décembre

Zenika vous propose pour décembre une sélection de sessions et événements techniques suivants :

- Jeudi 4 décembre à partir de 19h : NightClazz Java 8 avancée. Inscriptions à venir. Organisé par Zenika.
- 4-5/12 : Dockercon14 Amsterdam - Zenika sponsorise cette conférence. Informations sur http://europe.dockercon.com/
- Jeudi 11 décembre à partir de 9h : Club(21 organisé par Rupture(21 sur la transformation d'organisations agiles. Inscriptions à venir : http://rupture21.com/club21/
- 26-27 mars 2015 : NoSQL Matters Paris - Informations sur https://2015.nosql-matters.org/par/

Communauté Paris DevOps

A Paris, la communauté DevOps est très active. Chaque mois, elle se rassemble durant un meetup. Occasion de parler DevOps, expérience, bonnes pratiques, usages, etc. Pour en savoir plus : http://parisdevops.fr

Actus des communautés Java

ParisJUG annonce :

- une soirée performance Java le 9 décembre,
- une soirée Young Blood II le 12 janvier 2015
- une soirée Cassandra le 10 février 2015

Site : http://www.parisjug.org/xwiki/bin/view/Meeting/Next

Riviera JUG annonce pour le printemps 2015 son événement Riviera Dev ! A suivre ici : http://rivieradev.fr

La nuit de L'informatique : les 4 et 5 décembre

Du 04 au 05 décembre, à Toulouse ou ailleurs (32 sites) : 3000 étudiants, 600 personnels et partenaires industriels (bénévoles) issus de 80 écoles et universités pour répondre à plus de 54 défis nationaux dotés et lancés sur une thématique particulière : c'est la Nuit de l'Informatique bien sûr !

Contact : nuitinfo-contact@univ-tlse3.fr

Infos : http://www.nuitdelinfo.com

https://www.facebook.com/groups/nuitdelinfo2014/

Intel fait le bonheur des développeurs d'objets connectés

Un objet connecté, ou l'Internet des objets (IoT), se compose de deux éléments fondamentaux : du matériel (board, shield, capteurs...) et de logiciels (firmware, système, logiciels et services logiciels).

Intel propose pour la partie matérielle deux kits : Galileo gen2 (+ Grove Starter kit, kit de capteurs), Edison (board miniature basée sur un SoC). Pour le développement des IoT et des applications embarquées, Intel propose plusieurs IDE. Chaque IDE cible une catégorie de développeurs et de développement.
Galileo peut se comparer à Arduino dans son fonctionnement et son extensibilité. Galileo partage de nombreux points communs avec Arduino, mais ce n'est pas une plateforme Arduino bis. Galileo a son écosystème, sa communauté. Edison est véritablement taillé pour l'informatique Wearable, les IoT. D'une taille minuscule, Edison fonctionne sur un Soc Atom complet et permet de créer des objets connectés autonomes et très petits.

Les IDE : faites votre choix !

Arduino IDE

Dans l'univers Arduino, les développeurs vont la plupart du temps utiliser Arduino IDE. Léger et très sobre, Arduino IDE est le parfait outil pour démarrer en douceur avec Galileo et Arduino. Il est disponible sur Linux, OS X, Windows. Il permet de coder rapidement ses sketches et d'accéder à de nombreux exemples de code. Il permet de compiler et transférer à la carte. On dispose aussi d'un moniteur qui pourra, par exemple, sortir les données du capteur de température. Il permet de mettre à jour le firmware de sa board.

Intel XDK IoT Edition

XDK IoT Edition est le dernier environnement intégré d'Intel. La première version a été dévoilée fin septembre dernier. Disponible sur Ubuntu, OS X et Windows, cet IDE permet de développer des projets IoT avec les langages et technologies Web : HTML, JavaScript, Node.JS. On peut ainsi coder, déployer et exécuter du code JavaScript sur sa carte Galileo ou Edison. C'est une réelle nouveauté par rapport aux développements natifs. L'outil est régulièrement mis à jour. À l'heure de la rédaction de cet article, nous étions en version 75. Fonctionnalité importante, vous pouvez faire du remote debug de votre application JavaScript.
Si vous ne maîtrisez pas C ou C++, XDK IoT Edition est une excellente alternative pour vos objets connectés. Pour pouvoir déployer et exécuter ces applications CDK, il faut que le système Yocto (distribution Linux) installé sur la board (Galileo ou Edison) possède les librairies Node.JS propres à XDK. En principe, elles sont installées par défaut, mais elles peuvent nécessiter une mise à jour.
L'autre intérêt de XDK IoT Edition est de pouvoir créer rapidement une application compagnon à votre projet IoT. Cela signifie que vous créez une application (smartphone, tablette, desktop) pour utiliser, contrôler, interagir avec votre objet connecté. Cette app utilise les librairies Cordova pour l'interface graphique. Pour accéder aux services Cloud IoT, des API sont disponibles et utilisables par l'application cliente.

Intel System Studio

Intel propose une suite professionnelle pour les développeurs C, C++ : System Studio. Il s'agit d'une suite de développement incluant (selon l'édition) : le compilateur C++, les librairies mathématiques, Vtune, Threading Building Block, Inspector for Systems... Ces outils permettent d'aller très loin dans l'optimisation, les performances et la disponibilité des applications notamment en environnements embarqués comme Edison.

Intel IoT Cloud Analytics

Un des défis des objets connectés/IoT est la collecte et l'analyse des informations et données issues des capteurs, de l'objet en lui-même. Ces services Cloud permettent de traiter et stocker ces données. Intel IoT Analytics s'utilise avec Galileo et Edison. Pour assurer l'interconnexion du matériel connecté aux services Cloud, un agent (en JavaScript) doit être déployé sur l'IoT. Un tableau de bord sur le portail IoT Analytics permet d'activer le service.

Ressources

Galileo Software Downloads : vous y trouverez tous les outils, images systèmes/firmware, Arduino IDE pour Galileo. https://communities.intel.com/docs/DOC-22226

Edison Software Downloads : page dédiée aux outils et logiciels Edison. Vous y trouverez Arduino IDE, Yocto Edison, les différents SDK, les pilotes Windows. https://communities.intel.com/docs/DOC-23242

Page générale sur Intel XDK IoT Edition : https://software.intel.com/en-us/html5/xdk-iot

Page générale sur les solutions IoT (zone développeur) : https://software.intel.com/en-us/iot/downloads

C/C++ : https://software.intel.com/en-us/iot-c-eclipse
JavaScript : https://software.intel.com/en-us/installation-for-javascript-intel-xdk-galileo-and-edison
Visual Programming : https://software.intel.com/en-us/getting-started-for-visual-programming-wyliodrin-galileo
Arduino : https://communities.intel.com/docs/DOC-23147
Cloud Analytics : https://software.intel.com/en-us/iot-cloud-analytics-guide

La majorité des ressources techniques est en Anglais.

En résumé

	Arduino développeur	Programmation visuelle	Développeur JavaScript	Développeur c, c++	Wind river (galileo)
Type de développeur	Maker	Débutant	Intermédiaire	Avancé	Avancé, professionnel
Système	Yocto	Yocto	Yocto	Yocto	VxWorks
IDE	Arduino IDE	Wyliodrin	XDK	Intel System Studio Eclipse	WR Eclipse
Langages	Sketch, C++	Python	JavaScript	C, C++	C, C++
Outils/librairies	Arduino	Wyliodrin	XDK	ISS	Work Bench /ISS
Cloud	IoT Cloud Analytics	IoT Cloud Analytics Widget	IoT Cloud Analytics Mashery Tiers	IoT Cloud Analytics Mashery Tiers	WR Cloud

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

écoconception web

4e partie

Code client : éviter les interaction inutiles

Malgré les progrès des compilateurs JavaScript, l'usage parcimonieux d'un code optimisé reste la meilleure garantie de fluidité pour l'interface utilisateur.

Frédéric Bordage,
expert écoconception logicielle, GreenIT.fr, @greenit
Jérémy Chatard,
directeur technique de Breek, @breekfr

Les interfaces graphiques des sites Web sont restées statiques de 1994 à 1998. Elles n'incorporaient alors que des listes déroulantes et des formulaires. Pour aller plus loin, il fallait utiliser des ActiveX (Microsoft), des applets Java ou des composants Flash (Macromedia / Adobe). Ce n'est qu'avec l'avènement de HTML 4.2, des Cascading Style Sheet (CSS) 2.0, et des prémices d'ECMAscript fin 1998 que l'architecture Dynamic HTML (DHTML) a permis de créer des interfacesWeb vraiment interactives. HTML5 a ensuite poussé cette logique à l'extrême en transformant les navigateurs en socles d'exécution très évolués, capables d'effectuer des traitements complexes, de dialoguer avec des backends via l'architecture AJAX et de stocker des données relationnelles localement. Ces dernières années, nombre de développeurs ont abusé de ce potentiel en créant des interfaces utilisateurs gavées de Javascript, souvent sans aucune justification. C'est notamment le cas d'un fameux site de transport de voyageurs français. Résultat, la consultation du site devient tellement lente qu'il faut parfois remplacer son ordinateur par un plus puissant pour y accéder. L'écoconception Web vise à éviter ce genre de situation en s'attachant à n'utiliser des technologies propriétaires et / ou des traitements javascript complexes que lorsqu'ils sont absolument nécessaires.

Limiter le recours aux plugins #24

Eviter d'utiliser les plug-in tels que Flash, applet Java, Silverlight, etc. Préférer les technologies standard telles que HTML 5, ECMA Script, etc. Par exemple, pour réaliser un back office basé sur du drag & drop, préférer l'utilisation de JQuery à Flash.

Ecrivez un code JavaScript efficace #25, #26, #33

Evitez d'utiliser eval(). Cette fonction évalue et exécute une chaîne de caractères. Cela amène donc une recompilation et une réinterprétation du code lors de l'exécution. Ceci a pour inconvénient (outre les risques de sécurité) d'empêcher la mise en cache par le navigateur du code généré, car la chaîne de caractère peut contenir des variables qui évoluent au cours de l'exécution du programme. Dans l'exemple ci-dessous, le code contenu dans la variable "code" ne peut être mis en cache par le navigateur. Par exemple, ne pas écrire

```
function anyRoutine(index) {
  // Do something here
  console.log(index);
}

var foo = 5,
  code = 'for (var i = foo; i > 0; i--) { anyRoutine(i); };';

eval(code);
```

Donnez des fonctions en paramètre à setTimeout() et setInterval() plutôt que des strings. Ces deux fonctions, peuvent prendre des fonctions - ou - des chaînes de caractères en argument. Outre le fait qu'il est déconseillé d'utiliser eval() et donc de passer des chaînes de caractères, si l'argument passé est sous forme de chaîne, le moteur d'interprétation va devoir évaluer cette chaîne pour la transformer en code. Ce qui n'est pas le cas lorsque l'argument est une fonction ou référence à une fonction. Ceci permet d'économiser de la CPU. Par exemple, ne pas écrire :

```
var timeoutID = setTimeout('console.log(""Hello Dolly"");', 1000);
```

Mais :

```
var uneFonction = function() {
  console.log('Hello Dolly');
}
var timeoutID = setTimeout(uneFonction, 1000);
```

Vous pouvez aussi mettre en cache les objets souvent accédés en JavaScript. L'accès au DOM est coûteux en termes de ressources CPU. Lorsque vous utilisez plusieurs fois le même élément du DOM depuis JavaScript, stockez la référence de l'élément dans une variable afin de ne pas re-parcourir le DOM pour ce même élément. Par exemple, ne pas écrire :

```
document.getElementById('menu').property1 = 'foo';
document.getElementById('menu').property2 = 'bar';
```

Mais :

```
$menu = document.getElementById('menu');
$menu.property1 = 'foo';
$menu.property2 = 'bar';
```

Eviter les variables inutiles #37, #39

Privilégier les variables locales. En JavaScript lorsque l'on fait appel à une variable globale, le moteur d'interprétation doit vérifier qu'elle existe dans le scope actuel, dans celui du dessus..., que la variable dispose d'une valeur, etc. Il est souvent envisageable de passer les variables utiles en arguments des routines, les rendant locales. Ceci permet d'éviter toutes ces vérifications et donc d'économiser du temps de calcul. Par exemple, ne pas écrire :

```
var aGlobal = new String('Hello Dolly');
function globalLength() {
  length = aGlobal.length;
  console.log(length);
}
globalLength();
```

Mais :

```
var aGlobal = new String('Hello Dolly');

function someVarLength(str) {
  length = str.length;
  console.log(length);
}

someVarLength(aGlobal);
```

Toujours au niveau des variables, utiliser la concaténation de chaînes de façon optimale. JavaScript génère en tâche de fond des variables temporaires lors des concaténations de variables. Certaines syntaxes


NOUVELLE VERSION
920
NOUVEAUTÉS
Java
HTML
iOS
WINDEV : Logiciel professionnel de développement multi-plateformes: développez une application, recompilez-la pour tous les systèmes
Développements natifs
WINDEV
PLATEFORME INTÉGRÉE DE DÉVELOPPEMENT
WINDOWS, .NET, RAD JAVA, ...
20
PC SOFT
DÉVELOPPEZ 10 FOIS PLUS VITE
Fournisseur Officiel de la Préparation Olympique
20
DÉVELOPPEZ 10 FOIS PLUS VITE
www.pcsoft.fr
Des centaines de références sur le site
Version 20 annoncée. Logiciel professionnel. Document non contractuel

sont plus efficaces que d'autres et permettent donc de réduire la consommation de mémoire vive. Par exemple, remplacer :

```
a += 'x' + 'y';
```

Par :

```
a += 'x';
a += 'y';
```

Méfiez-vous du DOM #32, #42, #43, #44

L'accès au DOM via JavaScript est une procédure lourde qui consomme du temps CPU de façon significative surtout sur les applications utilisant beaucoup JavaScript. Réduisez autant que possible ces accès et évitez autant que possible de modifier le DOM lorsque vous le traversez. Modifier le DOM lorsqu'on le traverse peut générer des situations où la boucle devient très gourmande en ressources. Il est notamment assez courant de générer une boucle infinie lorsque l'on ajoute des éléments au DOM au fur et à mesure qu'on le traverse. Eviter par exemple :

```
<script>
$('a.extlink').each(function(el) {
  $(el).attr('rel', 'external nofollow');
});
</script>
```

Pour éviter les cycles CPU inutiles, il faut aussi réduire au maximum le repaint (appearence) et le reflow (layout). Le repaint est le changement d'apparence d'un élément du DOM. Le reflow est le changement / recalcul de la position des éléments dans le DOM. Ces deux opérations sont coûteuses en ressources. Il faut donc éviter de les déclencher. Par exemple, pour éviter les repaints, il faut éviter de changer les propriétés stylistiques d'un élément (couleur de fond, etc.). Pour éviter les reflows, il faut minimiser les changements de propriétés de positions, dimensions, de type de positionnement, de contenu, etc.
Vous pouvez aussi rendre les éléments du DOM invisibles lors de leur modification. Lorsqu'un élément du DOM doit être modifié par plusieurs propriétés, chaque changement de style ou de contenu va générer un repaint ou un reflow. Il est généralement plus économe de rendre l'élément invisible (propriété "display" à "none") (1 reflow) ; puis de modifier toutes les propriétés de l'élément et rendre l'élément à nouveau visible (1 reflow). Soit 2 reflows au maximum.
Une autre approche, lorsqu'elle est possible, consiste à modifier plusieurs propriétés CSS en 1 seule fois pour limiter le nombre de repaint/reflow en utilisant l'ajout/suppression de classes CSS. Ainsi, un seul changement permet de modifier plusieurs propriétés tout en ne générant qu'un repaint et/ou reflow. Ne pas faire :

```
<script>
$el.bind('error', function () {
  $el.css('color', 'red');
  $el.css('font-weight', 'bold');
});
$el.bind('running'), function () {
  $el.css('color', 'inherit');
  $el.css('font-weight', 'inherit');
};
</script>
```

Préférez plutôt :

```
<style>
.in-error {
  color: red;
  font-weight: bold;
}
</style>

<script>
$el.bind('error', function () {
  $el.addClass('in-error');
});
$el.bind('running'), function () {
  $el.removeClass('in-error');
};
</script>
```

Utilisez la délégation d'évènements #41

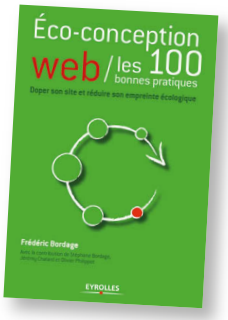
La délégation des évènements permet de ne pas surcharger la mémoire du navigateur et d'instancier un seul écouteur pour plusieurs éléments du DOM. La documentation de Mozilla vous aidera à mettre en place ce type de délégation. Par exemple :

```
<htl>
 <head>
   <title>DOM Event Example</title>
   <style type=""text/css"">
     #t { border: 1px solid red }
     #t1 { background-color: pink; }
   </style>
   <script type=""text/javascript"">

   // Function to change the content of t2
   function modifyText(new_text) {
     var t2 = document.getElementById(""t2"");
     t2.firstChild.nodeValue = new_text;
   }

   // Function to add event listener to t
   function load() {
     var el = document.getElementById(""t"");
     el.addEventListener(""click"", function(){modifyText(""four"")}, false);
   }

   </script>
 </head>
 <body onload=""load();"">
 <table id=""t"">
   <tr><td id=""t1"">one</td></tr>
   <tr><td id=""t2"">two</td></tr>
 </table>
 </body>
 </html>
```

Validez votre code JavaScript #35

Enfin, pour que votre code Javascript soit efficace, vous devez impérativement le valider avec JSLint. Cet outil vérifie que la syntaxe JavaScript utilisée sera comprise par tous les navigateurs. Le code obtenu respecte des contraintes syntaxiques permettant aux interpréteurs d'exécuter le code plus facilement et donc rapidement. La CPU est donc sollicitée moins longtemps.

Retrouver ces bonnes pratiques dans le livre :
Eco-Conception web
Editions Eyrolles



Métiers du test : situation actuelle et perspectives

Le CIGREF a publié fin octobre dernier sa nomenclature des métiers des DSI. Le « Testeur » n'a droit qu'à une seule ligne, malgré les nombreux profils de testeurs existants. Je vous propose de détailler un peu plus ce métier d'avenir.

Bernard Homès
Consultant Senior TESSCO sas Fondateur et ex-président du Comité Français des Tests Logiciels
bhomes@tesscogroup.com

Avec plus de 33 ans d'expérience dans le domaine du développement et du test de logiciels, il est Consultant Senior pour de grands clients, après plusieurs postes à l'international. Il de nombreuses certifications en test (CMAP, ISTQB CTFL, CTAL-TM, CTAL-TA, CTAL-TTA) et en gestion des exigences (IREB et REQB). Il est auteur de 2 ouvrages sur les tests de logiciels.

Valeur ajoutée

Un testeur identifie les anomalies avant la livraison. Il est entre les clients (pour le fonctionnel) et les développeurs (pour corriger des anomalies), et fournit aussi d'autres services :

- Par l'analyse des exigences, le testeur identifie des ambigüités, et évite l'introduction de défauts. S'il est impliqué tôt sur le projet, cela réduit les coûts et les délais de réalisation,
- En étudiant la maintenabilité du code et la qualité de la documentation, il aide à anticiper les coûts de maintenance,
- En analysant les anomalies et leurs corrections, le testeur identifie les manques de maturité et les opportunités d'amélioration des processus de conception et des activités de test,
- Via la fourniture de données statistiques (p.ex. pourcentage de couverture des exigences, taux d'efficacité des processus de test, etc.) le test permet d'identifier des tendances et de prendre (à temps) des décisions sur le projet et le produit.

Le testeur n'est pas seul responsable de la qualité du logiciel : tous les acteurs du projet (y compris les développeurs) sont responsables – chacun à son niveau – de cette qualité.

Un métier attrayant

Contrairement aux idées reçues, les métiers du test sont passionnants et pleins d'avenir.

C'est le défi permanent d'identifier dans de nouvelles applications des types d'anomalies différents, puis d'informer – sur le niveau de qualité fonctionnelle et technique – les acteurs du projet. Il y a simultanément :

- Une composante technique, les testeurs et les développeurs se focalisant sur de nouvelles technologies. Le testeur est obligé de se tenir informé des technologies émergentes. Dans de nombreux cas le "développement" est présent via l'automatisation des cas de test,
- Une composante utilisabilité et relations humaines, le testeur étant vu comme « la voix du client » sur le projet, et s'interfaçant avec toutes les parties prenantes du projet. Nous avons donc une composante relationnelle importante,
- Une composante de gestion : le test est situé sur le chemin critique du projet, tout retard impactera la date de livraison, il est important d'être à la fois efficace (trouver le plus d'informations possible) et rentable (utiliser le moins de ressources possible), un défi pour tout manager.

Le test propose, dans des environnements variés – allant des mainframes et du "Big Data" aux plateformes mobiles et tablettes – de nombreux défis techniques et méthodologiques.

Des profils divers

Comme mentionné précédemment, il y a plus d'un métier de testeur. Nous pouvons les classer selon l'activité principale effectuée :

- Technicien de test, exécutant des tests définis par d'autres. Avec ou sans certification ISTQB niveau fondation, il fait du test fonctionnel basé sur son expérience,
- Analyste de test (fonctionnel) : analyse les besoins et conçoit des tests fonctionnels à exécuter, et peut encadrer un ou plus testeurs. Expérimenté, il a souvent une certification ISTQB niveau avancé d'analyste de test,
- Analyste technique de test : conçoit des tests de performances ou de sécurité. Expérimenté, il a souvent une certification ISTQB niveau avancé d'analyste technique de test, et une spécialisation sur le(s) outil(s) utilisé(s),
- Automatiseur de test : traduit des tests en scripts de tests automatisés. Possède souvent une spécialisation sur le(s) outil(s) qu'il utilise,
- Chef de projet de test : encadre des testeurs et dirige un projet de test, rapportant aux parties prenantes. Possède une certification de Test Manager ISTQB niveau avancé,
- Administrateur d'environnement de test : gère et met en place les matériels de test, les sauvegardes et restaurations des données de test, des environnements, etc.,
- Consultant en test, fournit du conseil spécialisé en tests, (p.ex. utilisabilité, sécurité, performances, etc.) et possède une ou plusieurs certifications (p.ex. ISTQB, CMAP, etc.),
- Testeur dans un projet Agile : au sein de projets Agiles, il couvre plusieurs des activités décrites ci-dessus, avec un mode de gestion différent et complexe,
- Responsable qualité, processus et méthodes : en charge des activités "qualité" transverses, ce qui peut inclure la qualité logicielle et les tests.

Les développeurs sont aussi testeurs, sans le savoir : la relecture du code est du "test statique", la première exécution du code du "test unitaire", etc.

Qualités Techniques et Relationnelles

Les testeurs viennent de divers horizons :

- Du "métier", (p.ex. utilisateurs), et mettent à profit leur expérience du domaine ou du produit. Leur vision porte sur les fonctionnalités du produit et sur son utilisabilité,
- Du "support", (p.ex. hotline et support utilisateur), et utilisent leurs connaissances des types d'erreurs remontées pour améliorer le produit,
- De la "technique", (p.ex. développeurs), et ils se focalisent plus sur des aspects maintenabilité, sécurité, performances, etc. Ils font de très bons Analystes techniques de test, de bons automatiseurs de test, puis d'excellents Test Managers et consultants,

Quelle que soit l'origine du testeur, certaines qualités sont importantes :

- Un intérêt pour la qualité des logiciels et un souci à la fois de minutie et une vue d'ensemble,
- Des compétences relationnelles et de communication : il faut s'interfacer avec des personnes techniques (développeurs) et des utilisateurs sur des sujets parfois épineux,
- Des compétences "métier" ou "business" portant sur le domaine applicatif,
- Des compétences "informatiques" élargies, les défauts provenant de diverses causes (logicielles, matérielles, réseaux, OS, etc.),
- Une curiosité et une volonté d'apprendre, pour maitriser de nouvelles techniques, et être efficaces et rentables dans l'identification des défauts,
- Une capacité de travail importante, à la fois individuel et en équipe, doublée d'une résistance au stress (les pressions sur les délais sont fréquentes).

Un développeur possède déjà plusieurs de ces compétences, donc une évolution vers le domaine du test est facile.

Salaires moyens et débouchés

La grille salariale est plus basse que pour des développeurs, mais ceci est compensé par un travail attrayant et des perspectives de carrière accrues : voir tableau ci-contre.

Les débouchés sont nombreux, à la fois dans et en dehors du métier du test ou de l'informatique. Certains testeurs sont devenus CTO, d'autres Responsables de Départements métier, quelques-uns se sont tournés vers l'éducation, d'autres enfin se sont spécialisés et sillonnent le monde en donnant des conférences et du conseil. La triple compétence "Métier" + "Qualité" + "Logiciel" est de plus en plus utile dans les organisations, car elle met la satisfaction des utilisateurs et des clients en contact avec la DSI et les technologies.

Intitulé du poste	Expérience	Salaires
Technicien en test ou Testeur fonctionnel	< 2 ans	De 33 à 36 K€/a
Analyste de test fonctionnel	3+ ans	De 36 à 45 K€/a
Analyste technique de test (non fonctionnel)	3+ ans	De 42 à 55 K€/a
Automatiseur de test	2+ ans	De 38 à 48 K€/a
Test manager	5+ ans	De 45 à 55 K€/a
Responsable d'environnement de test	2+ ans	De 38 à 45 K€/a
Consultant en test	5+ ans	De 48 à 60 K€/a
Testeur sur projet Agile	2+ ans	De 33 à 38 K€/a
Responsable qualité, processus et méthodes	5+ ans	De 48 à 58 K€/a

En milliers d'€ / annuel.

Formations

On ne nait pas testeur, on le devient. S'il est possible d'étudier seul – à partir de livres (en anglais, ou en français(1)), d'articles, de webinars(2) et autres sources trouvées via des moteurs de recherche – ce mode d'apprentissage risque des lacunes dans certains domaines.

Suivre des formations basées sur un référentiel reconnu (p.ex. certifications ISTQB et/ou CMAP), permet de s'assurer de la couverture de tous les domaines principaux.

De nombreuses formations existent dans le domaine des tests de logiciels :

- Des formations spécifiques selon les niveaux de test (fondation, avancé ou expert), permettent d'atteindre un niveau (p.ex. ISTQB ou CMAP) et de passer la certification associée. Une certification est un différenciateur important sur un CV, et valide l'acquisition d'un niveau de connaissance.
- Des formations sur des domaines amont comme la gestion des exigences (référentiel IREB ou REQB) permettent de maitriser et limiter les ambigüités ou les lacunes des exigences.
- Des formations en gestion de projets de test sensibiliseront aux aspects d'efficacité et de rentabilité des activités de test, au reporting et à l'interprétation des mesures, ainsi qu'aux interfaces avec les parties prenantes.
- Des formations focalisées sur les méthodes permettront une meilleure efficacité en recherche d'anomalies.
- Des formations focalisées sur les outils (commerciaux ou open-source) permettent d'optimiser leur utilisation, et par exemple de relier des outils existants ou de valider les résultats de tests.
- Certaines formations existent se focalisant sur les plateformes mobiles (p.ex. formations CMAP) et identifient les éléments qu'un testeur devrait connaître sur ces technologies.
- Des formations aux techniques de test permettront d'apprendre ou d'approfondir certaines techniques afin de les utiliser plus efficacement, et ainsi détecter le plus d'anomalies.

Tout développeur tirera des bénéfices importants de ces formations car elles lui permettront d'améliorer son code, et donc éviter l'introduction de défauts qui seront à corriger ensuite.

Les défis des années à venir

Niels Bohr a dit : "les prédictions sont difficiles, surtout concernant le futur". Vu les tendances actuelles, les défis du test dans les années à venir incluront :

- Le "Big Data" et la vérification des résultats des algorithmes de calcul au vu de la multiplicité des données, car ces algorithmes devront être vérifiés sur la base de données sûres, et de telles données sont souvent parsemées de données périmées ou incorrectes. Le défi est de s'assurer de la validité des données remontées, et du respect de la confidentialité des informations sous contrainte des diverses législations en vigueur,
- L'informatique domestique connectée (domotique) et les risques associés à une utilisation non maitrisée par les utilisateurs de telles technologies. L'accès à votre logement, votre véhicule et aux sources de vos données personnelles permettraient à des personnes mal intentionnées, de prendre contrôle de votre existence et de votre identité. Le défi est d'assurer simultanément l'utilisabilité et la capacité de communication de ces technologies d'une part, et d'autre part la sécurité,
- Certaines technologies "assistives" (p.ex. lunettes fournissant des informations contextuelles) et les risques associés à la confidentialité des informations recueillies et/ou affichées. Les défis ici sont à la fois les aspects de confidentialité de la vie privée d'une part, et d'autre part la mise à disposition des informations pour les personnes autorisées,
- Le "Cloud", la globalisation et la sécurisation des informations personnelles ou commerciales qui y sont présentes. Comme dans les cas précédents, les aspects de sécurité des informations sont les premiers à sauter à l'esprit, mais les aspects de performances et de mobilité sont aussi à prendre en compte,
- Le nomadisme, avec les tablettes, smartphones et autres devices mobiles (des lunettes connectées jusqu'aux drones, de votre téléphone portable jusqu'au paiement sans contact), et les interactions qu'ils pourront avoir entre eux. Souvent cela oblige à prévoir des tests supplémentaires (p.ex. mesurer la consommation électrique pour éviter de réduire la durée d'utilisation d'une charge de batterie),
- Les aspects de fonctionnement des applications connectées dans un environnement où les communications sont limitées ou inexistantes. Donc des aspects de fonctionnement dégradé, de sauvegarde en interne de certaines informations, de synchronisations périodiques seront à traiter et à tester.

Ces défis sont déjà ceux des développeurs, et leurs compétences les aideront dans une carrière de testeur. Pour les testeurs, ces défis sont présents et relevés au quotidien.

1) P.Ex : "Les Tests Logiciels : Fondamentaux", éditeur Lavoisier, ISBN: 978-2-7462-3155-9
2) TESSCO sas propose des webinars mensuels gratuits en français, et RBCS-US en propose en anglais.

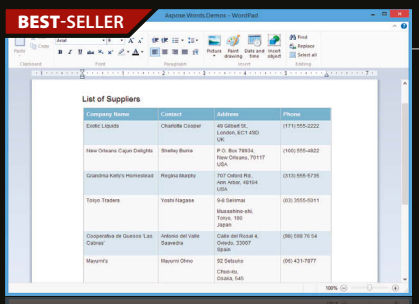
À LIRE DANS LE
PROCHAIN NUMÉRO n°181
en kiosque le
29 Décembre 2014

Xamarin
Développer des applications mobiles multiplateforme Android, iOS, Windows Phone

Parallèle / HPC
Comment et pourquoi utiliser la programmation parallèle ? Les outils, les frameworks, les bonnes pratiques
Exascale : comment exploiter 100 millions de coeurs processeurs ?

Scala
La puissance de l'objet et du fonctionnel au coeur de Java

BEST-SELLER



Aspose.Words for .NET | à partir de € 783

ASPOSE Your File Format APIs

Lisez, modifiez et écrivez des documents Word sans Microsoft Word.

- Création de documents, manipulation du contenu/formatage, puissante capacité de fusion de courrier et exportation en DOC/HTML
- Accès détaillé à tous les éléments d'un document par programmation
- Support les formats de fichiers: DOC, DOCX, WordprocessingML, RTF, HTML, OOXML, OpenDocument, PDF, XPS, EMF et EPUB

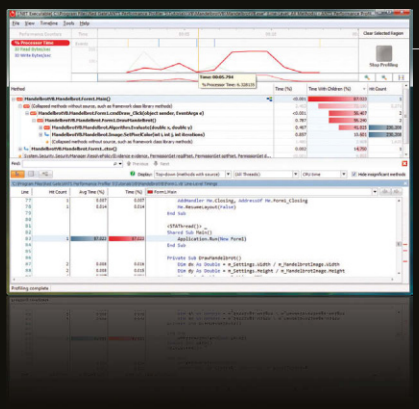
BEST-SELLER

ComponentOne Studio Enterprise 2014 v3 | à partir de € 1 160

ComponentOne a division of GrapeCity

Outils pour le Développeur Professionnel .NET : Windows, HTML5/Web et XAML.

- Contrôles d'interface utilisateur pour .NET, incluant grilles, graphiques, rapports et planificateurs
- Gamme HTML5/JavaScript pour le développement d'applications d'entreprise
- Thèmes intégrés/concepteurs pour la création de thèmes/styles personnalisés
- 40+ composants pour Windows 8.1 & Windows Phone 8.1 et support pour Universal Windows
- Supporte toutes les plateformes de Microsoft: Visual Studio 2013, ASP.NET, WinForms, WPF, etc



Red Gate .NET Developer Bundle | à partir de € 760

redgate®

Éradiquez et corrigez le code lent, identifiez le code .NET bogué et comprenez les raisons.

- Bénéficiez d'une vue d'ensemble des performances de vos applications et identifiez les goulots d'étranglement, dans le code ou la base de données
- Trouvez rapidement les fuites de mémoire et optimisez la mémoire de vos codes C# et VB.NET
- Comprenez et déboguez le code de tiers, frameworks, composants et bibliothèques inclus
- Standardisez la gestion des performances de votre équipe de développement

BEST-SELLER

DevExpress DXperience | à partir de € 1 176

DevExpress

Tous les outils DevExpress ASP.NET, WinForms, Silverlight, WPF et IDE Productivity en un.

- Abonnement de 12 mois pour tous les produits et mises à jour DevExpress et accès aux versions bêta en développement actif
- Modèles et thèmes d'application intégrés exceptionnels
- Support de nouvelle vue Windows 8 UI et panneaux ancrables tactiles
- Support interface codée pour test environnement utilisateur

Siège social en Europe
ComponentSource
30 Greyfriars Road
Reading
Berkshire
RG1 1PE
Royaume-Uni

Siège social aux États-Unis
ComponentSource
650 Claremore Prof Way
Suite 100
Woodstock
GA 30188-5188
États-Unis

Siège social au Japon
ComponentSource
3F Kojimachi Square Bldg
3-3 Kojimachi Chiyoda-ku
Tokyo
Japon
102-0083

Numero vert:
0800 90 92 62
www.componentsource.com

Visual Studio Community, Visual Studio 2015 preview, .net sur OS X et Linux, meilleur support iOS et Android : Microsoft ouvre les vannes !

Diversité, ouverture, multiplateforme, voilà comment nous pourrions résumer les nombreuses annonces de Microsoft lors de la conférence Connect, il y a quelques jours. « Nous cherchons à répondre (aux attentes) des développeurs non Microsoft. Et de continuer à faire de Visual Studio le meilleur outil de développement » précise Jakob Harttung (directeur du pôle stratégie plateforme et évangélisme technique – Microsoft France).

François Tonic
ftonic@programmez.com

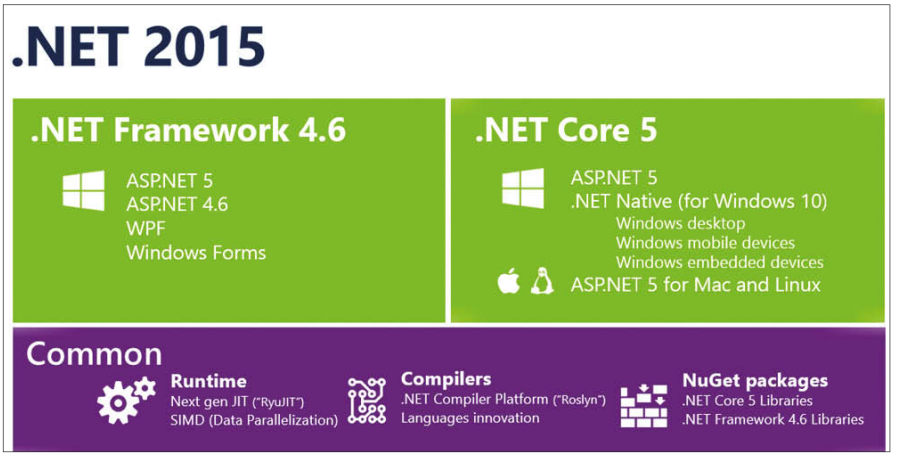
Connect () a donc été l'occasion pour Microsoft de dévoiler de nombreux outils et d'importantes annonces pour le développeur, qu'il soit développeur sur technologies .net/Microsoft ou non.

.net : de l'open source, du Mac et du Linux

Microsoft n'abandonne pas .net. Au contraire, l'éditeur fait un triple mouvement :

- continuer à le faire évoluer
- ouvrir certaines couches et les rendre open source
- porter officiellement certaines portions sur d'autres systèmes

Comme le précise l'article dédié, la disponibilité sur github concerne uniquement les parties .net Core Runtime et certaines librairies. On parle ici des couches nécessaires pour exécuter du code .net (incluant la CLR, le compilateur juste à temps, le ramasse-miettes et les librairies .net Core). Ce sont les couches basses du framework .net. Il s'agit aussi d'ouvrir plusieurs librairies : XML, SIMD, Metadatareader, ImmutableCollections.
Microsoft précise d'ailleurs que les contributions externes seront possibles. Et le mouvement devrait continuer dans les prochains mois. Cela renforce aussi la fondation .net lancée au printemps dernier.
À côté de cela, il y a l'annonce de la portabilité sur OS X et Linux de .net Core. Il s'agit de la brique serveur. Ainsi il sera possible de déployer des applications .net sur de multiples environnements. Cela signifie que l'on pourra déployer des applications ASP.Net 5 sur Windows, Windows Server, Azure, OS X et Linux... Attention : .net ne devient pas multiplateforme, ce portage concerne uniquement certaines zones du framework. Microsoft accentue donc l'ouverture de couches techniques. C'est une bonne nouvelle pour la communauté et les possibilités de ne plus être totalement dépendant de Windows comme plateforme d'exécution.
Miguel de Icaza (projet Mono) a bien accueilli la nouvelle. Et il est déjà prévu de remplacer des pans entiers du code Mono par ces nouveaux codes open source (et amélioration la compatibilité entre les deux piles techniques). Les relations entre Microsoft et Mono n'ont jamais été simples. Mais peu à peu, Mono a su trouver sa place et créer sa communauté. Post complet : http://tirania.org/blog/



Quelques mots sur Visual Studio Community

« Cette édition rejoint notre stratégie d'ouverture. Le développeur profite de toutes les fonctionnalités et notamment de l'extensibilité de l'environnement. » précise Jakob. VS Community s'appuie sur l'édition Visual Studio 2013 Professional.
Cette version cible les développeurs mobiles et multiplateformes, car il est possible de coder pour Windows, Android et iOS, d'utiliser Unity et Apache Cordova. Cette édition peut être utilisée par les développeurs indépendants, à la maison, dans les classes, la recherche, pour des projets open source et en entreprise (il ne faut pas dépasser 250 PC ou 1 million de revenus annuels et uniquement en développement open source).
Il n'est pas certain que les versions Express continuent à exister.

C++ pour le développement multiplateforme

L'annonce n'est pas passée inaperçue. Microsoft pousse C++ sur mobile. Il sera possible de développer des apps universelles pour mobile et pas seulement pour Windows ! Le but est de supporter Android et iOS !
Cette nouveauté arrive avec Visual Studio 2015. Actuellement, le « C++ crossplatform » est valable pour Android et Windows, la compilation iOS arrivera plus tard. Cette belle nouveauté embarque un nouveau type de projet « Native-Activity Application ». Template projet d'une application Android ! On pourra ainsi exploiter l'émulateur Android de VS 2015. C'est donc une 3e possibilité de développement mobile multiplateforme : C++, Cordova et Xamarin.
« C'est une reconnaissance à la diversité (des développements, NDLR). » recadre Jakob. On aurait pu croire que Microsoft empiétait fortement sur Xamarin mais pour le moment, Xamarin est plus complet et plus riche sur le développement natif multiplateforme mobile que l'offre Visual Studio 2015.

Pour en savoir plus : http://goo.gl/fCP1f0

Connect() : Quoi de neuf chez Microsoft ?

Les 12 et 13 Novembre 2014, Microsoft a tenu une conférence, nommée Connect(), au cours de laquelle un grand nombre d'annonces ont été faites, beaucoup d'entre-elles ayant eu un fort impact dans la communauté des développeurs. Faisons un rapide tour d'horizon de ce qui a été annoncé par les grands pontes de Microsoft !

INFINITE SQUARE

Thomas LEBRUN – *Consultant*
http://blogs.infinitesquare.com/b/tom
@thomas_lebrun

L'ouverture du Framework .NET :

L'annonce a fait l'effet d'une bombe lorsque Scott Guthrie l'a annoncé : Microsoft a décidé de rendre Open Source le Framework .NET. Attention, il ne s'agit pas de tout le Framework mais de la partie « serveur » (.NET Core Server) ainsi que des couches « basses » ; le tout sera accessible au travers de GitHub (http://github.com/Microsoft/dotnet & https://github.com/dotnet/corefx).

Dans la continuité de la mise en Open Source de .NET, Microsoft va également mettre à disposition une version de .NET qui puisse être installée sur Linux et Mac OS (en plus de Windows). Les équipes de Microsoft ont en effet travaillé avec Miguel De Icaza et les équipes du projet Mono afin de mettre à disposition une implémentation de .NET pour Windows, Linux et OS X.

Visual Studio Community 2013 :

Visual Studio, l'IDE de Microsoft, existe à l'heure actuelle en différentes version : de la version simple (et gratuite) à la version Ultimate, intégrant toutes les fonctionnalités possibles (Code Lens, IntelliTrace, etc.). L'inconvénient de la version gratuite (appelée « Express »), c'est qu'elle ne pouvait être utilisée que pour développer des applications d'un certain type (Web, Windows, Phone, etc.). Ainsi, si je voulais développer pour Windows Phone et le Web, je devais installer 2 versions de Visual Studio Express. Pour pallier ce problème, Microsoft met à disposition, une nouvelle version de Visual Studio : **Visual Studio Community 2013**. Cette nouvelle édition, entièrement gratuite pour tous les projets qui ne sont pas liés à l'entreprise, est une version complète de Visual Studio (basée sur la version Pro de Visual Studio) qui permet aux développeurs de cibler n'importe quelle plateforme, de l'application native au Web, en passant par le Cloud et le cross-plateforme. De plus, cette version supporte le modèle d'extensibilité mis en place dans les éditions existantes de Visual Studio. Ainsi, l'ensemble des plugins et composants disponibles sur Internet ou dans la galerie de Visual Studio (accessible ici : https://visualstudiogallery.msdn.microsoft.com) peuvent être téléchargés et installés dans Visual Studio Community 2013. C'est le cas, par exemple, de Resharper (https://www.jetbrains.com/resharper/), de l'excellent plugin Visual Studio Tools for Unity (https://visualstudiogallery.msdn.microsoft.com/20b80b8c-659b-45ef-96c1-437828fe7cf2) ou du bien connu plugin Web Essentials for Visual Studio (http://vswebessentials.com) : Fig.1.

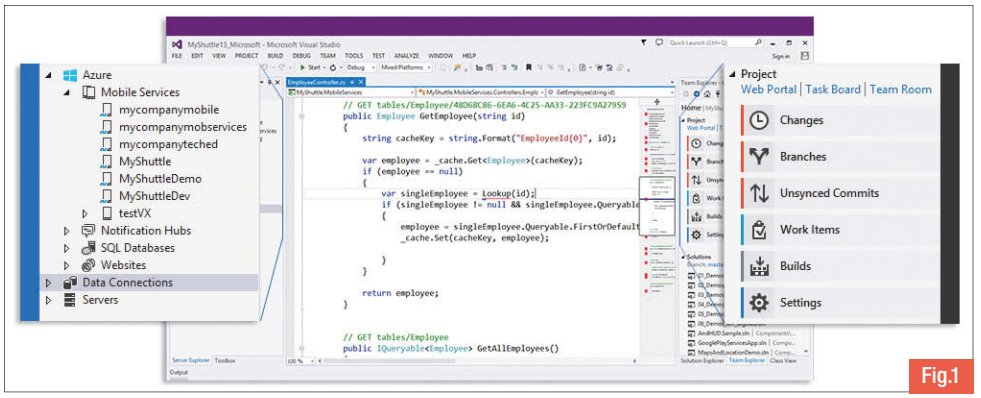


Fig.1

Visual Studio 2015 Preview et .NET 2015 Preview:

En marge de ces annonces (qui, à elles seules, ont provoqué une grande effervescence de la part de la communauté sur Twitter et autres réseaux sociaux), la conférence Connect() a également permis de faire le lancement de Visual Studio 2015 Preview et du Framework .NET 2015, en version préliminaire, là-encore. Ainsi, Visual Studio 2015 propose une intégration native de « Roslyn » et les nouvelles fonctionnalités du langage C#. Cette version est à présent testable dans sa version 6 et facilite grandement le travail du développeur en lui proposant tout un ensemble de techniques servant de « sucre syntaxique ». De plus, le nouveau système d'ampoule (que l'on retrouve sur Resharper) permet de proposer un refactoring proactif et des possibilités d'optimisations et d'améliorations de code à la volée, au fur et à mesure que vous écrivez votre code : Fig.2.

Afin d'améliorer la productivité des développeurs, des améliorations sur toute la partie « Debugging » ont également été apportées. Ainsi, la fonctionnalité PerfTips se révèle être très intéressante : il devient dorénavant possible de voir le temps qui a été passé sur l'exécution de chaque ligne de code (ou entre plusieurs points d'arrêt) : Fig.3.

Côté Framework, il y a eu là aussi des nouveautés. Ainsi, dans .NET 4.6 (la prochaîne version majeure du Framework .NET), le Framework WPF (Windows Presentation Foundation, qui permet de créer des applications Windows utilisant le XAML comme langage de description d'interfaces) a connu des évolutions. Lui, que beaucoup pensaient abandonné (à tort !), se voit enrichi, entre autres, du support pour les fenêtres transparentes. La Roadmap pour WPF a été annoncée par les équipes de développement à la suite de la conférence et, à la lecture de cette Roadmap (accessible ici : http://blogs.msdn.com/b/dotnet/archive/2014/11/12/the-roadmap-for-wpf.aspx), il est clairement possible d'affirmer que WPF est tout sauf mort, loin de là ! Et cela, c'est sans parler des outils, inclus dans Visual Studio 2015, dont le but est de faciliter le développement WPF : un outil de diagnostic visuel, un autre permettant d'analyser les performances et même un nouveau Blend, intégrant lui aussi tout un ensemble de nouvelles fonctionnalités (telle que la modification de code XAML durant le débogage, etc.) !

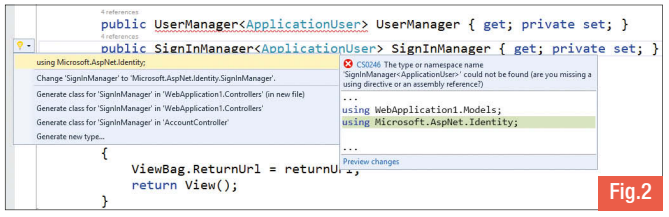


Fig.2

Fig.3

```
60          m_images = new ConcurrentBag<ImageItem>();
61
62          await GetImagesFromCloud();
63          await LoadImagesFromDisk(KnownFolders.PicturesLibrary);  ≤905ms elapsed
64
65          ImageListSource.Source = from image in m_images
```

La partie Cross-Plateforme n'est pas en reste dans cette nouvelle mouture. Ainsi, une collaboration de plus en plus proche entre Microsoft et Xamarin a été dévoilée aujourd'hui. En plus de l'intervention de Miguel De Icaza, lors de la Keynote de la conférence Connect(), en binôme avec Scott Hanselman, cette collaboration passe par 3 grandes étapes :

- Une réduction plus importante pour Xamarin (Business ou Entreprise), pour les abonnés MSDN (https://xamarin.com/msdn)
- Une intégration plus poussée entre Xamarin et Visual Studio, grâce à la possibilité d'installer Xamarin depuis Visual Studio, ainsi que la mise à disposition de modèles de projets/d'éléments pour Xamarin dans Visual Studio.
- Le support de Visual Studio dans l'offre « Starter » de Xamarin (jusqu'à maintenant, il fallait disposer de la version Business pour pouvoir développer des applications Xamarin dans Visual Studio) Fig.4.

Pour compléter cet aspect cross-plateforme, Microsoft a annoncé la mise à disposition, dans Visual Studio 2015, d'un émulateur pour Android ! Basé sur une architecture x86, celui-ci se veut disposer de performances optimales, tout en proposant des fonctionnalités telles que l'accéléromètre, la position de l'utilisateur, etc. Techniquement, il s'agit d'une image Hyper-V, et, si on lance l'émulateur, on constate d'ailleurs que cet émulateur ressemble énormément à celui de Windows Phone (d'un point de vue visuel) : Fig.5.

Pour le moment, seules 2 images Hyper-V sont disponibles (une pour cibler un périphérique de type Tablette et l'autre permettant d'identifier un périphérique de type Téléphone).

Cet émulateur sera utilisable avec les projets Xamarin et les applications Apache Cordova qui, dans Visual Studio 2015, se voit doté d'un support intégré permettant de développer, debugger, analyser, packager et déployer des applications iOS, Android et Windows. A noter que cette nouvelle version des Visual Studio Tools for Cordova inclut le support pour Windows 8.1 et les applications Universelles, ainsi que le débogage iOS directement dans Visual Studio : Fig.6.

Enfin, avec Visual Studio 2015, Microsoft met à disposition des développeurs une suite complète permettant de développer des applications mobiles cross-plateformes en... C++ ! Cela inclut le support pour le compilateur Clang et LLVM, ciblant Android pour le moment (mais le support d'iOS a déjà été annoncé par Microsoft). Techniquement, cela vous permet d'éditer et de déboguer un seul et même code C++, afin de faire en sorte que celui-ci fonctionne pour iOS, Android et Windows.

Coté Web, il y a également eu des nouveautés qui ont été annoncées. Ainsi, .NET 2015 intègrera ASP.NET 5, permettant de développer des applications pour le Web et pour le Cloud. Si ASP.NET vNext a déjà été explicité un grand nombre de fois (comme, par exemple, par Scott Hansleman : http://www.hanselman.com/blog/IntroducingASPNETVNext.aspx), la découverte ici a été d'apprendre qu'ASP.NET 5 sera cross-plateforme, open-source et, surtout, que les applications ASP.NET pourront s'exécuter en « side by side » avec des versions différentes du Framework. De plus, les applications ASP.NET 5 deviennent plus autonomes : il n'est plus nécessaire de disposer d'un serveur Web pour faire tourner les applications !

L'évènement Connect() a permis à Microsoft de montrer aux développeurs un aperçu du travail sur lequel les équipes de développement ont travaillé depuis quelques temps déjà. Certes, il s'agit là de versions préliminaires et donc devant être testées afin de détecter et remonter des bugs. Cependant, ces premières versions sont plus qu'intéressantes afin d'avoir un aperçu de ce que nous réserve Microsoft pour l'année 2015 !

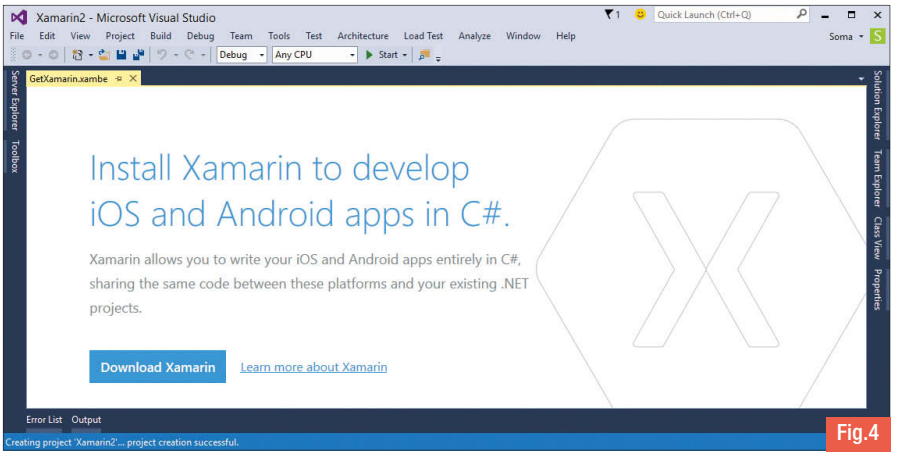


Fig.4

Fig.5

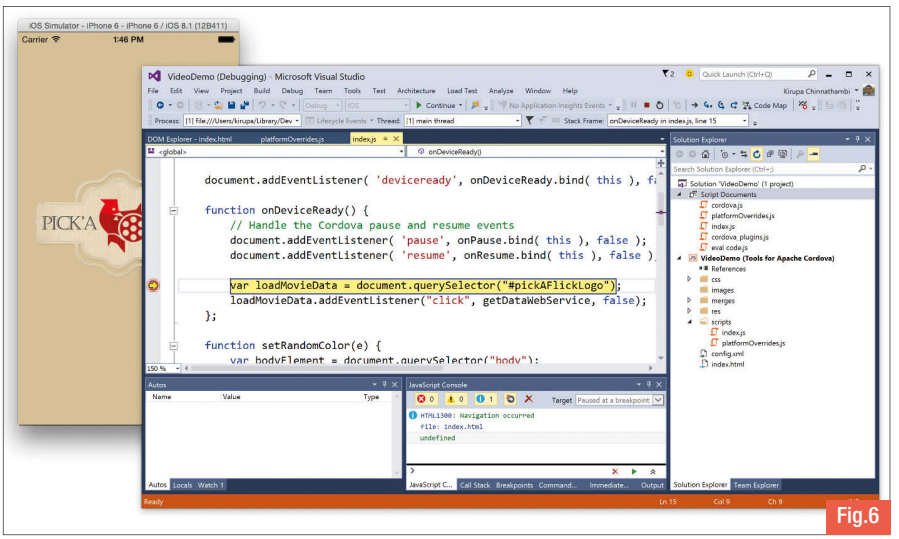


Fig.6

Développez avec Wordpress 2e partie

Maintenant que vous maîtrisez parfaitement l'outil « administration » de Wordpress (Programmez ! 179) nous allons nous occuper de la plomberie de ce fabuleux CMS. Wordpress est entièrement customisable et propose un large panel d'outils qui vous permettront de donner une personnalité (avec les thèmes) et des fonctionnalités (les plugins) uniques à votre site.

Freddy Hebrard
Responsable SI Centre de Ressources Régional Lorraine
MCTS/MCPD Microsoft
freddyhebrard@hotmail.com
@fredhebrard

Je vous conseille de jeter un petit coup d'œil à la bible de WordPress : le codex, que vous trouverez à cette adresse : http://codex.wordpress.org.
Ce fameux codex est le référentiel de l'ensemble des fonctionnalités de Wordpress ainsi que de l'aide au développement. Une fonction vous échappe, un petit tour coté codex ;)

WORDPRESS LA STRUCTURE

Déjà 11 ans

Comme vous avez pu le lire dans notre introduction du mois dernier, la première version de Wordpress est sortie en 2003,. Cela à une grande importance dans notre code, le code d'origine WORDPRESS est complètement procédural, sans aucune trace de POO. Avec les évolutions de PHP et de WORDPRESS, les nouvelles versions et nouveaux outils de la plateforme se sont développés de plus en plus en POO.
Dans la version actuelle de notre CMS, afin de garder une rétrocompatibilité importante, dans les entrailles de la bête, nous avons du code procédural et du code objet; ceci a son importance quand vous développerez, car il y aura à jongler avec les deux dans certains cas.

Les Bases de données.

Souvenez-vous sur l'article du mois dernier, lors de l'installation, Wordpress nous proposait de choisir le préfixe des tables, par défaut c'est « wp», nous partons du postulat que vous n'avez rien changé.
Il existe nativement avec Wordpress 4.0, lors de l'installation, 11 bases de données. L'ensemble des commentaires du site, rattachés aux articles par exemple sont contenus sur ces 2 tables :

- *wp_commentmeta*
- *wp_comments*

L'ensemble des paramètres du site, titre, slogan et autres joyeusetés, sont stockés ici :

- *wp_options*

Pour la rétrocompatibilité avec les anciennes versions de WP, une table notée « depracated » dans le « codex » permettait de stocker les informations relatives aux liens entrés dans la fonction « Link » de Wordpress.

- *wp_link*

Les contenus des publications du site se trouvent dans ces 2 tables : articles, pages, menus... Dans WordPress chaque article et page sont soumis au « versionning », il existe donc plusieurs versions dans ces tables.

- *wp_postmeta*
- *wp_post*

Les informations liées aux tags, liens avec les différents articles du site, catégories sont stockées directement dans ces tables.

- *wp_term_relationships*
- *wp_term_taxonomy*
- *wp_terms*

Toutes les données liées aux utilisateurs sont stockées dans ces 2 tables.

- *wp_usermeta*
- *wp_users*

Comprendre l'arborescence du dossier Wordpress

Quand vous naviguez sur votre système de fichier, dans le répertoire d'installation de Wordpress, vous pouvez voir un certain nombre de choses qu'il est utile de comprendre.
Les fichiers à la racine de ce dossier sont pour la majeure partie des fichiers de configuration (comme wp-config.php) ou des fichiers directement appelés à l'utilisation du site (par exemple le fichier index.php qui est notre fichier de base.)
Regardons de plus près, les 3 dossiers importants à la racine. Le dossier d'installation des thèmes et des plug-ins, à l'intérieur il y a deux autres dossiers respectivement appelés « thème » et « plugins » :

- **Wp-content**

La base, le moteur, les fonctionnalités de Wordpress sont stockées dans ce dossier :

- **Wp-includes**

L'interface de l'administration et les différentes fonctionnalités liées :

- **Wp-admin**

Ce que nous comprenons en regardant cette structure, c'est que notre développement de thème et de fonctionnalité sera dans un endroit précis et complètement détachable du reste de Wordpress.
Il est donc inutile quand vous développerez vos plugins et vos thèmes de toucher à autres choses à l'extérieur des dossiers thèmes et plugins contenus dans le dossier « wp-content », cela est même vivement déconseillé.

LES BASES DU DÉVELOPPEMENT WORDPRESS AVEC LA CRÉATION D'UN THÈME

Vous avez vu dans la première partie de ce dossier comme installer un thème, nous ne reviendrons pas dessus, nous allons pouvoir entrer tout de suite dans le vif du sujet.

Création et structure d'un thème

Avant tout, il faut créer un dossier au nom du thème, dans le dossier « thème », dans le dossier « **wp-content** ».
Mais que doit contenir ce dossier pour rendre le thème viable ? Simplement 2 fichiers :
1 fichier de style et un fichier modèle:

- « style.css »
- « index.php »

Le fichier style.css :

Ce fichier doit avoir une entête bien particulière pour fournir des informations particulières aux gestionnaires des thèmes, ces informations sont visibles sur la page de gestion des thèmes dans l'administration de Wordpress :

Exemple des éléments de base :

```
/*
Theme Name: Programmez !
Theme URI: http://wordpress.org/
Description: Premier theme by PROGRAMMEZ
Version: 1
Author: Programmez
Author URI: http://www.programmez.com
Tags: theme, wordpress, développement.
*/
```

Le fichier index.php :

Si le fichier modèle est non existant WORDPRESS vous le signalera en vous indiquant que le thème est « endommagé », dans l'administration de WORDPRESS (sélection du thème) Fig.1.

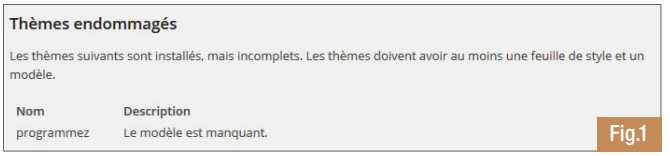


Fig.1

Structure d'un Template

Dans un Template vous trouverez généralement 3 éléments essentiels :

- Les feuilles de style
- Les images
- Les template

Un « Template » est un fichier qui permet de structurer les données récupérées en utilisant un modèle défini par le développeur et réutilisable.
Il est important de porter un focus tout particulier sur le système de « Hiérarchie de Template » de Wordpress.

Fallback et Hiérarchie Template

Wordpress utilise un mécanisme appelé le « Fallback », ce qui veut dire que quand il ne trouve pas un fichier particulier modèle (« Template »), il utilise un fichier modèle de niveau inférieur, plus générique, il « retombe » sur un fichier par défaut. Le fichier par défaut le plus bas est « index.php », ce qui explique sa présence obligatoire dans le dossier du thème. Ce système est utile et permet d'être rapidement réactif sur le développement, en effet vous ne développez que les Templates qui vous sont nécessaires; s'il n'y pas besoin de Template précis dans une situation donnée, alors on laisse le traitement par le Template plus bas, ce qui permet la mutualisation du code pour des affichages similaires. Par définition, le Template le plus bas doit être structuré avec le code le plus générique possible pour toute situation.

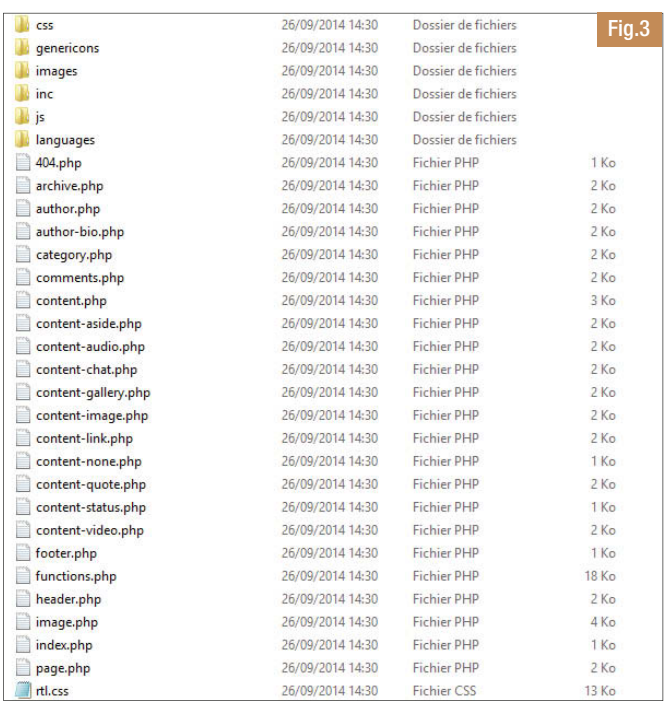


Fig.3

css	26/09/2014 14:30	Dossier de fichiers	
genericons	26/09/2014 14:30	Dossier de fichiers	
images	26/09/2014 14:30	Dossier de fichiers	
inc	26/09/2014 14:30	Dossier de fichiers	
js	26/09/2014 14:30	Dossier de fichiers	
languages	26/09/2014 14:30	Dossier de fichiers	
404.php	26/09/2014 14:30	Fichier PHP	1 Ko
archive.php	26/09/2014 14:30	Fichier PHP	2 Ko
author.php	26/09/2014 14:30	Fichier PHP	2 Ko
author-bio.php	26/09/2014 14:30	Fichier PHP	2 Ko
category.php	26/09/2014 14:30	Fichier PHP	2 Ko
comments.php	26/09/2014 14:30	Fichier PHP	2 Ko
content.php	26/09/2014 14:30	Fichier PHP	3 Ko
content-aside.php	26/09/2014 14:30	Fichier PHP	2 Ko
content-audio.php	26/09/2014 14:30	Fichier PHP	2 Ko
content-chat.php	26/09/2014 14:30	Fichier PHP	2 Ko
content-gallery.php	26/09/2014 14:30	Fichier PHP	2 Ko
content-image.php	26/09/2014 14:30	Fichier PHP	2 Ko
content-link.php	26/09/2014 14:30	Fichier PHP	2 Ko
content-none.php	26/09/2014 14:30	Fichier PHP	1 Ko
content-quote.php	26/09/2014 14:30	Fichier PHP	2 Ko
content-status.php	26/09/2014 14:30	Fichier PHP	1 Ko
content-video.php	26/09/2014 14:30	Fichier PHP	2 Ko
footer.php	26/09/2014 14:30	Fichier PHP	1 Ko
functions.php	26/09/2014 14:30	Fichier PHP	18 Ko
header.php	26/09/2014 14:30	Fichier PHP	2 Ko
image.php	26/09/2014 14:30	Fichier PHP	4 Ko
index.php	26/09/2014 14:30	Fichier PHP	1 Ko
page.php	26/09/2014 14:30	Fichier PHP	2 Ko
rtl.css	26/09/2014 14:30	Fichier CSS	13 Ko

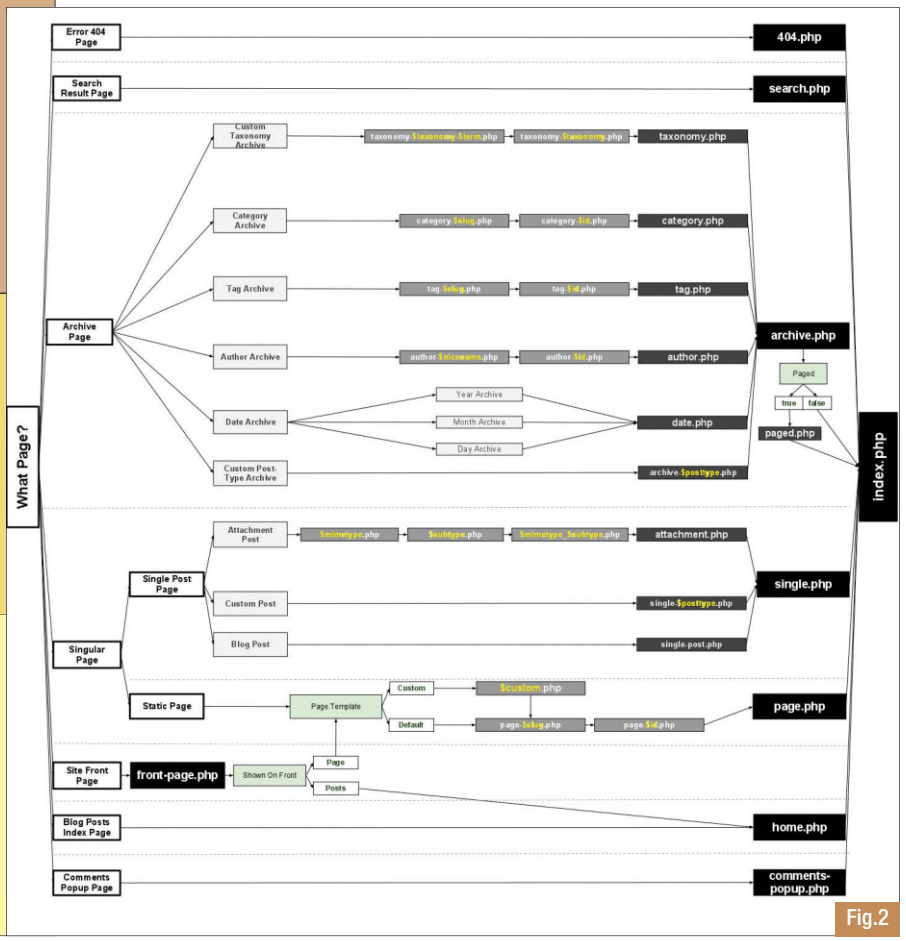


Fig.2

Schéma de hiérarchie de Template de Wordpress : Fig.2

Exemple de Template :

Le fichier modèle « category.php » permet de définir un affichage générique pour l'ensemble des « catégories » d'articles. Concrètement, si vous avez utilisé un lien vers une catégorie dans votre menu principal, par exemple une catégorie « programmez », le lien cherchera à afficher le fichier modèle (« Template ») « category-programmez.php », qui détermine la structure d'affichage pour cette catégorie précise; s'il ne trouve pas alors il retombera vers le fichier générique « category.php » qui permettra de lister les articles en utilisant la structure HTML, avec vos stylesCSS et vos effets JavaScript que vous avez décidés en amont.
Si il ne trouve pas le fichier category.php, grâce au mécanisme de fallback, il appellera un fichier modèle de niveau inférieur, et en dernier recours « index.php ».

Concrètement :

- L'utilisateur clique sur la catégorie « programmez » dans son menu

Bienvenue dans l'univers du code avec

PROGRAMMEZ!
le magazine du développeur

08-22-13 © esenkartal

Spécial étudiant 39€(*)
1 an 11 numéros

1 an 11 numéros 49€ seulement (*)

2 ans 22 numéros 79€ seulement (*)

ABONNEZ-VOUS !

(*) Tarifs France métropolitaine

Toutes nos offres sur www.programmez.com

Oui, je m'abonne

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

☐ ***Abonnement 1 an au magazine :*** 49 € (au lieu de 65,45 €, prix au numéro)
☐ ***Abonnement 2 ans au magazine :*** 79 € (au lieu de 130,9 €, prix au numéro)
☐ ***Abonnement spécial étudiant 1 ans au magazine :*** 39 €
Photocopie de la carte d'étudiant à joindre
Tarifs France métropolitaine

Offre spéciale : abonnement + clé USB Programmez!
☐ 1 an (11 numéros) + clé USB : 60 €
☐ 2 ans (22 numéros) + clé USB : 90 €
Clé USB contenant tous les numéros de Programmez! depuis le n°100, valeur : 29,90 €
Tarifs France métropolitaine

☐ M. ☐ Mme ☐ Mlle Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

Tél : ***(Attention, e-mail indispensable pour les archives sur internet)***

E-mail : @

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

PROG 180
Offre limitée, valable jusqu'au 31 décembre 2014

- Wordpress cherche category-programmez.php et le renvoie s'il le trouve.
- Autrement, Il renvoie « category.php »
- S'il ne trouve pas il renvoie « index.php »

Exemple de structure système de fichier pour un thème existant dans Wordpress : Fig.3.

Nous pouvons voir un ensemble de fichiers (dont « index.php »); il existe plusieurs types de modèles que vous pouvez créer, on a vu « category.php », mais aussi « content.php » qui détermine l'affichage générique d'un contenu (article/page), « footer.php » qui représente la structure d'un pied de page, « header.php » la structure de la partie haute de votre page, « 404.php » inutile de la présenter, etc.

Le fichier functions.php

Il existe un fichier qui attire notre attention dans le répertoire de notre thème : « functions.php ». Il permet de définir un ensemble de fonctions, d'actions, de paramètres propres au thème utilisé. Il est chargé automatiquement par le moteur de Wordpress lorsqu'il se trouve à la racine du dossier de thème. C'est pour cela que beaucoup d'éléments sont initialisés dans ce fichier.

Bonne pratique : Ce fichier peut faire appel à d'autres fichiers de fonction, afin de rendre le code plus lisible et plus clair (bonne pratique) afin d'éviter de se retrouver avec une page functions.php regroupant toutes les fonctions de votre thème les unes avec les autres. Ça devient rapidement illisible.

Exemple de code dans functions.php (thème par défaut twentythirteen) Fig.4.

Les hooks : les actions et les filtres

Les hooks sont des « outils » de WordPress qui permettent d'intercepter des évènements de WordPress (ou d'appeler à des moments précis une/des fonctions) afin de modifier un comportement ou des données.

Il existe deux types de « Hooks » : les actions et les filtres.

Les actions :

Les actions sont des fonctions particulières qui sont appelées lors d'un déclenchement d'évènement dans WordPress.

Un évènement peut être observé par plusieurs fonctions, elles sont rattachées à cet évènement, il peut être observé par un nombre indéfini de fonctions.

Quand un évènement est déclenché, l'ensemble des fonctions rattachées sont exécutées dans un ordre définissable par le développeur.

L'action s'ajoute par l'utilisation de la fonction add_action() :

```
add_action($event_wordpress,$nom_de_ma_fonction_a_appeler,
$priorite) ;
```

`$event_wordpress` : nom de l'évènement à observer

`$nom_de_ma_fonction_a_appeler` : nom de la fonction à rattacher, sans parenthèse.

`$priorité` : priorité d'exécution de la fonction par rapport aux autres observants, la valeur par défaut étant 10. (plus le niveau est faible plus elle est prioritaire).

Déclencher un évènement avec la fonction do_action() :

```
do_action($event_wordpress,$arga,$argb[…] ) ;
```

```
// This theme uses wp_nav_menu() in one location.
register_nav_menu( 'primary', __( 'Navigation Menu', 'twentythirteen' ) );

/*
 * This theme uses a custom image size for featured images, displayed on
 * "standard" posts and pages.
 */
add_theme_support( 'post-thumbnails' );
set_post_thumbnail_size( 604, 270, true );
```

Fig.4

`$event_wordpress` : nom de l'évènement à lancer.

`,$arga,$argb[…]` : série d'arguments (s'il y a) à passer en paramètres des fonctions accrochées et séparées par des virgules.

Un exemple bien précis de l'utilisation des actions dans Wordpress :

Si vous avez lu le premier article sur l'administration et l'installation de Wordpress, on a parlé rapidement des widgets que l'on peut rajouter dans certaines « zones » de notre page (déterminées par le développeur)

L'initialisation de ces widgets, au chargement de la page, se passe grâce au déclenchement de l'évènement « widget_init ». Concrètement, un ajout d'une zone de widget se passe grâce aux actions.

Voici un exemple que vous trouverez directement dans le fichier functions.php du thème par défaut twentythirteen de Wordpress :

```
function twentythirteen_widgets_init() {
  register_sidebar( array(
    'name'          => __( 'Main Widget Area', 'twentythirteen' ),
    'id'            => 'sidebar-1',
    'description'   => __( 'Appears in the footer section of
the site.', 'twentythirteen' ),
    'before_widget' => '<aside id="%1$s" class="widget %2$s">',
    'after_widget'  => '</aside>',
    'before_title'  => '<h3 class="widget-title">',
    'after_title'   => '</h3>',
  ) );

  register_sidebar( array(
    'name'          => __( 'Secondary Widget Area', 'twentythirteen' ),
    'id'            => 'sidebar-2',
    'description'   => __( 'Appears on posts and pages in
the sidebar.', 'twentythirteen' ),
    'before_widget' => '<aside id="%1$s" class="widget %2$s">',
    'after_widget'  => '</aside>',
    'before_title'  => '<h3 class="widget-title">',
    'after_title'   => '</h3>',
  ) );
}
add_action( 'widgets_init', 'twentythirteen_widgets_init' );
```

Dans ce code nous avons :

- Une fonction (twentythirteen_widgets_init()) qui nous permet d'initialiser 2 zones de widgets, grâce à la méthode d'enregistrement « register_sidebar() ; » qui prend en paramètre un tableau de paramètre de notre zone.
- L'enregistrement à l'observation de l'évènement 'widgets_init' de notre fonction grâce à l'action add_action().

A l'appel de la fonction **do_action('widgets_init',...)** ; la fonction twentythirteen_widgets_init()) sera exécutée.

Les filtres :

Ils servent à modifier du texte ou des types à l'affichage navigateur ou sur votre BDD; nous aborderons ce sujet plus en détails avec la création des plugins/widget.

Template tags.

Comprendre le développement avec Wordpress passe par cette étape cruciale de l'utilisation des Template Tags.

Les Template Tags sont des fonctions de bases fournies par Wordpress que nous pouvons appeler dans nos développements pour restituer du contenu comme :

- Un article : son titre, son contenu
- Les informations du site : slogan, titre, adresse.
- Des outils pratiques.

Exemple de Template tags :

- the_title() : qui nous donne le titre d'un article ou d'une page.
- the_content() : qui nous donne le contenu d'un article ou d'une page.
- the_permalink() : le permalien de l'article ou de la page.
- Etc...

Mais aussi des outils :

- get_calendar() : récupere le calendrier.
- wp_loginout() : permet d'afficher le login / logout.
- Etc..

Ou encore les infos sur le site :

- bloginfo('name') : titre du blog
- bloginfo('charset') : le charset utilisé
- etc...

Ou des portions complètes de code :

- get_header() : permet de récuperer l'entête contenue dans le fichier header.php
- get_footer() : permet de récupérer le pied de page contenu dans le fichier footer.php
- comments_template() : inclue le fichier comments.php qui est le Template des commentaires d'une publication. Ce Template tag est à fournir dans « la boucle », que nous voyons tout de suite après.

L'utilisation de ces deux Template tags nécessite de bien concevoir comment on va construire ses pages :

Exemple dans le fichier index.php d'un de nos thèmes par défaut dans Wordpress (code volontairement tronqué) : Fig.5.

Toutes les balises ouvrantes <html> <body> etc... devront se trouver dans le fichier « header.php » inséré par l'appel du template tag « get_header() » et des balises fermantes de ces même éléments dans le fichier « footer.php ». Dans le fichier « index.php », on aura inséré nos deux « template tags » et entre les deux notre contenu. Ça marche évidement avec n'importe quel Template autre que « index.php »

```php
<?php
/**
 * The main template file
 *
 * This is the most generic template file in a WordPress theme and one
 * of the two required files for a theme (the other being style.css).
 * It is used to display a page when nothing more specific matches a query,
 * e.g., it puts together the home page when no home.php file exists.
 *
 * @link http://codex.wordpress.org/Template_Hierarchy
 *
 * @package WordPress
 * @subpackage Twenty_Fourteen
 * @since Twenty Fourteen 1.0
 */

get_header(); ?>

<div id="main-content" class="main-content">

<?php
    if ( is_front_page() && twentyfourteen_has_featured_posts() ) {
        // Include the featured content template.
        get_template_part( 'featured-content' );
    }
?>
[.........]

<?php
get_footer();
```

Fig.5

```php
    // Start the Loop.
    while ( have_posts() ) :
        the_post();
        the_title();
        the_content();

    endwhile;
```

Fig.6

Le contenu : comprendre la boucle !

Nous avons maintenant le minimum pour créer un thème dans Wordpress : la structure d'un thème dans le système de fichiers, les hooks, la compréhension synthétique de la hiérarchie de Template, les Template tags.

Cependant quand vous lancez votre page, rien ne se passe, c'est le vide complet ! En effet il n'y a aucun code dans votre index.php, je vous propose ici de comprendre l'élément fondamental dans la restitution de données : la boucle.

La boucle est l'outil, dans Wordpress, qui nous permet de parcourir l'ensemble d'un contenu. Un contenu peut être basé sur les données d'une page, d'un article, d'une catégorie d'articles, d'un tag d'articles etc. L'utilisation de la boucle passe par... l'utilisation d'un Template tag : have_posts() dans une boucle.

Exemple :

```
while (have_posts()) :
    [****** mon code ******]
endwhile;
```

Simple non ? Le Template tag « have_posts() » renvoie un booléen (true/false) afin de déterminer s'il reste des objets à parcourir.

Je vous donne un exemple simple; j'ai créé un fichier Template « category.php » de base générique qui permet d'afficher l'ensemble des articles pour une catégorie donnée, have_posts() me renverra « true » tant que la liste d'articles contenus dans la catégorie n'a pas été complètement parcourue Fig.6.

Le Template tag « the_post() » permet de récupérer l'article pointé, the_title(), le titre de l'article, et the_content(), son contenu.

La boucle à toutes les sauces, même pour une page :

La boucle est utilisée pour lister un ensemble d'articles comme ici, mais aussi pour récupérer le contenu d'une seule page ou d'un seul article !

De part sa hierarchie de template, l'écriture du code devra être la plus générique possible au fur et à mesure que l'on se rapproche de l'index.php..

Le cas particulier, c'est un blog, comment afficher les commentaires ?

Dans les exemples de template tag, je vous ai parlé rapidement de « comments_template() » qui nous permet l'intégration du template « comment.php ».

Dans ce fichier il suffit d'appeler un autre template tag qui permet le listing des commentaires : wp_list_comments(), sans paramètre de base entre deux balises <ol>, une ouverte et une autre fermante.

Il prend aussi un tableau de paramètres, pour définir son affichage :

```
wp_list_comments( array(  'style'      => 'ol',
       'short_ping' => true,
       'avatar_size'=> 34,
) );
```

Ajouter des zones de fonctionnalités à nos templates.

Nous savons ajouter le contenu dans nos templates en utilisant la boucle, mais nous avons besoin d'une navigation pour notre site et nous souhaitons aussi pouvoir intégrer des Widgets facilement sur notre page sans passer à chaque fois par le code, c'est ce que nous allons découvrir dans cette section.

Zone de navigation / menu :

L'ajout d'une zone de menu par deux étapes importantes :

- Observation de l'évènement «init » et enregistrement par « register_nav_menu() »
- Intégration HTML dans le Template.

Il existe une action pour accrocher une fonction à un évènement de manière à pouvoir l'observer.

L'évènement observable est « init », la fonction rattachée doit impérativement appeler la fonction d'enregistrement de notre menu register_nav_menu().

Voici en détail les paramètres de cette fonction :

```
register_nav_menu($identifiant, $nom_afficher);
```

`$identifiant` : identifiant unique de notre zone de menu.
`$nom_afficher` : le nom qu'on affiche dans la personnalisation de notre site, dans l'administration.

Bonne pratique : ce code doit se trouver dans le fichier functions.php ou un fichier inclus dedans.

Et l'intégration HTML ?

Rien de plus simple elle passe par le template tag « wp_nav_menu() » à intégrer dans le/les templates où vous désirez voir apparaître cette zone de navigation.

Cette fonction prend plusieurs paramètres sous forme d'un array, dont 1 obligatoire : l'identifiant.

```
<?php
wp_nav_menu(array(
  'theme_location' => 'main_menu',
  […]
  ));
?>
```

Liste des parametres :

- theme_location : identifiant de la zone de menu à insérer.
- menu_class : class CSS pour cette zone de menu à appliquer.
- before : contenu HTML à intégrer avant la zone.
- after : contenu HTML à intégrer après la zone.
- container : permet de définir la balise contenant la zone de navigation, par défaut c'est une .

Zone de widget :

L'ajout d'une zone de widgets est simple et passe par deux étapes importantes :

- Observation de l'évenement « widgets_init » et enregistrement par « register_sidebar() ».
- Integration HTML dans le template.

Nous l'avons entrevu un peu plus haut dans notre article, il existe une action pour ça: observer l'évènement « widgets_init »;. Dans notre fonction rattachée, nous devons appeler une autre fonction : « register_sidebar() » qui prend en paramètre un array() de paramètres pour l'enregistrement de notre zone de widgets.

Regardons en détail le contenu des paramètres que nous avons dans le fichier functions.php de nos templates par défaut de Wordpress :

```
register_sidebar( array(
    'name'          => __( 'Main Widget Area', 'twentythirteen' ),
    'id'            => 'sidebar-1',
    'description'   => __( 'Appears in the footer section of the site.', 'twentythirteen' ),
    'before_widget' => '<aside id="%1$s" class="widget %2$s">',
    'after_widget'  => '</aside>',
    'before_title'  => '<h3 class="widget-title">',
    'after_title'   => '</h3>',
) );
```

Liste des paramètres :

- name : correspond au nom qui sera affiché dans la zone coté administration (ex : barre latérale principale).
- description : texte qui s'affiche dans l'administration des widgets pour décrire le contenu de cette zone ou autre (ex : Barre Latérale principale qui apparait à gauche).
- id : id unique de notre zone.
- before_widget : balise et autres code HTML qu'on affiche avant CHAQUE widget.
- after_widget : la même, mais après CHAQUE widget.
- before_title : balises et autres code HTML qu'on doit afficher avant chaque titre de widget.
- after_title : idem que le before mais après les titres.

Bonne pratique : ce code est à placer de préférence dans functions.php ou dans un fichier séparé appelé dans functions.php Fig.7.

Comment afficher notre zone ?

Simplement en intégrant dans le Template qui intègre la zone :

```
<div><?php dynamic_sidebar('id_de_la_zone');?></div>
```

Conclusion

Maintenant, vous avez les clefs pour bâtir déjà pas mal de choses avec Wordpress 4.0 et les thèmes. Vous maîtrisez les concepts de boucle, Template tags, actions, filtres, hooks, hiérarchie Templates, Templates, et vous savez mettre en œuvre une structure de Template réutilisable et exportable dans le CMS. Comment aller plus loin ? En lisant l'article du mois prochain où nous parcourrons ensemble la création d'un plugin et widget, l'internationalisation, une revue sur les filtres. Entretemps n'hésitez pas un faire un tour du côté du codex.

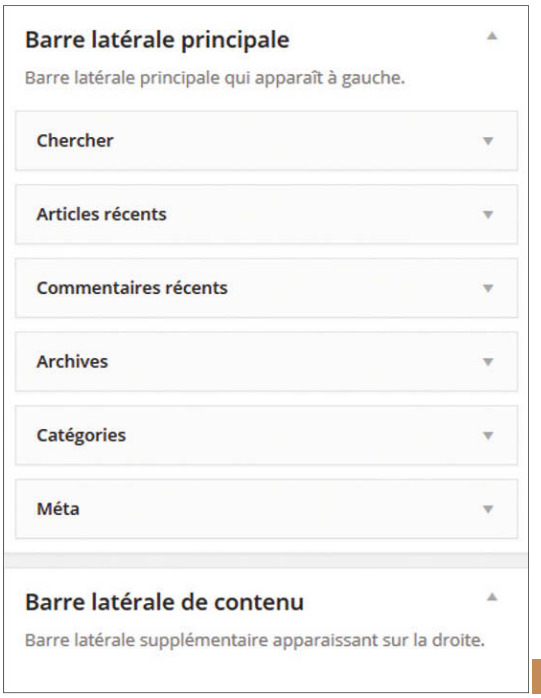


Fig.7

SWIFT : le nouveau langage pour OS X et iOS

Ce nouveau langage à destination des plateformes de la marque à la pomme (tout du moins pour l'instant), est présenté comme plus simple, plus moderne, plus flexible et plus drôle que ses ancêtres C et Objective-C.

Stéphane Cordonnier
Directeur Technique – Bliz (Spécialiste Microsoft .NET, SharePoint, Mobilité)
http://www.bliz.fr

Le but n'est pas de proposer un nouveau langage de développement juste pour le plaisir. Au lieu de cela, prenez ce qui se fait de mieux sur toutes les plateformes existantes à ce jour (C, Objective-C, C#, Java, Ruby, Python, JavaScript, etc...) et regroupez le tout afin de permettre aux développeurs de créer des applications plus stables et plus puissantes de manière plus rapide et plus efficace que ce qu'ils connaissent jusqu'ici.

TOUR D'HORIZON DU LANGAGE SWIFT

Le premier programme écrit par les développeurs consiste à afficher le texte « Hello world » au sein d'une application. Pour cela rien de plus simple, et une seule ligne de code suffit :

```
println ("Hello world")
```

Nul besoin de définir un point d'entrée à son application comme cela est le cas dans d'autres langages. Avec Swift, tout code écrit à un niveau global est utilisé comme point d'entrée de l'application.
On remarque également qu'il n'est pas nécessaire de mettre un point-virgule à la fin de la ligne afin d'indiquer que les instructions sont terminées. Chaque retour à la ligne est considéré comme la fin d'une instruction. Il est bien évidemment possible de contrôler plus finement le flux d'exécution des instructions comme nous le verrons ultérieurement.

Variables/constantes utilisant des types simples

La déclaration de variables et/ou de constantes est généralement le point suivant auquel tout développeur aura recours pour réaliser son application. Pour cela il suffit d'utiliser la syntaxe ci-dessous :

```
var anInteger = 42
anInteger = 50
let integerConstant = 42
```

Comme on peut le voir ici, Swift utilise deux mots clés pour différencier les variables (var) des constantes (let). Cette distinction syntaxique permet en un coup d'œil de comprendre la volonté du développeur de vouloir déclarer une variable pouvant être modifiée ultérieurement (la variable nommée anInteger ci-dessus). Cette distinction permet également au compilateur de pouvoir générer une erreur avant même l'exécution du programme, notamment si le développeur écrivait quelque chose comme :

```
integerConstant = 50
```

On constate également dans la déclaration des variables et constantes, que la notion de type de données n'apparaît pas. Les types de données sont déterminés automatiquement par le langage au moment de la compilation de l'application.
Pour cela le langage utilise la notion d'inférence de types. Afin de typer différemment les données lors de la déclaration, il suffit d'utiliser la syntaxe adéquate comme on peut le voir ci-dessous :

```
let anInteger = 70
let aDouble = 70.0
let aString = "Hello world"
```

Si le développeur souhaite malgré tout spécifier lui-même quel type de donnée il souhaite utiliser pour la déclaration d'une variable et/ou constante, il peut le faire de la manière suivante :

```
let anInteger: Int = 70
let aDouble: Double = 70
let aString: String = ''Hello world''
```

Une des particularités de Swift par rapport à Objective-C, concerne le fait que le langage est fortement typé. Cela signifie qu'aucune conversion de type ne sera effectuée par l'application sans que le développeur l'ait explicitement demandé. Si le développeur souhaite combiner des types différents, il est de sa responsabilité de convertir correctement les données :

```
let aString = "The integer value is "
let anInteger = 70
let integerString = aString + String(anInteger)
```

Lorsque l'on souhaite manipuler des chaînes de caractères comme dans l'exemple précédent, il existe toutefois une syntaxe simplifiée afin d'arriver au même résultat :

```
let anInteger = 70
let integerString = "La valeur est égale à \(anInteger)"
```

Parmi les types de données régulièrement utilisés en développement, on retrouve également les tableaux et dictionnaires de données. Leur déclaration s'effectue en respectant la syntaxe suivante :

```
var stringArray = ["Value1", "Value2", "Value3", "Value4"]
var stringDictionary = [
  "Key1": "Value1",
  "Key2": "Value2",
]
```

Si l'on souhaite créer des tableaux et/ou dictionnaires vides, il est nécessaire de préciser le type de données qui sera stocké à l'intérieur comme on peut le voir ci-dessous :

```
let stringArray = [String]()
let stringDictionary = [String: String]()
```

Contrôler le flux d'exécution

Dans toutes les applications, il est nécessaire de pouvoir contrôler le flux d'exécution en fonction des valeurs des variables/constantes. Pour cela, Swift utilise la syntaxe if-else que l'on retrouve dans de très nombreux langages.

```
let anInteger = 42
if anInteger > 50 {
   prinln("La valeur est supérieure à 50")
 } else {
```

```
    prinln("La valeur est inférieure ou égale à 50")
  }
```

Pour les cas contenant plus que deux possibilités comme ci-dessus, il est possible d'utiliser la syntaxe switch-case, qui permet de gérer des cas plus ou moins complexes :

```
let aColor = "Rouge Foncé"
switch  aColor  {
case "Noir":
 println("La couleur est noire")
case "Bleu", "Blanc":
 println("La couleur est bleu ou blanc")
case let x where x.hasSuffix("Foncé"):
 println("La couleur est foncée")
default:
 println("La couleur est indéterminée")
}
```

Lorsque l'on souhaite itérer plusieurs fois de suite (ex : parcourir les données d'un tableau), il est possible d'utiliser différents types de boucles tels que *for*, *for-in*, *while* et *do-while* :

```
var stringArray = ["Value1", "Value2", "Value3", "Value4"]
for aString in stringArray {
println(aString)
}
```

Fonctions, paramètres et valeur de retour

Afin de structurer de manière optimale le code d'une application, il est nécessaire d'organiser celui-ci en blocs répondant à des besoins métiers particuliers (ex : effectuer un calcul, afficher des données, etc...). Swift offre cette possibilité via la création de fonctions qui peuvent (ou pas) prendre des paramètres, et peuvent (ou pas) retourner des valeurs.
La déclaration et l'appel d'une fonction simple s'effectuent de la manière suivante :

```
func helloWorld() {
 println("Hello world")
}
helloWorld()
```

Si le comportement de la fonction nécessite l'utilisation de paramètres, ceux-ci peuvent être spécifiés entre les parenthèses comme ceci :

```
func helloWorld(name: String) {
 println("Hello \(name)")
}
helloWorld("Stéphane")
```

De la même manière, si la fonction est censée retourner une valeur à l'issue de son exécution, cela peut être fait comme indiqué ci-dessous :

```
func sumOfIntegers(numbers: Int...) -> Int {
 var sum = 0
 for number in numbers {
   sum += number
 }
 return sum
}
sumOfIntegers()
sumOfIntegers(10, 20, 30)
```

Vous aurez probablement noté dans l'exemple précédent, les trois petits points dans la déclaration des paramètres de la fonction. Cette syntaxe permet de définir que la fonction peut prendre un nombre indéterminé de valeurs lors de l'appel de celle-ci. C'est notamment ce que l'on voit avec les deux appels successifs où le premier ne fournit pas de valeurs à la fonction, tandis que le second fournit trois valeurs à cette même fonction.

Comment bien démarrer avec Swift ?

Deux ouvrages très complets ont été rendus disponibles depuis le premier jour sur iBooks Store, accessible depuis tous les Macs, iPhone et iPad. Ces ouvrages ont été mis à jour à plusieurs reprises depuis l'annonce de Swift en Juin dernier.
Le premier nommé tout simplement « Welcome to Swift » contient près de 450 pages permettant de découvrir le langage tout d'abord de manière rapide. Si vous souhaitez ensuite aller plus loin, il reprend en détail chacun des aspects du langage avec de nombreux exemples à l'appui pour mettre tout cela en application.
Le second ouvrage nommé « Using Swift with Cocoa and Objective-C » s'attarde plus précisément sur l'interopérabilité entre les langages et avec le framework d'Apple. Sont également abordés dans ce livre les aspects de migration de son code Objective-C vers Swift et les impacts à prendre en considération.
On retrouve au sein de Xcode, toute la documentation associée au langage ainsi que la déclaration de toutes les classes, propriétés, méthodes, énumérations, etc... disponibles grâce à la compatibilité assurée entre Foundation et Swift. Afin de vous faire gagner du temps dans votre apprentissage de Swift, voici quelques-uns des liens les plus utiles afin de découvrir le langage en allant du débutant voulant simplement découvrir de quoi il s'agit, à l'expert voulant pousser le langage dans ses derniers retranchement :

Swift Blog – https://developer.apple.com/swift/blog
Awesome Swift - https://swift.zeef.com/robin.eggenkamp
Ray Wenderlich Tutorials - http://www.raywenderlich.com/tutorials
NSHipster - http://nshipster.com/
180+ Swift Language Tutorials
https://www.youtube.com/playlist?list=PLxwBNxx9j4PUpjCEVwjqFvNecNvQ6Dj6G

Définition de classes et manipulation d'objets

La programmation objet est un standard du marché depuis très longtemps et Swift est bien évidemment en pleine ligne avec cette approche et permet donc de retrouver tout ce que l'on est en droit d'attendre quand on parle de POO (héritage, agrégation, surcharge, instanciation...).
Pour déclarer un objet, rien de plus simple que d'utiliser tous les éléments évoqués précédemment et de les englober dans une classe comme on le fait dans quasiment tous les autres langages orientés objets :

```
class Car {
 var numberOfDoors = 3
   func carDescription() -> String {
     return "La voiture possède \(numberOfDoors) portières."
 }
}
```

Pour manipuler l'objet défini, il est nécessaire de créer une instance de celui-ci afin de pouvoir manipuler ses propriétés et méthodes :

```
var aCar = Car()
car.numberOfDoors = 5
println(car.carDescription())
```

Si vous souhaitez maîtriser ce qui se passe durant la phase d'instanciation d'un objet par exemple pour effectuer des traitements complémentaires, il suffit de définir une fonction nommée init qui prendra (ou pas) les arguments nécessaires à l'instanciation de l'objet.

```
class Car {
 var numberOfDoors = 3
 var brandName: String
 init(brandName: String) {
   self.brandName = brandName
 }
 func carDescription() -> String {
   return "La voiture possède \(numberOfDoors) portières."
 }
}
```

L'utilisation du mot-clé *self* dans la fonction *init*, permet de lever l'ambiguïté qui peut exister entre le nom de propriété *brandName* de l'objet de la variable *brandName* passée en argument de la méthode *init*.
Lorsque l'on utilise les principes d'héritage en programmation objet, il est nécessaire de pouvoir définir de quelle classe parente hérite un objet, et de pouvoir surcharger les fonctions/méthodes définies dans la classe parente. Pour mettre en œuvre ces mécanismes, il suffit d'écrire ceci :

```
class Ferrari : Car {
 override func carDescription() -> String {
    return "Cette Ferrari possède \(numberOfDoors) portières
 mais surtout un très gros moteur."
 }
}
```

Enumérations

Afin de rendre le code plus lisible, les développeurs souhaitent généralement regrouper des valeurs associées au sein d'une syntaxe explicite, plutôt que de stocker des valeurs de type 1, 2, 3, 4, etc...
Pour cela, Swift offre la possibilité de créer des énumérations. La création d'une énumération se fait de manière très simple comme on peut le constater ici :

```
enum CompassPoint {
 case North
 case South
 case East
 case West
}
```

Une fois une énumération définie, il est possible de l'utiliser pour déclarer des variables du type en question, et d'affecter les valeurs aux variables de manière relativement simple.

```
var aPoint = CompassPoint.North
aPoint = .West
```

Comme on peut le voir, une fois le type de données de la variable déterminée par le compilateur (grâce à l'inférence des types), il n'est pas nécessaire de préciser le nom de l'énumération lors de l'affectation d'une valeur à la variable.
Ce principe fonctionne de la même manière si l'on souhaite tester les différentes valeurs d'une variable au sein d'un bloc switch-case :

```
switch aPoint {
case .North:
 println("Nord")
case .South:
 println("Sud")
case .East:
 println("Est")
case .West:
 println("Ouest")
}
```

Maintenant que nous avons abordé les grands principes du langage et de la syntaxe de celui-ci, voyons quelques-unes des nouveautés et particularités de celui-ci, notamment pour les développeurs Objective-C qui se demandent pourquoi ils devraient utiliser ce nouveau langage.

Les nouveautés/spécificités du langage pour les développeurs Objective-C

Tuples

Il arrive de temps à autre, notamment dans le cadre d'une fonction, que le développeur souhaite retourner plus d'une seule valeur. Ceci peut être réalisé en retournant un tableau/dictionnaire mais n'est pas très pratique et peut être source d'erreur. Swift offre aux développeurs la possibilité de manipuler des *Tuples* de valeurs, afin de permettre de gérer plusieurs valeurs comme un ensemble de données facilement manipulable.
La déclaration d'un *Tuple* se fait de la manière suivante :

```
let internalException = (1234567890, "Internal Exception")
```

Une fois un *tuple* défini, celui-ci peut être manipulé de manière simple comme on peut le voir ici :

```
let (errorCode, errorMessage) = internalException
println("Le code de l'erreur est \(errorCode)")
println("Le message de l'erreur est \(errorMessage)")
```

Optionals

Ce concept permet de définir qu'une variable peut avoir ou ne pas avoir de valeur. Cela n'existe pas en Objective-C même si un principe s'en approche fortement à savoir l'utilisation de la valeur *nil*, que l'on va d'ailleurs retrouver avec Swift pour gérer ce cas.
Lorsque l'on déclare une variable en Swift, celle-ci doit obligatoirement être initialisée sous peine d'avoir une erreur de compilation. Le fait de déclarer une variable comme étant optionnelle, permet de répondre au besoin que la variable puisse ne pas avoir de valeur lors de son initialisation ou même ultérieurement.

```
var aString : String  // Erreur de compilation
var anotherString : String ?// Déclaration d'une variable optionnelle
```

Pour manipuler une variable optionnelle par la suite, il est nécessaire de demander à Swift de forcer la lecture de la valeur, en ajoutant un point d'exclamation après la variable, si l'on est sûr que celle-ci contient une valeur. Si la variable ne contient pas de valeur, une erreur d'exécution se produit et l'application se terminera instantanément.

```
var aString : String ?
if aString != nil {
 println("La valeur de la variable est \(aString!)")
}
println("La valeur de la variable est \(aString!)")
 // Erreur d'exécution si la variable est nil
```

Generics

Il arrive souvent lorsque l'on écrit une application, de se rendre compte que des portions de code effectuent les mêmes traitements mais avec des types de données différents (ex : faire une somme de nombres entiers et faire une somme de nombres flottants). Afin de réduire la quantité de code écrite lorsque les traitements sont identiques, Swift offre la possibilité de définir des éléments comme génériques, c'est-à-dire dont les types ne sont pas figés à l'avance comme le nécessitent généralement les langages fortement typés. Pour reprendre l'exemple précédent d'ajout de nombres entiers et de nombres flottants, cela se traduirait par

l'écriture de fonctions distinctes bien que le traitement soit le même.

```
func addInteger(a: Int, b: Int) -> Int {
  return a + b
}
func addFloat(a: Float, b: Float) -> Float {
  return a + b
}
```

L'utilisation des génériques, permet de simplifier le code et de n'avoir qu'une seule et même méthode pour répondre à ces deux cas :

```
func addNumber<T>(a: T, b: T) -> T {
  return a + b
}
```

Ces principes de généricité de code s'appliquent également aux classes afin de rendre celles-ci moins dépendantes des types de données qu'elles manipulent.

Closures

Les *Closures* sont des blocs de code, permettant d'implémenter une fonctionnalité métier, et qui peuvent être appelés ou passés en tant que paramètres à des fonctions. Ce concept est très proche de la notion de *Blocks* que l'on trouve en Objective-C ou des *Lambdas* présentes dans d'autres langages. Pour voir un exemple d'utilisation des Closures, prenons la fonction *sorted* présente dans la bibliothèque standard de Swift qui permet de trier un tableau de données. Cette fonction prend deux arguments qui permettent de spécifier le tableau de données à trier, et une *Closure* permettant de définir la règle de tri de données. L'utilisation telle qu'on pourrait la retrouver dans n'importe quel langage serait celle-ci :

```
let cities = ["Paris", "Rouen", "Meudon", "Lille", "Le Havre"]
func backwards(s1 : String, s2 : String) -> Bool [
  return s1 > s2
}
var sortedCities = sorted(cities, backwards)
```

Bien que fonctionnelle, cette syntaxe peut être longue à écrire car nécessitant de créer une fonction qui est ensuite appelée dans le code. Le principe des Closures permet d'écrire le code suivant :

```
let cities = ["Paris", "Rouen", "Meudon", "Lille", "Le Havre"]
var sortedCities = sorted(cities, { (s1: String, s2: String) -> Bool in
  return s1 > s2
})
```

Certains trouveront ce code moins lisible que la première version présentée. D'autres trouveront qu'elle a le mérite d'être plus concise et plus simple à comprendre, car l'intégralité du fonctionnement du tri de notre tableau de noms de villes, est situé au même endroit dans le code.

Protocols

Les *Protocols* permettent de définir un modèle comprenant un certain nombre de propriétés et de méthodes qui devront être implémentées au sein des éléments (classes, structures, énumérations...) souhaitant se conformer à ce protocole. Ce concept est similaire à la notion d'interfaces que l'on retrouve dans d'autres langages. La déclaration d'un protocole se fait de la manière suivante :

```
protocol Person {
  var firstName : String { get set }
  var lastName : String { get set }
  func sendMessage(message: String)
}
```

Une fois le protocole défini, son adoption par un élément se fait de la manière suivante :

```
class Human : Person {
  var firstName : String
  var lastName : String
  func sendMessage(message: String) {
    println("Envoi du message : \(message)")
  }
}
```

Par défaut, toutes les informations définies dans un protocole doivent être implémentées dans les éléments qui adoptent ce protocole. Si toutes les informations contenues ne sont pas obligées d'être implémentées, il est possible de définir certaines d'entre elles comme optionnelles.

```
protocol Person {
  var firstName : String { get set }
  optional func sendMessage(message: String)
}
```

Extensions

Les extensions permettent d'enrichir les classes, énumérations ou structures existantes à l'aide de nouvelles propriétés et/ou méthodes. Ce mécanisme est très utile, notamment concernant les classes, lorsque l'on souhaite ajouter des informations à celles-ci sans avoir à implémenter d'héritage. La création d'une extension se fait d'une manière très simple :

```
extension Human {
  var fullName: String { return firstName + " " + lastName }
}
```

Property Observers

Les *Property Observers* permettent de pouvoir intercepter les changements de valeur d'une propriété afin d'effectuer des traitements particuliers, avant ou après que la valeur ne soit réellement modifiée.
Pour mettre ce mécanisme en place, il suffit d'ajouter la syntaxe suivante au sein des propriétés devant tirer parti de cette possibilité :

```
class Human {
  var age: Int = 0 {
     willSet(newAge) {
      println("L'âge après modification sera de \(newAge)")
     }
    didSet {
      println("Avant modification, l'âge était de \(oldValue)")
Surprise de la dernière et traditionnelle Worldwide Developer
Conference (WWDC 2014) organisée tous les ans par Apple, le
langage Swift débarque sur iOS et OS X sans crier gare.
    }
  }
}
```

La partie *willSet* est appelée juste avant la modification de la valeur, tandis que la partie *didSet* est appelée juste après la modification. A noter que ces méthodes ne sont pas appelées lors de la phase d'initialisation d'un objet.

Access Control

En Objective-C, et de par son héritage du C qui faisait que la déclaration des classes (entre autres) était effectuée dans un fichier en-tête (.H) distinct de l'implémentation de celles-ci (.M), il était assez simple de définir quelles propriétés et méthodes étaient accessibles ou non à un développeur. Swift ne fait plus la distinction entre définition et implémentation d'un objet. Toutes les informations sont contenues au sein du même

fichier (.SWIFT). Il fallait donc trouver un moyen pour définir la visibilité.
Les créateurs de Swift ont donc introduit des mots-clés dédiés à cet effet dans le langage. Pour cela, trois mots sont à disposition des développeurs à savoir *internal*, *private* et *public*. Ces mots doivent être positionnés devant un élément (classe, structure, énumération, protocole...) afin de définir le niveau de visibilité de celui-ci.

```
public class Human {
  public var firstName: String
  public var lastName: String
  internal var age: Int
  private var address: String
}
```

La valeur public indique que l'élément est visible partout au sein de l'application et même en dehors (ex : si la classe est définie dans un framework). La valeur *private* indique que l'élément n'est visible qu'au sein de l'élément où il a été déclaré (ex : dans un fichier, dans une classe...).
La valeur *internal* indique que l'élément est visible partout au sein de l'application où il a été déclaré.
Si ces attributs ne sont pas précisés lors de la déclaration d'un élément, c'est la valeur *internal* qui est implicitement utilisée par Swift.

MES PREMIERS DÉVELOPPEMENTS AVEC SWIFT

Voyons maintenant comment mettre tout cela en pratique en réalisant nos premiers développements à l'aide de Swift.

Développement Swift avec Xcode 6

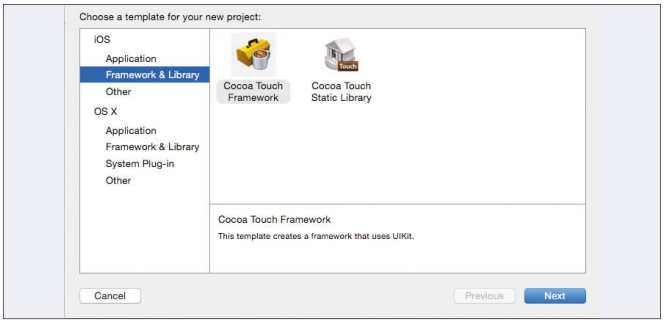
Il faut tout d'abord noter qu'il est obligatoire d'avoir installé Xcode 6 pour utiliser Swift. Comme à leur habitude, les équipes Apple introduisent systématiquement les nouveautés dans les dernières versions de leurs OS et de leurs outils et n'assurent que rarement la compatibilité avec les versions antérieures. Pour vous procurer Xcode 6, il suffit de vous rendre sur le Mac App Store puis de télécharger et d'installer l'environnement de développement commun à toutes les plateformes Apple. Une fois celui-ci installé sur votre ordinateur, il suffit de l'exécuter pour pouvoir commencer à utiliser Swift.
Pour commencer à écrire du code avec Swift, deux possibilités s'offrent à vous selon ce que vous souhaitez faire. La première dont nous reparlerons consiste à créer une application (iOS ou OS X) et dans ce cas vous demanderez de créer un nouveau projet Xcode.
La deuxième possibilité qui s'offre à vous, qui est une des grandes nouveautés de Xcode 6, concerne la possibilité de créer un « Playground » qui vous permettra de pouvoir utiliser le langage Swift à des fins de tests mais de manière interactive.

Présentation de « Playgrounds »

Lorsque vous exécutez Xcode 6 et que vous sélectionnez l'option « Get started with a Playground » dans la boîte de dialogue qui est affichée par défaut au lancement, vous êtes guidé pour créer un fichier portant l'extension « .PLAYGROUND ».
Ce fichier un peu particulier vous permet, une fois créé et ouvert dans Xcode, de pouvoir commencer à taper du code Swift, puis d'exécuter celui-ci, et de voir instantanément le résultat de chaque ligne sans avoir à compiler puis lancer une application.
Certains parmi vous vont certainement se demander à quoi peut bien servir de taper du code si ce n'est pour créer une vraie application qui pourra ensuite être exécutée sans avoir recours à Xcode.
Eh bien en fait cette fonctionnalité est très utile à bien des égards, au premier rang duquel l'apprentissage de Swift. En effet, comme avec tout nouveau langage de développement que vous découvrez, vous allez faire des erreurs dans la syntaxe du code que vous écrirez, vous ferez des erreurs de logique dans la construction de vos algorithmes, vous voudrez voir à quoi ressemble le résultat de telle ou telle portion de code indépendamment du reste, etc...
Playgrounds vous fournit un « bac à sable » dans lequel vous pourrez effectuer tous les tests que vous désirez, sans avoir la lourdeur de tout ce qui entoure une véritable application comme la construction d'une interface graphique, la compilation du code, l'exécution et le débogage avec des outils compliqués, etc...
Outre la possibilité de vous permettre d'expérimenter votre code Swift, Playgrounds possède de nombreux outils vous permettant de mieux appréhender le fonctionnement de ce que vous écrivez en proposant des outils visuels pour répondre à vos besoins.
Ainsi il est possible de positionner à côté de l'éditeur de code, une « Timeline » qui vous permettra par exemple de voir l'évolution des valeurs de vos variables dans le temps. Cette fonctionnalité est très intéressante lorsque l'on développe des algorithmes mathématiques basés sur une notion temporelle.
De la même manière que la Timeline, il est possible d'avoir un rendu visuel des éléments que vous manipulez par programmation. Si par exemple vous chargez des images et que vous souhaitez avoir un rendu de ces éléments à chaque étape d'exécution de votre code, vous pouvez demander à l'éditeur du Playground d'afficher les éléments que vous désirez suivre.
Si vous êtes amenés à développer des scènes animées pour la réalisation de jeux par exemple, vous pourrez voir ces mêmes scènes fonctionner au sein de votre bac à sable sans avoir à créer toute l'application qui va autour. C'est notamment cet exemple qu'Apple a présenté lors de la WWDC 2014. Comme tous les systèmes de bac à sable qui existent sur d'autres plateformes, cette fonctionnalité possède des limitations parmi lesquelles l'impossibilité d'interagir avec l'utilisateur pour qu'il saisisse des données ou utilise une interface graphique, l'exécution du code saisi dans le bac à sable sur un iPhone/iPad, l'utilisation de frameworks tiers, etc... Ces limitations seront certainement levées au fur et à mesure des versions qui sortiront dans les prochains mois et les prochaines années car Playgrounds n'en est qu'à sa toute première version qui est déjà très aboutie mais surtout pleine de promesses.

Les différents types de projets

Outre l'utilisation de Playgrounds pour développer avec Swift au sein de Xcode 6, il existe d'autres typologies de projets permettant de pouvoir mettre en pratique le nouveau langage créé par Apple.
Commençons par évoquer les projets dédiés à la création d'applications iOS. Xcode 6 propose désormais cinq modèles de projets fournissant chacun une base de départ pour réaliser des applications à destination de la plateforme mobile : *Master-Detail Application, Page-Based Application, Single View Application, Tabbed Application* et *Game*. Tous ces modèles de projets permettent de pouvoir utiliser au choix Objective-C et/ou Swift



comme langage de développement. Nous disons bien Objective-C et/ou Swift car il est tout à fait possible de mixer les deux langages au sein d'une même application afin de faciliter la phase de transition d'un langage vers l'autre. Transition car il semble clair qu'Apple mise énormément sur Swift pour les années à venir mais l'écosystème Objective-C étant tellement important, il serait compliqué de faire comprendre aux développeurs ayant misé sur ce langage depuis de nombreuses années, que tout soit remis en cause dès aujourd'hui.

Outre les modèles d'applications iOS, un autre modèle de projet permet de s'adonner aux joies de Swift. Ce modèle est une nouveauté de Xcode 6 visible dans la section *Framework & Library*.

Il est désormais possible de créer des *Cocoa Touch Framework* alors que précédemment, seules les *Cocoa Touch Static Library* étaient disponibles. Il était possible de créer des frameworks au sein de Xcode 5 mais uniquement pour OS X.

La particularité des frameworks par rapport aux librairies, concerne le fait de pouvoir intégrer autre chose que du code à l'intérieur. Parmi les éléments que l'on retrouve souvent au sein des frameworks, nous pouvons citer des ressources comme les listes de chaînes de caractères traduites dans différentes langues ou bien encore des images, icônes et autres fichiers sonores.

Comme pour les modèles d'applications iOS, il est possible de mélanger Objective-C et Swift au sein des frameworks Cocoa Touch. Une bonne chose pour les personnes qui développent ce type de composants et qui étaient jusqu'ici limitées par les librairies.

Concernant les librairies justement, elles ne permettent pas à ce jour de pouvoir utiliser Swift comme langage de développement. Celles-ci sont donc pour l'instant toujours cantonnées à l'utilisation d'Objective-C, et il n'est pas sûr que cette situation change dans le futur vu que les frameworks répondent au même besoin en proposant plus de souplesse/possibilités.

Ma première application iOS en Swift

Afin de voir en pratique comment utiliser Swift pour réaliser une application, nous allons créer un projet iOS. Pour cela nous allons partir du modèle de projet Single View Application, et voir comment enrichir celui-ci avec une des fonctionnalités les plus répandues dans les applications iOS, à savoir, l'utilisation d'une vue d'une liste de données (UITableView) comme on peut en retrouver dans l'application de messagerie fournie en standard depuis la toute première version d'iOS.

Nous ne détaillerons pas toutes les étapes de création de l'application, notamment comment utiliser l'éditeur d'interface graphique de Xcode. Nous n'expliquerons pas non plus toute la mécanique de fonctionnement d'une application iOS qui reste identique entre Objective-C et Swift (de très nombreux tutoriaux existent en Objective-C traitant de ce sujet). Nous nous concentrerons sur ce qui nous intéresse au premier plan à savoir le code Swift nécessaire au bon fonctionnement de l'application.

Une fois le projet créé, en spécifiant bien d'utiliser Swift comme langage de développement, nous retrouvons dans Xcode 6 un projet avec une structure identique à celle-ci.

La première chose que nous constatons et comme évoqué précédemment, il n'existe plus de distinction entre fichiers en-tête (.H) et fichiers d'implémentation (.M). Dans les projets utilisant Swift, la déclaration d'un type et son implémentation se trouvent dans un seul et unique fichier (.SWIFT). Dans le projet que nous avons créé, il existe deux fichiers portant cette extension : *AppDelegate.swift* et *ViewController.swift*. Le premier concerne le délégué de notre application qui est appelé lors de divers évènements (lancement de l'application, suspension de l'application, arrêt de l'application...). Le second concerne le contrôleur de la vue principale de l'application. Si nous regardons le contenu du fichier *AppDelegate.swift*, celui-ci commence par importer le framework UIKit afin de pouvoir manipuler les classes se trouvant à l'intérieur (UIApplication, UIWindow, UIView...).

Une classe est ensuite définie, et déclarée comme point d'entrée de l'application à l'aide de l'attribut @UIApplicationMain positionné juste devant. Cet attribut remplace le fichier *Main.m* qui existe dans les projets écrits Objective-C.

La classe définie dans ce fichier hérite de la classe de base *UIResponder* et indique se conformer au protocole *UIApplicationDelegate* qui sert à intercepter les évènements applicatifs évoqués précédemment.

Sont ensuite déclarées et implémentées toutes les fonctions que nous souhaitons utiliser et définir au sein du protocole *UIApplicationDelegate*.

```
import UIKit
@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {
 var window: UIWindow?
 func application(application: UIApplication, didFinishLaunching
WithOptions launchOptions: [NSObject: AnyObject]?) -> Bool {
   return true
 }
 func applicationWillResignActive(application: UIApplication) {
 }
 func applicationDidEnterBackground(application: UIApplication) {
 }
 func applicationWillEnterForeground(application: UIApplication) {
 }
 func applicationDidBecomeActive(application: UIApplication) {
 }
 func applicationWillTerminate(application: UIApplication) {
 }
}
```

Le fichier *ViewController.swift* contient pour sa part la structure basique d'un contrôleur, à savoir une classe héritant de la classe de base *UIViewController* ainsi que la surcharge des deux méthodes *viewDidLoad* et *didReceiveMemoryWarning*.

```
import UIKit
class ViewController: UIViewController {
 override func viewDidLoad() {
   super.viewDidLoad()
 }
 override func didReceiveMemoryWarning() {
   super.didReceiveMemoryWarning()
 }
}
```

Nous avons ajouté au sein du contrôleur créé par défaut avec l'application, une liste de données (UITableView) que nous avons paramétrée de telle manière que la source de données et le délégué soient le contrôleur. Reste maintenant à écrire le code Swift permettant à notre liste d'afficher des données. Cela commence par modifier la déclaration du contrôleur afin que celui-ci soit conforme aux protocoles *UITableViewDataSource* et *UITableViewDelegate* et que notre liste de données soit associée à une variable nous permettant de pouvoir la manipuler.

```
class ViewController: UIViewController, UITableViewDataSource,
 UITableViewDelegate {
 @IBOutlet weak var myTableView: UITableView!
}
```

L'attribut *@IBOutlet* permet de faire le lien avec le concepteur graphique

de Xcode. L'attribut *weak* a la même particularité qu'en Objective-C et nous aborderons celui-ci ultérieurement dans la section traitant de l'interopérabilité entre Objective-C et Swift. Afin de pouvoir créer des cellules de données dans notre liste, nous devons définir à partir de quel type ces cellules seront instanciées ultérieurement. Pour cela nous modifions la méthode *viewDidLoad* pour enregistrer des cellules classiques (*UITableViewCell*). Vous constaterez que pour identifier la classe associée à un type de données, nous utilisons la syntaxe *UITableViewCell.self*. L'équivalent de cette syntaxe en Objective-C est *[UITableViewCell class]*. On remarque également que la particularité d'Objective-C dans l'appel des méthodes, avec des paramètres nommés (*forCellReuseIdentifier*), a été conservée sur Swift.
Le but de cette syntaxe étant de rendre le code plus compréhensible au lieu de simplement se baser sur l'ordre des paramètres dans l'appel d'une méthode comme le font d'autres langages.

```
override func viewDidLoad() {
  super.viewDidLoad()
  myTableView.registerClass(UITableViewCell.self, forCellReuse
Identifier: "MyCell")
}
```

Pour que notre liste de données puisse afficher des données, et pour nous conformer au protocole UITableViewDataSource, nous devons implémenter les deux méthodes suivantes. La première permet de définir combien de lignes possèdera la liste. La deuxième permet de créer chacune des cellules en fonction des paramètres qui lui sont passés.
On notera tout particulièrement dans la deuxième méthode, l'utilisation du point d'interrogation derrière la propriété *textLabel* de la variable *cell*. Cette syntaxe permet de tester, pour les propriétés définies comme *optional*, si elles possèdent ou non une valeur. Si elles ne possèdent pas de valeur, le traitement du code ne va pas plus loin afin d'éviter de déclencher des exceptions à l'exécution puis passe à la ligne de code suivante.

```
func tableView(tableView: UITableView, numberOfRowsInSection
 section: Int) -> Int {
 return 10
}
func tableView(tableView: UITableView, cellForRowAtIndexPath
 indexPath: NSIndexPath) -> UITableViewCell {
 var cell: UITableViewCell = tableView.dequeueReusableCell
WithIdentifier("MyCell", forIndexPath: indexPath) as UITable
ViewCell
 cell.textLabel?.text = "Cellule n° \(indexPath.row)"
 return cell
}
```

Reste ensuite à implémenter la méthode du protocole *UITableViewDelegate* permettant de déterminer quelle ligne l'utilisateur a sélectionné sur la liste.

```
func tableView(tableView: UITableView, didSelectRowAtIndex
Path indexPath: NSIndexPath) {
 println("Vous avez sélectionné la cellule n°\(indexPath.row)")
}
```

Nous possédons désormais une application pleinement fonctionnelle, contenant une liste de données affichant 10 lignes, chaque ligne affichant un texte. Lorsque l'utilisateur sélectionne une ligne, un message est affiché dans la console de débogage de Xcode.
L'utilisation des objets définis au sein d'UIKit et plus généralement des frameworks fournis par Apple avec iOS, se fait de manière similaire si nous comparons ce code à son équivalent en Objective-C. Seule la syntaxe spécifique à Swift diffère mais fonctionnellement, les applications sont capables de reproduire les mêmes comportements.

Utilisation du framework Foundation

Comme nous venons de le voir avec la réalisation d'une première application iOS, l'utilisation d'UIKit qui à l'origine a été réalisée pour être utilisée en Objective-C, ne pose pas de souci particulier lors de son utilisation avec Swift. Mais qu'en est-il du framework Foundation, dont la plupart des classes sont utilisées pour réaliser bon nombre d'opérations telles que les communications réseaux, le stockage d'informations dans des bases de données, la manipulation du système de fichiers, l'exécution de tâches de manière asynchrone, etc... ?
Eh bien on peut dire qu'Apple a fait les choses correctement car l'intégralité des classes définies dans Foundation peuvent être utilisées avec Swift. On peut notamment voir cela en lisant la documentation disponible dans Xcode dans laquelle on se rend compte que systématiquement les déclarations des classes, propriétés, méthodes, etc... sont disponibles en Objective-C et en Swift.
Prenons l'exemple des classes permettant d'effectuer des requêtes vers des serveurs Web (NSURLSession). La documentation laisse apparaître ceci pour instancier cette classe :

Objective-C

```
+ (NSURLSession *)sessionWithConfiguration:(NSURLSessionConfiguration
 *)configuration
```

Swift

```
init(configuration configuration: NSURLSessionConfiguration)
 -> NSURLSession
```

Mais sans aller jusqu'à des choses aussi poussées, il en est de même par exemple pour les méthodes de la classe NSString. Le type String qui existe avec Swift possède en quelque sorte un pont vers son équivalent Foundation. Toutes les méthodes disponibles sur la classe NSString, s'appliquent au type String.

Objective-C

```
- (NSString *)substringFromIndex:(NSUInteger)anIndex
```

Swift

```
func substringFromIndex(_ anIndex: Int) -> String
```

La possibilité d'utiliser l'intégralité de Foundation au sein de son code Swift, permet d'ores et déjà d'offrir une grande richesse fonctionnelle à un langage qui est encore très jeune et qui ne couvre pas l'ensemble des besoins des développeurs.
En effet, et comme nous le verrons ultérieurement, la bibliothèque des fonctions standard de Swift est encore assez pauvre. Si vous souhaitez par exemple effectuer des requêtes vers un serveur HTTP afin de récupérer un flux JSON/XML pour ensuite parser celui-ci en vue de son traitement, aucune classe n'est disponible pour l'heure dans Swift afin de répondre à ce besoin.
En revanche, ces classes étant offertes par Foundation, vous avez la possibilité de les utiliser au sein de vos applications Swift afin de réaliser tout ce dont vous avez besoin.

Guide d'utilisation de Swift

Interopérabilité entre Swift et Objective-C

Il est possible de faire cohabiter Objective-C et le nouveau langage. Cela est valable dans les deux sens, à savoir, utiliser du code Swift dans des applications Objective-C ou bien utiliser des librairies Objective-C au sein d'applications Swift.
Mais bien évidemment tout n'est pas magique et certains concepts propres à chaque langage doivent être respectés afin que la cohabitation

se passe sans soucis. Toutefois même en respectant toutes les règles, tout n'est pas possible en fonction de ce que vous souhaitez faire.
Par exemple, et pour ne citer que quelques cas, si vous utilisez certaines des nouveautés de Swift telles que les *Generics*, les *Tuples*, les alias de types ou les types imbriqués, votre code ne pourra être utilisé au sein d'applications Objective-C car ces fonctionnalités n'ont pas d'équivalent dans ce langage.
D'autres éléments sont à prendre en considération, et nous allons aborder certains des facteurs importants qui facilitent l'interopérabilité.

Automatic Reference Counting (ARC)

Le système ARC a été introduit il y a plusieurs années, au sein d'Objective-C afin de simplifier la vie des développeurs en ce qui concerne toute la partie gestion mémoire. Dans sa conception interne, Swift a été basé sur ce même mécanisme afin de gérer, à la place du développeur, la libération de la mémoire lorsqu'une donnée n'est plus utilisée. Le fonctionnement d'ARC est normalement bien connu des développeurs Objective-C mais un peu moins des développeurs qui découvriraient Swift. Globalement ARC fonctionne sur le principe du comptage de références d'objets pour déterminer si la mémoire allouée à une donnée peut être libérée ou non. Si un objet est utilisé par deux autres objets, le premier considère qu'il a deux références pointant sur lui. Tant que ce compte n'est pas redescendu à zéro, ARC considère que la mémoire ne doit pas être libérée car elle reste utilisée.
Là où cela se complexifie, c'est lorsque les objets ont des références mutuelles, ce qui peut amener à des fuites mémoire, car une référence cyclique est créée et la mémoire n'est jamais libérée. Cela amène généralement à ce que l'application plante à un moment donné, faute de mémoire disponible sur l'ordinateur qui exécute celle-ci. Afin de résoudre cela, des mots clés tels que strong et weak ont été introduits dans Objective-C afin que le développeur puisse aider ARC à déterminer qui détient une référence forte (l'objet référenceur originel) et qui détient une référence faible (l'objet référencé). On retrouve ces mêmes principes au sein de Swift. Par défaut, et si un objet en référence un autre, celle-ci est considérée comme une référence forte (strong). Cet exemple se traduit lorsque l'on déclare des classes par quelque chose ressemblant à ceci.

```
class Person {
  var apartment: Apartment?
}
class Apartment {
  var tenant: Person?
}
```

Si le développeur ne fait pas attention à la manière dont son code fonctionne, ce cas peut engendrer des fuites mémoire selon la manière dont les objets sont libérés lorsqu'ils ne sont plus utilisés. Afin de laisser ARC gérer la mémoire, on peut modifier la déclaration de la manière suivante :

```
class Person {
  var apartment: Apartment?
}
class Apartment {
  weak var tenant: Person?
}
```

Avec cette déclaration, lorsque l'objet Person est libéré, et si aucune autre référence ne pointe dessus, l'objet Apartment associé à celui-ci est automatiquement et instantanément libéré par ARC. Cela améliore évidemment la performance globale du système puisque vous rendez au système d'exploitation la mémoire dont vous n'avez plus besoin.
Un autre mot clé a été introduit avec Swift pour les cas où l'objet référencé possède toujours une valeur (weak acceptant les valeurs nil). Dans ce cas il est possible de modifier les déclarations des classes comme suit :

```
class Person {
  var apartment: Apartment?
}
class Apartment {
  unowned var tenant: Person
}
```

On voit que la classe *Apartment* possède une référence vers *Person* mais que cette valeur ne peut être *nil*, la propriété n'étant plus marquée comme *optional* via le point d'interrogation après le type de donnée.

Données de type « *id* »

Contrairement à Swift, Objective-C n'est pas un langage à typage fort. Cela signifie qu'il est possible d'assigner un type de donnée indéterminé aux variables. Cela se fait via l'utilisation du mot clé *id* qui permet de stocker un pointeur vers n'importe quelle typologie d'information.
Swift étant un langage à typage fort, Apple a dû trouver une solution afin de permettre aux deux langages de pouvoir cohabiter lorsque des variables étaient déclarées comme *id*. Pour ce faire, un type de donnée particulier existe avec Swift à savoir le type *AnyObject*.
Ainsi attendez-vous, lorsque vous manipulerez avec Swift des données provenant de librairies écrites en Objective-C, à ce que le type de donnée renvoyé soit *AnyObject* quand ces librairies n'utilisent pas de typage fort. Charge au développeur ensuite de connaître le type de donnée manipulé et de faire les opérations de *casting* nécessaires.

Exposer une classe Swift à Objective-C

Lorsque vous déclarez des classes au sein d'une application ou d'une librairie écrite avec Swift, celle-ci est directement manipulable en Objective-C sans avoir quoi que ce soit de particulier à faire pour que cela fonctionne. Toutefois lorsque l'on se penche plus en détail sur le fonctionnement de ces classes dans du code Objective-C, on se rend compte que tout n'est pas si simple que cela. C'est notamment le cas lorsque l'on utilise le runtime Objective-C, qui permet, entre-autres, de pouvoir effectuer des opérations d'introspection (également nommé reflection dans d'autres langages). Prenons l'exemple d'une classe déclarée de telle manière au sein d'une application :

```
public class SwiftObject {
  public func demoSwift() { … }
}
```

Si l'on souhaite pouvoir utiliser cette classe au sein d'une application Objective-C, il est nécessaire de la référencer en utilisant une convention de nommage du type *ModuleName.ClassName* (où ModuleName représente le nom de votre application/librairie).
Cela n'est pas très pratique, surtout lorsque l'on est habitué à utiliser uniquement le nom de la classe pour créer une instance du type en question. On peut aussi vouloir que le nom de la classe, le nom des propriétés ou des méthodes, soient différents en Objective-C et en Swift pour, par exemple, éviter des conflits de noms entre du code existant et les nouveaux développements écrits à l'aide de Swift.
Apple a pensé à ces cas de figure et a rendu cela possible en ajoutant des attributs devant le nom de la classe, des propriétés ou des méthodes comme on peut le voir ci-dessous :

```
@objc(MySwiftObject)
public class SwiftObject {
```

```
  @objc(mySwiftMethod)
  public func demoSwift() { … }
}
```

Les exemples cités ici ne sont que quelques cas à prendre en considération quand on parle d'interopérabilité entre Swift et Objective-C car beaucoup d'autres sujets tels que la comparaison d'objets, l'extension de classes, l'intégration avec l'environnement Xcode, l'utilisation de framework comme CoreData, le transtySurprise de la dernière et traditionnelle Worldwide Developer Conference (WWDC 2014) organisée tous les ans par Apple, le langage Swift débarque sur iOS et OS X sans crier gare. page de données, etc... sont des sujets vastes qui pourraient faire l'objet d'articles entiers pour les aborder en détail.

Passage en revue de fonctions de la bibliothèque standard Swift

Parmi les choses qui font le succès d'un nouveau langage de développement, la bibliothèque des fonctions offertes en standard au sein de celui-ci est sans nul doute un élément important. Comme nous l'avons évoqué précédemment, Apple se repose en grande partie sur le framework Foundation pour apporter de très nombreuses fonctionnalités à Swift telles que les communications réseaux, parser des flux JSON/XML, interagir avec le système de fichiers, manipuler les chaînes de caractères, etc... Toutefois, ce n'est pas pour autant que Swift n'offre aucune fonction propre à ce nouveau langage. Même si celles-ci ne sont pas toutes présentes dans la documentation disponible avec Xcode, de nombreuses fonctions (près d'une centaine) sont à disposition des développeurs pour effectuer certaines tâches couramment utilisées dans le développement d'applications. Afin d'en avoir la liste exhaustive, il est possible dans Xcode d'utiliser le raccourci Cmd+Clic sur le nom d'une fonction pour accéder à la déclaration de toutes les fonctions contenues dans le même module (la même librairie) que celle contenant la fonction sur laquelle vous avez cliqué. Voyons quelques-unes d'entre elles ainsi que leur syntaxe :

abs(signedNumber)

Retourner la valeur absolue d'un nombre

```
abs(-1) == 1
abs(-42) == 42
abs(42) == 42
```

contains(sequence, element)

Retourne True si l'élément est contenu dans la séquence (tel qu'un tableau)

```
var languages = ["Swift", "Objective-C"]
contains(languages, "Swift") == true
contains(languages, "Java") == false
```

countElements(collection)

Retourne le nombre d'éléments contenu dans la collection

```
countElements("Swift") == 5
```

find(sequence, element)

Retourne l'index de l'élément au sein de la séquence ou nil si non trouvé

```
var languages = ["Swift", "Objective-C"]
find(languages, "Objective-C") == 1
find(languages, "Java") == nil
```

map(sequence, transformClosure)

Retourne une nouvelle séquence en appliquant à chaque élément de celle-ci le code contenu dans la Closure passée en paramètre

```
map(1...3, { $0 * 5 }) ==  [5, 10, 15]
```

reduce(sequence, initial, combineClosure)

Retourne une valeur unique en réduisant les éléments à l'aide de la Closure passée en paramètre

```
var languages = ["Swift", "Objective-C"]
reduce(languages, "", { $0 + $1 }) == "SwiftObjective-C"
```

reverse(sequence)

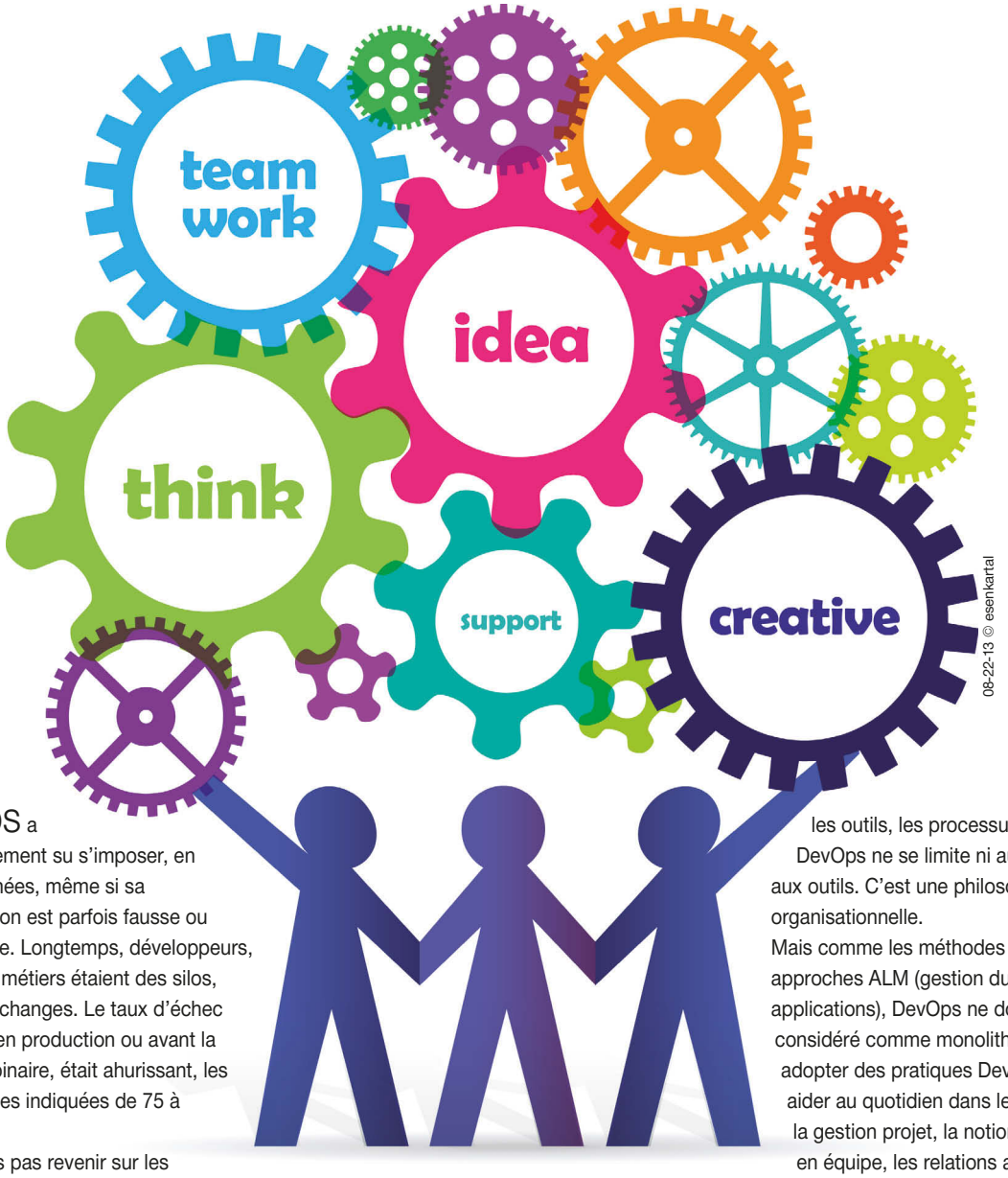
Retourne une séquence dans l'ordre inverse de la séquence initiale

```
reverse([1, 2, 3]) == [3, 2, 1]
```

Mon verdict sur Swift

La question, que se posent beaucoup de développeurs et celle que je me suis évidemment posée quand un nouveau langage est disponible, est : « dois-je m'y mettre, et qu'est-ce que cela va m'apporter de plus que ce que j'utilise déjà (C#, Java, Python, JavaScript...) ? ».
En tant que développeur ayant utilisé de nombreux langages tels que C++, Delphi, Visual Basic en passant par C# et Objective-C, la première chose que j'ai appréciée est que l'on ne se sent pas perdu. Swift ayant été bâti en reprenant les meilleurs concepts de nombreux langages, on retrouve rapidement ses habitudes quand il s'agit de manipuler des classes, définir des attributs privés/publics, surcharger des méthodes, etc...
Passés les premiers tests basiques, j'ai commencé à vouloir pousser le langage un peu plus loin en réalisant de véritables applications qui effectuent des traitements asynchrones, se connectent à des serveurs Web, manipulent des flux JSON/XML, stockent des informations dans une base de données, etc...
C'est à ce moment que l'on s'aperçoit que le langage est encore récent. Outre le manque de documentation « avancée », on se retrouve vite face à des cas où ce qui parait simple et rapide à mettre en place dans d'autres langages, car il existe des classes/fonctions permettant de répondre au besoin, devient plus obscur à mettre en œuvre. La bibliothèque de fonctions offertes par Swift est encore « pauvre » à cause de la jeunesse du langage. Apple répond à cela en préconisant d'utiliser Foundation / Cocoa / Cocoa Touch, qui, il est vrai, permettent d'enrichir Swift avec de nombreuses fonctionnalités.
La chose est encore plus vraie lorsque l'on a besoin de concepts avancés. J'ai notamment besoin dans certains cas (ex : développement de frameworks) de pouvoir effectuer de l'introspection (reflection). Bien qu'un embryon de solution soit offert dans Swift, on atteint vite les limites du système. Les API équivalentes du runtime Objective-C qui peuvent être utilisées, ne couvrent pas tous les cas de figures (ex : generics, optionals...). Mais tout n'est pas noir pour autant, bien au contraire. Avoir dans Swift des concepts tels les generics, les optionals, les lambdas, l'inférence des types, etc... apporte à ce nouveau langage ce que les développeurs Objective-C réclamaient depuis des années. Certes il aura fallu attendre un nouveau langage pour avoir tout cela à disposition mais c'est enfin là, et certainement pour de nombreuses années. Même si tout n'est pas parfait à l'heure actuelle, Swift est un langage jeune (annoncé il y a seulement 5 mois) mais prometteur. Il permet d'ores et déjà de développer des applications iOS et OS X, tout en alliant la puissance que l'on retrouve dans les langages phares du marché mais qui ont pour la plupart 10 à 15 ans d'existence.

DevOps : ce n'est pas un outil, mais une philosophie



DevOps a incontestablement su s'imposer, en quelques années, même si sa compréhension est parfois fausse ou approximative. Longtemps, développeurs, productions, métiers étaient des silos, avec peu d'échanges. Le taux d'échec des projets, en production ou avant la livraison du binaire, était ahurissant, les études menées indiquées de 75 à 90 %.

Nous n'allons pas revenir sur les causes profondes des échecs, des retards des projets informatiques. Depuis l'apparition des Google, Facebook, Twitter, Netflix, etc., les méthodes de développement et la manière de déployer les nouveaux codes/fonctions (la fameuse « release ») ont été radicalement transformées. On parle de développement continu, d'intégration continue. Bref, la notion même de version est dépassée. Bien entendu, cette approche radicale du développement ne peut pas être adoptée par tous les développeurs, toutes les applications. Mais on peut s'inspirer de cette méthode pour fluidifier le développement et les notions de builds. Et faciliter le déploiement de l'application.

Le Cloud Computing, les apps mobiles ont déjà modifié les méthodes de travail : toujours plus souple, toujours plus vite, au risque, d'aller trop vite et de couper dans les fondamentaux de la programmation, comme la sécurité, la structure et la propreté du code. DevOps est un pont entre le développeur et l'opérationnel/la production. Il faut alors agir sur trois éléments (en référence à la réflexion de Patrick Debois, créateur du mot DevOps) :

« DevOps est réel, mais ne croyez pas que c'est juste en achetant un outil ».

(Luke Kanies (Puppet Labs)

les outils, les processus, les gens.

DevOps ne se limite ni aux personnes ni aux outils. C'est une philosophie globale et organisationnelle.

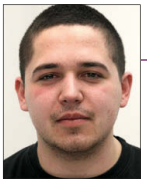
Mais comme les méthodes agiles ou les approches ALM (gestion du cycle de vie des applications), DevOps ne doit pas être considéré comme monolithique. Vous pouvez adopter des pratiques DevOps pour vous aider au quotidien dans le développement, la gestion projet, la notion de build, le travail en équipe, les relations avec les autres personnes (production, utilisateur, métier, etc.). Nous pouvons appeler cela le « DevOps à la demande » ou « DevOps à la carte ».

Vous l'aurez compris, il faut agir plusieurs leviers à la fois. Même quand on travaille seul ou dans une petite équipe, vous pouvez mettre en place des éléments DevOps comme les outils et des processus de fonctionnement plus fluides. Mais il faut un volontarisme des différentes personnes. On peut parler d'un ADN DevOps à injecter.

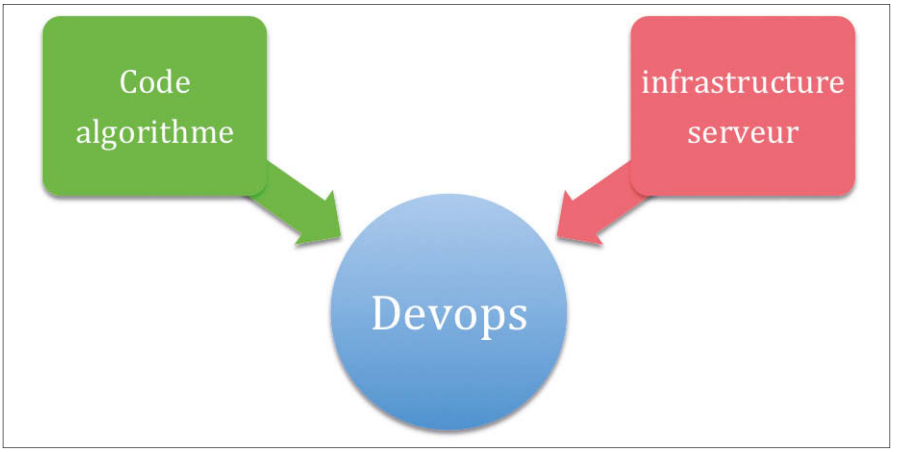
Dans ce dossier, nous allons revenir sur la notion même de DevOps [bonnes pratiques, outils, mise en œuvre] et voir des cas concrets d'usage que vous pouvez reproduire et adapter.

Quelques rappels

Depuis peu, le principe de devops se fait de plus en plus une place dans les entreprises. On entend aussi parler d'ingénieurs fullstack. D'où viennent ces principes? Quels sont les rôles de ces nouveaux postes ?



Steven Pojer,
Responsable système et réseau
ETNA



Développeur vs adminsys

Souvent, les développeurs et les administrateurs système sont séparés, voir cloisonnés. Une frontière se place toujours à un moment sur la chaîne de la création d'une solution. Pourtant, pour la plupart des projets, les deux doivent coexister, s'entraider, afin que chacun connaisse les besoins de l'autre. Mais où peut on définir la limite des deux rôles? Doit-il toujours y avoir cette séparation avec les évolutions technologiques d'aujourd'hui?

L'ère du Cloud et évolution technologique

Le Cloud Computing a apporté une nouvelle façon de concevoir l'infrastructure. De quelques serveurs multitâches, on passe à plusieurs services cloisonnés et optimisés. L'élasticité de la gestion des serveurs, privés ou publics, demande une gestion de plus en plus complexe. L'automatisation de la gestion des services est donc plus sophistiquée et amène des connaissances en développement non négligeables.

L'évolution technologique apporte de plus en plus de solutions pour répondre à une problématique. Les langages, les frameworks, les bases de données SQL ou NoSQL, ou d'autres services imposent au développeur d'avoir une meilleure connaissance de son projet, qui ne se limite plus à son code. La connaissance de plusieurs technologies, de la manière dont elles sont gérées et dont elles fonctionnent prend une place importante dans la réalisation du projet.

DevOps

DevOps, dev- pour développement et -ops pour « operations » (exploitation), amène l'idée que la frontière entre les dev et les ops ne devrait pas exister. On entend souvent dire qu'un bon développeur sait faire de l'administration système et qu'un bon administrateur système sait développer, ce qui est vrai la plupart du temps.
Bien sûr, dans les grosses structures, où chacun a un rôle bien précis, il sera difficile de s'écarter de son « scope » originel et qui force à établir des limites.
Mais il est intéressant de faire travailler toutes les ressources humaines ensemble sur un projet, en fonction des compétences de chacun, ce qui permet de promouvoir l'agilité de l'équipe.

Le principe devops repose sur une équipe mais aussi sur une personne. En effet, afin de faciliter la gestion de son infrastructure, un administrateur système se doit d'utiliser des technologies plus ou moins complexes afin d'automatiser la création et la configuration de ces serveurs. Script shell, Chef, Puppet, et d'autres encore, amènent un administrateur à "coder son infrastructure". D'autre part un développeur, afin de pouvoir s'assurer de la bonne mise en oeuvre de son logiciel, se doit de connaître les différents services et technologies existants. Pour ce dernier exemple, on peut parler d'ingénierie fullstack.

Fullstack

Un ingénieur fullstack a connaissance de toutes les couches qui composent un projet. Cette personne se doit de tout connaître afin de choisir les bonnes solutions et de les assembler afin de composer un projet. De l'administration système jusqu'au client en passant par la gestion des données et l'application, le but est d'avoir une vue complète du projet afin de garantir les bonnes connexions entre les différentes couches.

A l'ETNA, le mouvement devops est bien présent et fait prendre conscience du fait qu'un projet, même s'il est composé de plusieurs étapes différentes, se doit d'être conçu par une équipe qui a plusieurs rôles, et non par plusieurs équipes avec des rôles spécifiques.

Introduction à IBM Bluemix et IBM DevOps Services dans un exemple avec Node.js

Bluemix est une nouvelle solution d'IBM qui permet de construire, de déployer et de gérer tout type d'applications (mobile, smart devices, Internet des objets, web, big data, analytics).

Philippe THOMAS
IBM Bluemix Architect
La Gaude IBM Innovation Center | Ecosystem Development
Email : thomas1@fr.ibm.com

La solution est basée sur les standards ouverts du marché (Cloud Foundry) et il utilise une plate-forme Cloud (SoftLayer) pour déployer ces nouvelles applications. Parmi la longue liste de fonctionnalités de Bluemix, on notera la capacité de construire des back-end applicatifs pour les applications mobiles, le monitoring et l'analyse des performances intégrés ainsi que le dispositif de scalabilité automatique qui peut être ajouté à toute application pour lui permettre d'absorber une charge plus importante. Bluemix est une plate-forme ouverte car en plus des services IBM proposés (plus de 40), elle intègre également les services des partenaires ainsi que ceux de la communauté Open Source. L'un des avantages principaux de Bluemix est de permettre la construction et le déploiement très rapide d'une application et de ses services associés (base de données SQL ou non, file d'attentes, envoi de SMS, récupération de tweets, analytics, Watson Q&A, ...). Dans cet article, nous allons créer et déployer une application simple développée en JavaScript (avec Node.js).

Tout d'abord, il faut s'enregistrer pour obtenir un compte Bluemix. Cette opération est assez simple. Elle s'effectue à partir de la page d'accueil sur http://ibm.biz/bluemixeurope, en cliquant sur SIGN UP ou bien en cliquant sur le bouton central Sign Up for a free. Un email vous sera ensuite envoyé afin de valider votre inscription. Si vous disposez déjà d'un compte Bluemix, vous pouvez vous connecter en cliquant sur LOG IN en haut à droite Fig.1.

Une fois connecté au site, Bluemix vous offre une quantité importante de renseignements en cliquant sur Docs dans le menu en haut de l'écran. Nous vous conseillons les sujets suivants pour un apprentissage rapide :

- Bluemix Overview
- Creating Apps

Nous allons maintenant créer et déployer notre première application.

La première chose que nous voyons, c'est le Dashboard de Bluemix : Fig.2.

Ce tableau de bord nous donne une vision glo-

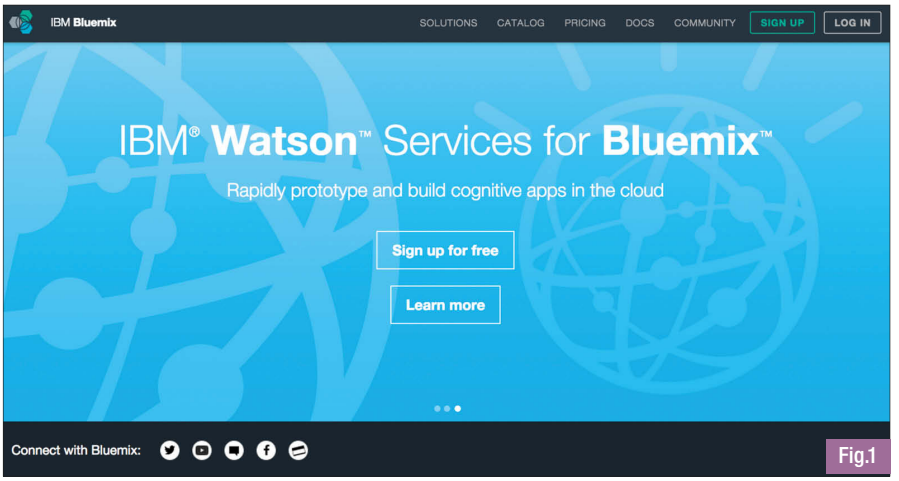


Fig.1

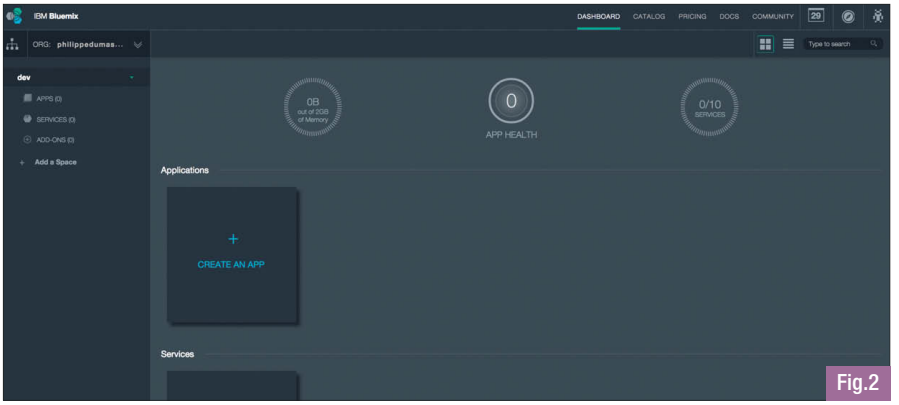


Fig.2

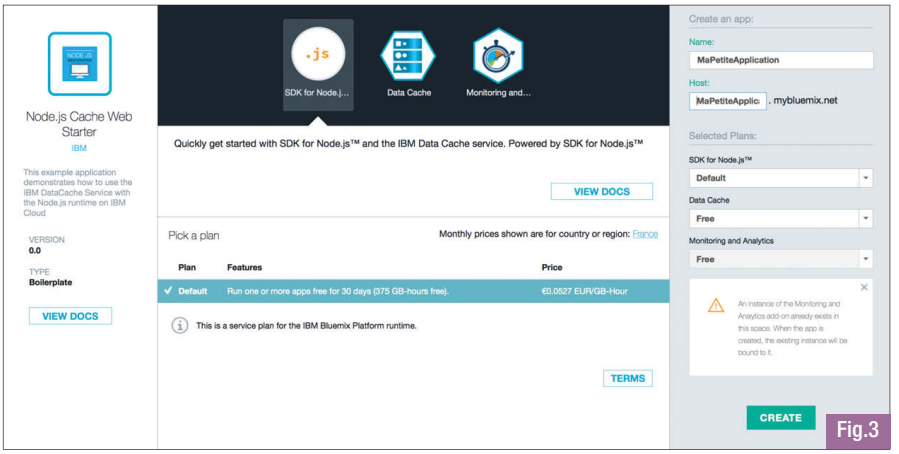


Fig.3

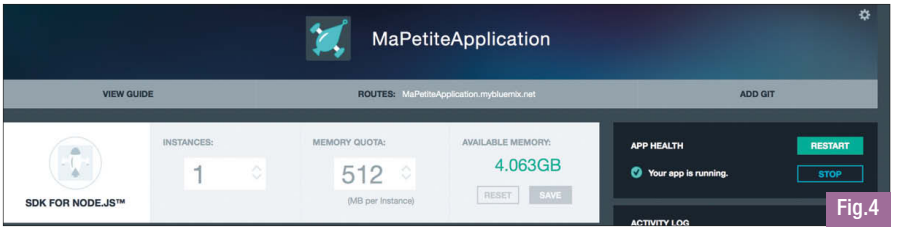


Fig.4

bale de l'espace actif de notre organisation. Par défaut, l'espace s'appelle **dev** et notre organisation porte le nom de notre adresse email. Si vous créez différents espaces dans Bluemix, par exemple **prod**, cela vous permettra de dissocier vos applications en court de développement (dev) de celles en production (prod).
Cliquez sur **CREATE AN APP** afin de choisir parmi plusieurs runtimes (tels que Java ou Ruby), des services (tels que IBM DataCache ou MongoDB), et des boilerplates, qui sont en fait des exemples d'applications préconfigurés. Dans notre cas, nous allons choisir le boilerplate **Node.js Cache Web Starter;** vous pouvez au passage revoir sa description. L'application est assez simple et utilise un serveur Node.js qui intègre du code JavaScript, un service IBM DataCache (stockage et persistance de paires clé/valeur) et un service de monitoring de notre application (ce service de monitoring s'inscrit parfaitement dans le contexte DevOps car les développeurs pourront ainsi contrôler les temps de réponse et l'utilisation des ressources par leur toute nouvelle application). A ce stade, vous êtes juste à un clic de faire tourner cette application :

- Remplir avec précaution le **nom** de l'application et surtout le champ **Host**. Le nom du Host doit être unique dans le réseau Bluemix pour vous permettre d'accéder à votre application (utiliser vos initiales et quelques chiffres en plus du nom de l'application par exemple).
- Cliquer sur **CREATE** Fig.3.

Cela ne prendra que quelques secondes pour créer un conteneur (dans une VM existante de Bluemix) puis de charger le code JavaScript dans ce conteneur et enfin de lier cette application aux services associés. Ensuite Node.js est lancé avec cette application et nous pouvons voir que l'application est démarrée dans le Dashboard lorsqu'elle passe au vert (**Your app is running**) Fig.4. Il est alors possible de cliquer sur l'URL associée à ROUTES et cela vous emmène directement dans une nouvelle fenêtre du navigateur sur cette nouvelle application Fig.5. A ce point, nous avons construit et déployé une application simple parfaitement utilisable par quiconque depuis n'importe quel navigateur via Internet.

INTÉGRATION DANS GIT ET DÉVELOPPEMENT AVEC IBM DEVOPS SERVICES

Comment charger du code dans cette plateforme et surtout comment mettre à jour ce code. Pour travailler avec Bluemix, vous avez plusieurs possibilités pour éditer et développer votre application : il est possible d'utiliser votre éditeur favori, un simple Notepad, Eclipse, Visual Studio ou encore un IDE Web. **DevOps Services** est un parfait exemple d'environnement de développement collaboratif comprenant un IDE Web qui permet de développer une application depuis un repository Git et permet ainsi de la déployer dans Bluemix.
Nous allons donc utiliser notre exemple précédent et l'intégrer dans Git. Cliquez sur **ADD GIT** pour pousser le code de notre application dans GIT. Un simple clic crée un repository Git chargé avec notre code applicatif Fig.6.
Vous devrez sans doute saisir votre mot de passe pour vous assurer que DevOps Services peut agir à votre place et cliquer **Sign In** Fig.7.

> **Note :** La première fois que vous utilisez IBM DevOps Services, il se peut que vous soyez obligé de créer un identifiant (JAZZ ID) et que vous l'associiez à votre identifiant IBM ID (le même que celui utilisé pour Bluemix) Fig.8.

Assurez-vous que la case **Populate the Git repository** est bien cochée et cliquez sur **CONTINUE** Fig.9. En quelques clics, vous avez donc créé un repository Git chargé avec le code source de notre exemple. Pour voir le code, cliquer sur **EDIT CODE** à droite de l'URL GIT Fig.10. DevOps Services s'ouvrira dans une nouvelle fenêtre de votre navigateur. Cette page contient des informations que nous reverrons par la suite. Pour valider l'état de votre application, cliquer sur **BUILD & DEPLOY** en

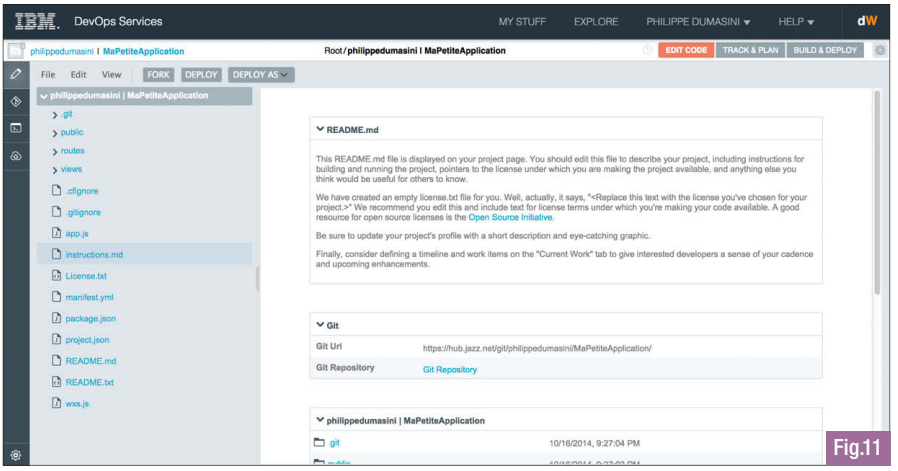


Fig.11

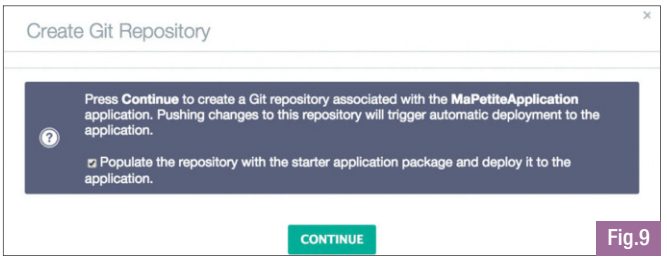


Fig.9

Fig.6

Fig.10

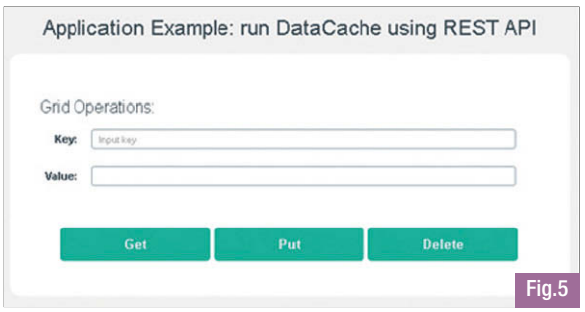


Fig.5

Fig.7

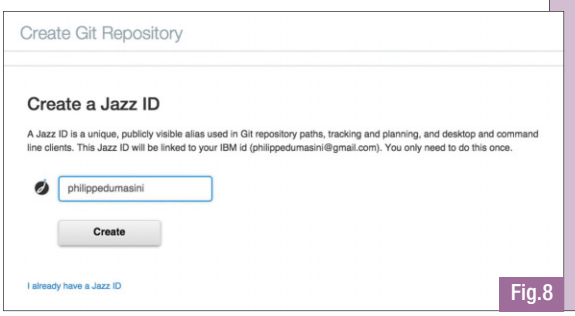


Fig.8

haut à droite. Fig.11. Une page s'ouvre et vous pouvez ainsi voir et configurer votre projet. Fig.12. Notez le message d'avertissement en haut de la page qui indique que des changements sont survenus (il s'agit de ceux qui proviennent de la création et du chargement du repository Git). Pour changer les paramètres de déploiement, cliquer sur **CONFIGURE**. Pour appliquer un changement, on utilisera **REQUEST DEPLOY**.

DÉPLOYER DES MISES À JOURS EN UTILISANT DEVOPS SERVICES

DevOps Services fournit un grand nombre de fonctionnalités très pratiques pour le développeur dont certaines sont mentionnées ci-dessous (ou sur https://hub.jazz.net/learn):

- La prise en charge de projets publics,
- La gestion de plusieurs repositories de code source,
- Support de développement agile,
- Des possibilités de personnalisation pour Bluemix,
- Un mode « Continuous delivery » pour adaptés à Bluemix,
- Un mode « auto-deployment » qui déploie automatiquement les mises à jour.

Cliquer sur le bouton **EDIT CODE** pour accéder à l'éditeur Web (basé sur le projet Open-source **Eclipse Orion**). Votre projet est prêt à être édité (vous pouvez cliquer sur le programme principal **app.js**) : Fig.13.

Depuis cet environnement, vous pouvez éditer vos fichiers sources avec la vérification de syntaxe (pour un grand nombre de langages comme HTML, CSS, JavaScript, Ruby, Python et bien d'autres). Pour certains langages comme JavaScript, l'éditeur propose également la complétion du code. Cela est vrai pour le langage lui-même et pour les extensions fournies par Bluemix (utiliser **Ctrl + espace** pour visualiser l'assistant) : Fig.14.

Nous allons faire une modification dans le code HTML de l'application (dans le titre de l'application). Cliquer sur le crayon en haut à gauche pour venir sur la page de l'éditeur puis utiliser la navigation à gauche pour trouver le fichier : **`views/index.ejs`**. Faire un changement dans le titre de l'application (ligne 8) Fig.15. Pousser ensuite le changement dans Git et suivre les étapes suivantes : Cocher le champ **Select All** afin de pousser les fichiers modifiés puis entrer un message dans la zone de saisie (Enter the commit message) et enfin, cliquer sur **COMMIT** en haut à droite : Fig.16. Puis pousser les changements en cliquant sur **PUSH (Force Push All)** Fig.17. Pour voir les changements dans l'application, cliquer sur **BUILD & DEPLOY** et attendre que l'application soit redéployée. Pour accéder à l'application et constater le changement, cliquer sur l'URL à cté de **Deploy to** Fig.18.

Conclusion

Dans cet exercice, nous avons effectué un simple déploiement avec Bluemix et DevOps Services. Pour plus de contrôle sur le déploiement, vous pourrez utiliser les options de déploiement avancées : il existe un dispositif d'auto-déploiement qui permet de déployer automatiquement le code de votre repository Git (même en cas de développement en équipe). Bluemix est basé sur **Cloud Foundry**. Il permet de déployer nativement les applications depuis des éditeurs comme DevOps Service, Eclipse (à l'aide du plugin Bluemix) ou en utilisant la commande en ligne (CLI) de Cloud Foundry. Bluemix propose également un add-on permettant un développement sans programmation appelé **RapidApps** qui permet de faire du développement rapidement en utilisant du glisser-déposer avec de nombreux outils.

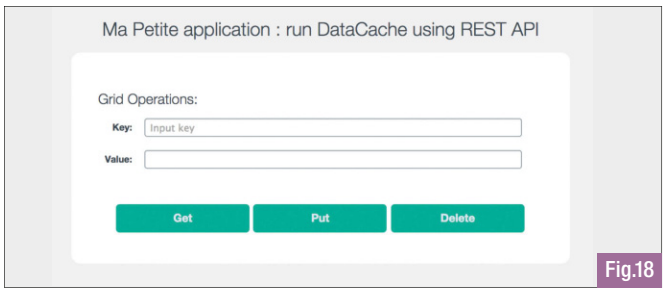


Fig.18

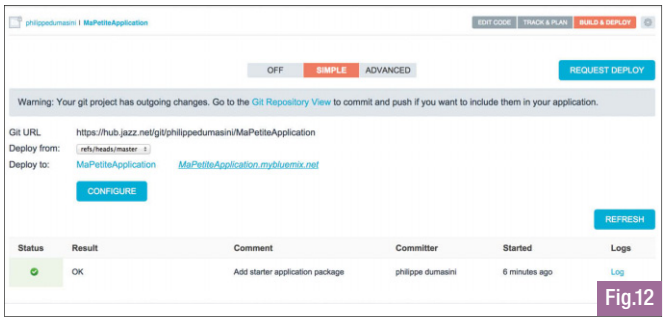


Fig.12

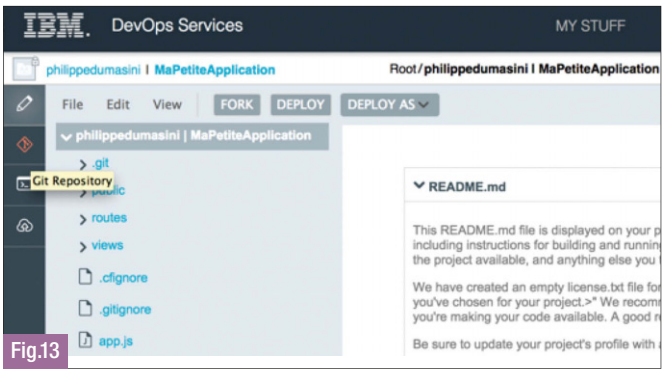


Fig.13

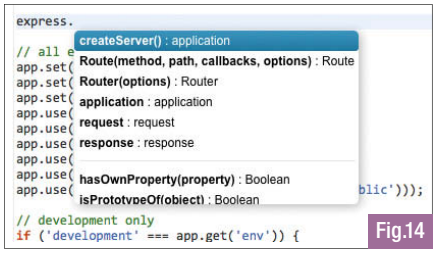


Fig.14

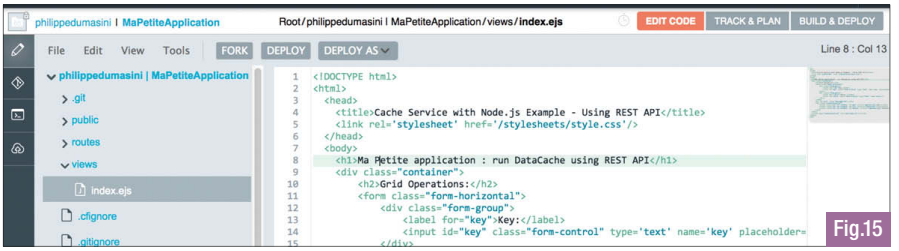


Fig.15

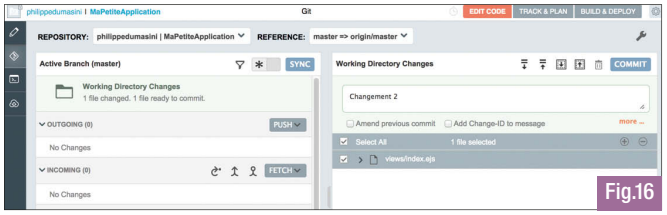


Fig.16

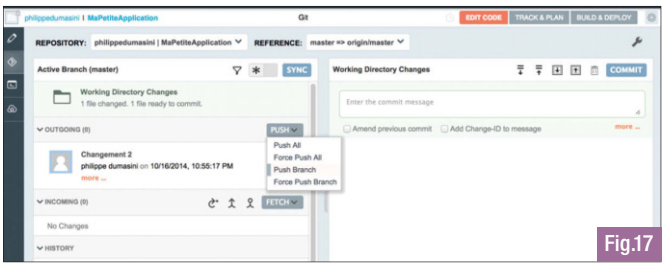


Fig.17

DevOps : développement & production

A travers ce dossier, Xebia vous guide afin que « DevOps » ne soit pas un buzzword de plus pour vous.

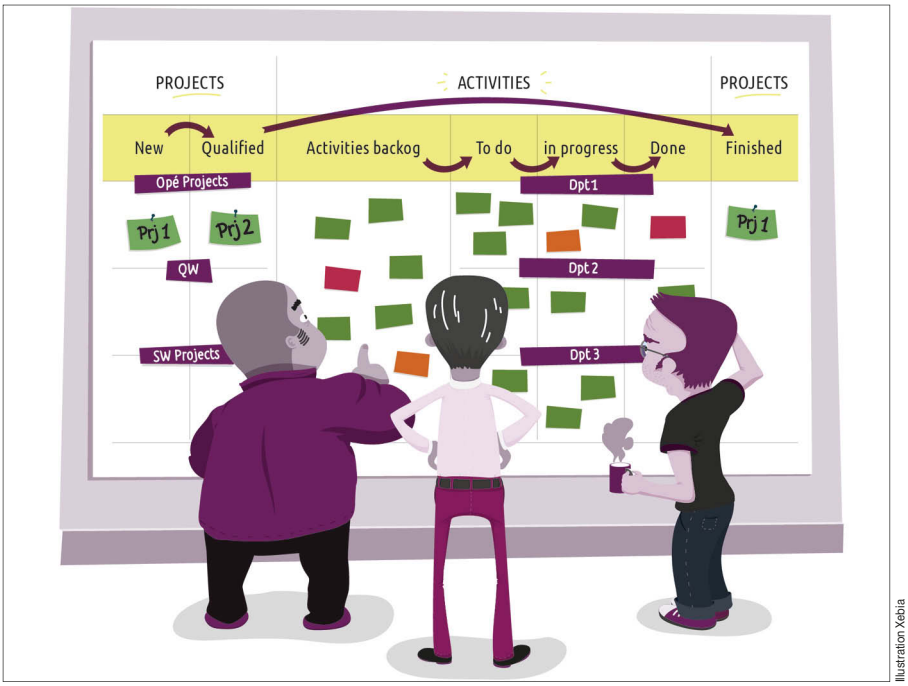


Illustration Xebia

Depuis quelques années, l'agilité connaît un essor grandissant dans les DSI. Aujourd'hui, l'éventail de méthodologies est large et adaptable à tous les contextes. Les valeurs clés de l'agilité sont la collaboration, la transparence, la culture de la qualité, l'adaptation et la simplicité. De l'équipe Scrum à la Feature Team, les transformations agiles ont convaincu de nombreuses directions de SI par leur efficacité au sein des équipes de développement. L'agilité, quand elle n'est appliquée qu'au développement, se trouve néanmoins freinée par les tâches d'exploitation qui surviennent après chaque livraison.
Le but du mouvement DevOps est d'abattre cette frontière en créant une synergie entre les équipes d'exploitation (Ops) et les équipes de développement (Devs). Traditionnellement, Ops et Devs ont des objectifs antagonistes : les uns sont les garants de la stabilité et de la disponibilité des systèmes, là où les autres sont employés à l'évolution de ces derniers. Cela crée un clivage entre ces équipes, appelées à travailler ensemble et à tendre vers un même objectif : délivrer le meilleur logiciel aux clients de l'entreprise. De plus, il est courant que les Ops aient des notions de développement, et les Devs, d'exploitation. Cela entraîne immanquablement des conflits. Qu'elles soient bloquantes ou non, ces frictions dégradent la productivité. Plusieurs symptômes sont révélateurs de ces problèmes :

- En cas de crise, combien de temps faut-il pour lever une alerte, récupérer les logs, les analyser puis identifier la défaillance ? Combien de temps pour livrer un correctif en production ? La rapidité d'exécution de ces actions est fortement liée à la qualité de la coopération entre équipes.
- La fréquence et la simplicité des mises en production sont également des indices révélateurs. Les Ops sont rarement impliqués au démarrage des projets. Il s'ensuit des délais allongés entre la livraison des applications et celle des machines qui serviront de socle. Cela aboutit souvent à une détection de problèmes en pré-production, seul environnement suffisamment proche de la production pour une validation. Les open spaces grouillent d'anecdotes du même acabit. Nous allons vous guider dans notre vision d'une démarche d'amélioration, pour pallier ces problèmes efficacement.

COOPÉRER

La culture et l'organisation de votre entreprise sont un pilier de votre transformation vers DevOps. Les leitmotifs de DevOps peuvent néanmoins paraître galvaudés : qui ne se targue pas de vouloir travailler de manière collaborative et en toute transparence ? De ce fait, quelles sont les implications concrètes sur l'organisation d'un département exploitation derrière le mouvement DevOps ? Est-ce une réelle rupture ou un simple effet de mode ? Les organisations de type « You build it, you run it » sont sans compromis et semblent un objectif utopique à atteindre pour beaucoup. Voyons ici quelques axes de travail et les outils que vous pouvez utiliser pour démarrer rapidement une démarche DevOps sans avoir à révolutionner votre organisation.

Restaurer la confiance

L'activité des Ops est jalonnée de nombreuses crises. Qu'elles soient liées à des pannes matérielles ou à une mise à jour provoquant une instabilité du système. Les Ops sont familiers de la gestion de crise, avec ses horaires à rallonge et ses diagnostics parfois laborieux. Les bureaux d'Ops résonnent d'histoires croustillantes sur ces Devs totalement inconscients qui mettent la production en péril avec des déploiements de binaires hasardeux. L'un des premiers axes de progrès est donc de restaurer la confiance dans un contexte de defiance mutuelle. Pour ce faire, deux outils sont à votre disposition. D'une part, la conclusion d'une mise en production doit être l'objet d'une réunion de retour d'expérience (appelée rétrospective) entre Devs et Ops afin de capitaliser sur les bonnes pratiques et identifier les points d'amélioration. Une pratique régulière des rétrospectives devrait diminuer sensiblement le nombre de crises et restaurer la confiance entre ces deux protagonistes.


Illustration Xebia

Concevoir des produits « Ops-ready »

Les projets agiles manquent souvent d'une vision Ops très tôt dans la conception du produit. Ce manque est induit par le fait que le Product Owner, propriétaire du backlog, a une vision métier centrée sur les fonctionnalités à délivrer à l'utilisateur. La dimension Ops sera, au mieux sous-estimée, au pire passée sous silence. Les Ops sont des parties prenantes dont le PO doit tenir compte. Il est essentiel de les impliquer dès la constitution du Product

Backlog. Le Product Backlog représente tout ce qui apporte de la valeur au produit. Des exigences non fonctionnelles comme certificat apportent de la valeur, elles doivent apparaître dans le backlog. D'autres besoins Ops se traduiront par des critères d'acceptation des user stories. Par exemple, PCA ou sécurité.

Anticiper l'activité ops

Une autre difficulté pour les Ops est d'anticiper les demandes des Devs et les livraisons de nouveaux binaires. Il n'est pas rare de voir des Devs se plaindre du manque de réactivité des Ops, et des Ops se plaindre du manque d'organisation des Devs qui ne savent pas anticiper leurs besoins. Quelles qu'en soient les raisons, les échanges sont souvent tendus et les urgences sont la norme, plus que l'exception. Il suffit parfois de partager un simple outil de management visuel pour créer de la transparence sur les travaux en cours cote Devs et permettre aux Ops d'anticiper les demandes. Il est également possible d'inviter des Ops dans des revues de sprint afin qu'ils s'approprient le produit avant de voir arriver le livrable pour mise en production. Cette dernière pratique est à utiliser avec parcimonie car les Ops sont, en général, peu intéressés par le contenu fonctionnel d'un sprint.

FLUIDIFIER

Maintenant qu'une véritable coopération entre équipes s'est installée, les cérémonies agiles mêlant Devs et Ops sont prolifiques. Chacun écoute les problèmes de l'autre. Cela donne l'occasion d'harmoniser l'ensemble :

- Développer en accord avec la cible qu'est l'environnement de production.
- Adapter, dans la mesure du possible, le SI aux besoins des développeurs.

De plus en plus de structures ont réussi cette transition. Du rapprochement entre les équipes sont nés de nombreux outils d'automatisation. Le but de ces outils, que nous allons voir plus en détail, est de transformer le SI en un self-service pour développeurs, géré par les Ops.

Usine logicielle

Première étape de tout projet : la mise en place des outils de développements. Il faudra à une équipe :

Un dépôt de code versionné

Git ou Mercurial rempliront le job sans difficulté. Ils permettent tous deux de faire des commits locaux aux machines des développeurs avant d'envoyer leur travail par paquet au dépôt central. Cela leur donnera la possibilité de faire des commits plus petits et donc, de réduire grandement les problèmes de

Illustration Xebia

merge en intégrant leur code au tronc commun.

Il doit être possible pour n'importe quel développeur de créer de nouveaux dépôts simplement. Pour cela, des outils comme GitLab, GitBlit ou RhodeCode vous offriront une interface Web de gestion des utilisateurs et des dépôts de code.

Un serveur d'intégration continue

Jenkins, de par sa simplicité d'usage et ses nombreux plugins, est tres populaire. Vous pourriez également regarder TeamCity ou Bamboo pour déterminer quel outil vous convient le mieux. Tous ces outils sont également disponibles en Service Cloud pour vous éviter de les mettre en place en interne et gagner en vitesse de mise en place.

Un dépôt d'artefact

En fin de build, les artefacts produits doivent être stockes pour servir aux processus de déploiement. Là, le choix est extensif en fonction de votre approche :

Si votre intégration continue délivre des fichiers WAR ou un autre artefact type Java, un dépôt Maven tel que Archiva ou Nexus est à conseiller.

Si vous souhaitez pousser l'intégration continue à produire des paquets systèmes Linux (des .deb ou des .rpm par exemple), il faudra alors adopter le mode de distribution ad-hoc.

Dans tous les cas, l'essentiel est d'avoir au moins l'équivalent d'un serveur FTP, ou vous stockerez les artefacts en les rangeant correctement par numéros de version.

Infrastructure

La mise en place de l'infrastructure doit passer par des outils de provisionnement automatique, comme VMWare Cloud Template ou AWS CloudFormation. Cela implique bien sur d'avoir un socle de virtualisation qui permette de scripter le démarrage des ressources.

- La virtualisation des différents environnements, du développement à la production, garantit :
- Une bonne réactivité dans le provisionning,
- La reproductibilité totale de l'installation des plateformes,
- La possibilité d'obtenir à moindre cout de l'élasticité sur la plateforme de production,
- La possibilité de recréer à la demande une plateforme depuis zéro (pour les disaster recovery ou bien les tests de charge par exemple).

En utilisant cette approche depuis la phase de développement, et en considérant que les Ops participent à l'équipe, on obtiendra une infrastructure versionnée, déployable à la demande dans les environnements choisis (développement, recette ou production) en restant isoproduction. La mise en place de répartiteurs de charge, de groupes de scaling

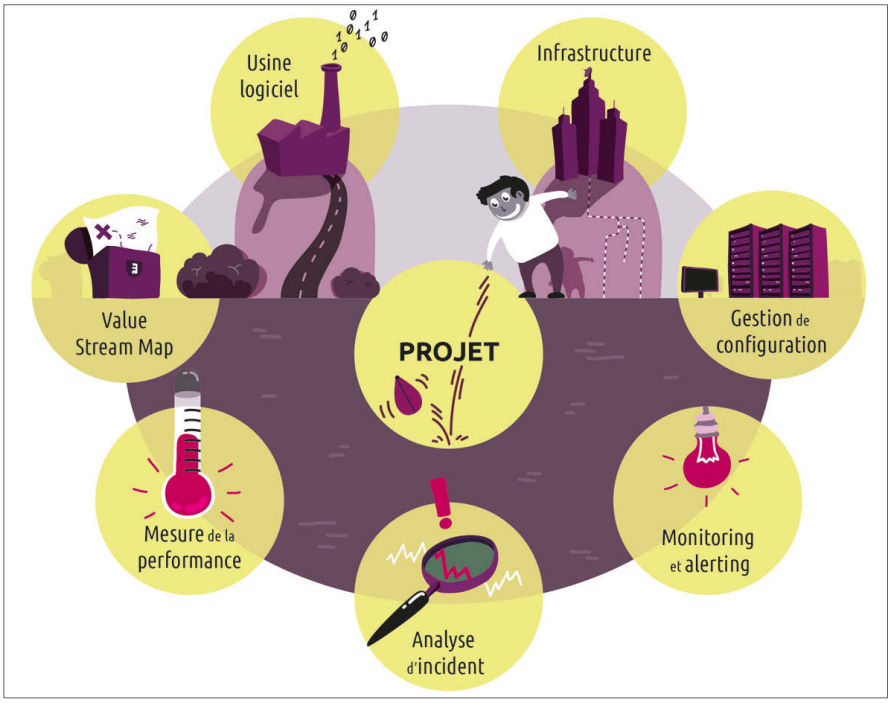


Illustration Xebia

automatiques et de groupes de sécurité doit également faire partie de ce provisionning et être rédigée par toute l'équipe, pour repartir la connaissance.

Gestion de configuration

Avant même de songer à centraliser les fichiers de configuration des serveurs, il faut que l'application développée soit apte à recevoir de la configuration externe. Toutes les valeurs nécessaires doivent pouvoir être surchargées. Une discussion entre Devs et Ops est essentielle pour déterminer si l'application est suffisamment configurable. À la charge des Devs de rédiger la configuration correspondant à leur environnement de validation.

Une fois cette analyse faite, il faudra intégrer un système de centralisation de configuration, tel Ansible, Chef, Puppet ou CFEngine. Ces outils sont bases sur un système de serveur central pilotant la configuration des serveurs clients. Ils peuvent également servir à piloter de façon uniforme un ensemble de machines pour effectuer par exemple, une mise à jour. Tous ces outils sont de tres bonne qualité et utilises par des acteurs majeurs du Web; à vous de consulter la littérature pour voir lequel vous appelle.

Le monitoring et l'alerting

L'expérience de mise en place des méthodes agiles au sein des équipes Devs a montré que l'augmentation de la visibilité sur l'avancement du travail induit des progrès à différents niveaux :

- Afficher les résultats des tests automatises et les indicateurs de qualité de code sur les écrans d'un open space responsabilise les développeurs.
- Certaines équipes opérationnelles barrent, sur un gigantesque calendrier, les jours sans incident notable afin d'apporter un feedback positif à l'équipe sur son travail.
- Certaines startups affichent, sur de grands écrans, des l'entrée de leurs locaux, les chiffres de vente du jour. C'est un moyen de focaliser l'attention de l'ensemble de l'entreprise sur le but final, qui est le fonctionnement du groupe et non d'une équipe isolée.

Pour être efficace, ces outils de supervision doivent pouvoir accueillir un nombre illimité de mesures : Devs et Ops doivent pouvoir exposer autant de métriques qu'ils le souhaitent, sans mettre en péril la production et sans être freinés par des considérations de coût ou de matériel. Il faut donc faciliter la mise en place de nouvelles "sondes" au sein du système. Parmi diverses solutions, nous avons sélectionné un outil open source, baptise Graphite, qui :

- Est facilement clusterisable, donc pourvu d'un stockage virtuellement extensible à l'infini,
- Peut conserver un très grand nombre de points sous forme de séries temporelles, et les agréger lorsqu'une granularité fine n'est plus nécessaire,
- Permet d'appliquer sur ces points des formules mathématiques avancées.

Ce logiciel est un simple collecteur, il est non-intrusif vis à vis des applications de production. De nombreux plugins Graphite sont disponibles : ces plugins vont du traçage des valeurs système à l'interception de trappe Nagios. Couplé avec un autre outil open source, JmxTrans, il devient tres facile de grapher des variables exposées par JMX, le système préférentiel pour exposer des mesures en Java.

On ne peut évidemment pas rester toute la journée les yeux rives sur les compteurs, c'est pourquoi il est bon d'avoir un système d'alerte efficace. Classiquement, on déclenche des alertes sur des passages de seuils, comme "le CPU de telle machine dépasse 90 %". Avec un couple Graphite-JmxTrans, il est maintenant possible de mettre en place des alertes sur des tendances, comme "le CPU de telle machine dépasse en moyenne les 90 % depuis plus de 10 minutes". Comme la collecte des métriques est centralisée, il est même possible de croiser plusieurs sources d'information pour déclencher des alertes tres ciblées, comme « sur les 10 dernières minutes, le CPU dépasse en moyenne les 90%, le pool d'accès aux services externes est sature et le temps de parcours client au moment du paiement est supérieur de 50% au temps moyen constate en fonctionnement normal ». Si cette dernière alerte se déclenche, alors il est temps d'avertir mon partenaire paiement que son service est dégradé. Cet alerting peut se faire via des outils classiques comme Nagios ou, bien sur, un logiciel open source dédié tel Seyren.

Ce monitoring technique et fonctionnel a un autre effet bénéfique : il fait entrer l'entreprise dans la culture de la mesure, qui se résume à cette phrase : « Vous ne pouvez améliorer que ce que vous pouvez mesurer ».

Analyse d'incident

Le monitoring est également utile au-delà de la visualisation de données métier. Les logs générés par les briques de votre SI doivent être traités avec une attention particulière. Ils sont le matériau de base de toute analyse d'incident. Le problème technique qui se pose rapidement (et qui enlise de nombreux SI) est de savoir exploiter les logs générés par un grand nombre de machines, à travers tout le SI. Sans stratégie de collecte et d'archivage centralisée, vos équipes vont devoir commencer par déterminer sur quelle machine un problème a eu lieu, avant même de pouvoir commencer à l'analyser.

Pour parer à cela, il convient de mettre en place sur chaque machine un agent de collecte qui enverra les logs émis par une machine sur un serveur central pour tri, archivage et indexation. De cette façon, n'importe quel membre d'équipe saura immédiatement venir consulter les logs et les métriques de monitoring, collectes depuis toutes les instances de serveur.

Une solution technique puissante, qui évitera d'avoir à faire des recherches exactes dans des fichiers de log de plusieurs Go, est le triptyque LogStash, ElasticSearch, Kibana :

- LogStash vous fournira les agents locaux et la collecte centralisée,
- ElasticSearch est un moteur d'indexation et de recherche full-text qui servira de stockage,
- Kibana est une interface Web pour requêter sur l'ensemble des logs collectes et construire des tableaux de bords clairs.

La mesure de la performance

Il est assez courant de rencontrer, dans un SI, des applications éclatées, réparties sur différents serveurs, ou des actions utilisateurs mettant en jeu plusieurs applications. Dans ce genre de cas, le monitoring de chaque machine, même avec une consultation centralisée, ne suffit plus à travailler efficacement. L'analyse du comportement du système complexe que devient alors le SI nécessite un outillage plus puissant.

Deux approches complémentaires permettent de lutter contre l'effet "Works for me" qui peut apparaitre :

- Pratiquer des tests de charge continus,
- Mesurer les performances réelles en production.

Value stream map

Tous les outils et méthodes que nous venons de passer en revue vont donner beaucoup plus de pouvoir à vos équipes sur votre SI. Ils seront en mesure d'observer, de comprendre, de diagnostiquer et de réparer au plus vite. On peut néanmoins aller plus loin encore. Un autre outil permet d'analyser le fonctionnement de vos équipes plutôt que celui de votre SI : la Value Stream Map. Cet outil issu du Lean Management permet de calculer l'efficience de votre processus et d'identifier les points d'amélioration avec une vue d'ensemble. Dans

un contexte DevOps, il s'agira d'identifier précisément toutes les étapes nécessaires au déploiement d'un binaire en production depuis sa livraison par les Devs. Nous associons à chaque étape sa durée de travail effectif et le temps de latence constate avec l'étape précédente. Le ratio entre le cumul des temps de travail effectifs et le cumul des temps d'attente vous donne votre efficience globale. Il s'agira ensuite de cibler votre effort d'amélioration pour augmenter ce ratio.

LIVRER

Maintenant que Devs et Ops sont équipés d'outils efficaces et ont confiance en leur mise en production, ils n'attendent plus qu'une chose : des binaires. Jusqu'ici, nous avons lié Devs et Ops autour de la connaissance du système au sens large.
Si les équipes arrivent à ce stade de maturité, elles vont commencer à réellement partager un processus bout en bout. Une même équipe aura les connaissances et les outils pour développer une fonctionnalité et la mettre en production. C'est ici la terre promise du Déploiement Continu. Voyons ensemble quelques bonnes pratiques et stratégies de déploiement à considérer pour exploiter votre nouveau SI et vos équipes dopées à la DevOps-amine.

Mise en place du déploiement continu

Il est important que vos équipes définissent leur stratégie de gestion de branche dans le code source. Il faut qu'elles soient conscientes qu'il y aura des patchs à faire sur la version de production et qu'elles doivent pouvoir mettre temporairement de cote leur travail en cours pour résoudre un problème. Deux stratégies ont aujourd'hui fait leurs preuves.

Simple Branching

Le Simple Branching est, comme son nom l'indique, la stratégie la plus simple. On garde une branche dont le code est la version déployée en production. On crée également une branche de développement dans laquelle une nouvelle version de l'application est en cours de réalisation.

Feature Branching

Le Feature branching consiste, quant à lui, à avoir une branche principale contenant la version actuellement en production et plusieurs branches de développement. Pour chaque fonctionnalité en cours de développement, une nouvelle branche est créée. Lorsque son implémentation est terminée et validée par le métier, les modifications sont rapatriées sur la branche principale pour livraison en production.

Plasticité applicative

Toggle Feature Pattern

Le Toggle Feature pattern est un outil de conception logicielle. Les Devs peuvent positionner des interrupteurs dans le code de l'application pour encapsuler des fonctionnalités entières. Ces derniers doivent être activables par configuration et manipulables à chaud par les Ops pour modifier le comportement de l'application.
En protégeant les fonctionnalités en cours de rédaction par le biais de ce mécanisme, il est possible de maintenir le rythme et de déployer des fonctionnalités, même si d'autres sont encore en chantier.

Illustration Xebia

Circuit Breaker Pattern

Outre la ségrégation des fonctionnalités non terminées, toute fonctionnalité reposant sur des services externes doit pouvoir être activée ou désactivée à chaud, sans stopper l'ensemble de l'application.
Le monitoring peut avertir les Ops qu'un service est indisponible, mais il est également possible d'intégrer, directement dans les applications, des mécanismes de protection contre les pannes externes.
Le principe du pattern Circuit Breaker est de conditionner les accès vers des composants externes par des interrupteurs logiques qui détectent les appels en erreur. Quand un seuil (configurable) d'erreurs consécutives est atteint, l'application passe en mode dégradé et n'assure plus la fonctionnalité qui dépend du service en panne. Pendant que l'application est en mode dégradé, elle envoie une alerte au monitoring et déclenche une procédure pour vérifier régulièrement si le service est revenu. Dès que cette procédure détecte un retour du service, l'application repasse en mode standard et assure de nouveau la fonctionnalité.

A/B Testing

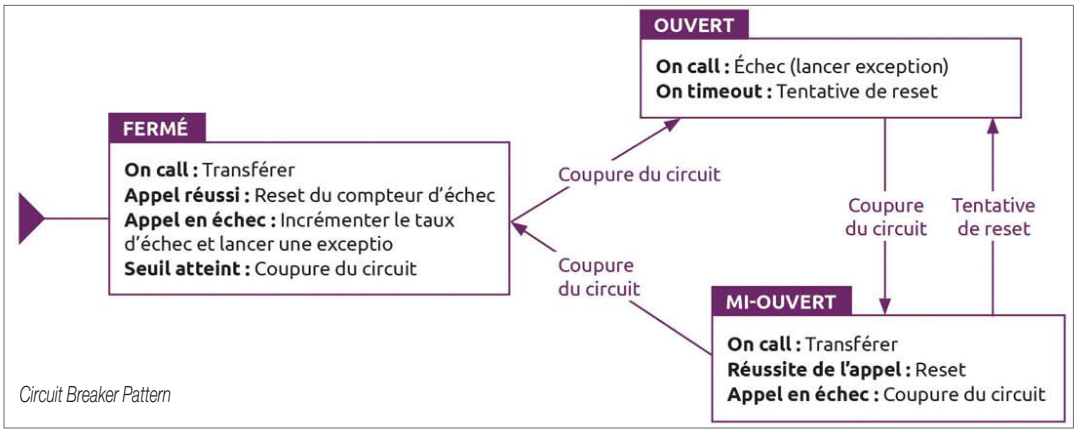
Une fois la technique du Toggle Feature Pattern maîtrisée par vos équipes, il est possible de l'utiliser, non plus seulement pour masquer une fonctionnalité instable, mais aussi pour proposer aux utilisateurs différentes versions d'une même fonctionnalité.
En effet, de plus en plus d'entreprises procèdent à des tests grandeur nature, en production, par échantillonnage. Par exemple, via la mise en place d'un A/B testing : 2 pages d'accueil, au design différent, sont proposées de manière aléatoire à 2 populations d'internautes. Le monitoring permet de dégager des tendances indiquant la page la plus pertinente.
Le principe du A/B Testing n'est pas limité à 2 options et il est toujours possible de jouer sur différentes nuances d'ergonomie, de simplicité de parcours, etc.
La pertinence du A/B Testing doit se mesurer en valeur métier. Typiquement, pour un site marchand, on attribuera le taux de transformation en vente d'une version de l'application. La meilleure version devient la norme et on tente d'autres alternatives sur cette nouvelle base pour améliorer encore l'expérience utilisateur.

Dark Launch

Autre pratique corollaire du Toggle Feature et de l'A/B Testing : le Dark Launch, qui permet de tester des fonctionnalités alternatives considérées comme risquées. On va alors activer une fonctionnalité à une frange des utilisateurs, en se basant sur des certains critères (peu d'utilisateurs, leur type de compte, leur nationalité, leurs préférences, etc.). Une fonctionnalité typiquement considérée comme à risque est la modification du parcours de paiement pour un site marchand. Comme c'est à cette étape que se concentre la majorité des abandons d'achat, il est tres délicat de décider de la modifier. L'utilisation d'un Dark Launch pour confirmer la pertinence d'une modification avant de la généraliser est ici appropriée.

Stratégies de déploiement

Votre projet s'appuie maintenant sur un processus de déploiement bien rodé, au sein duquel l'ensemble des étapes de build, de tests et de provisioning sont automatisées. Vous êtes capable de livrer votre application en continu mais le déploiement en production nécessite toujours un Go/NoGo. L'étape suivante consiste à automatiser le déploiement en production en tant qu'action finale de la chaine de production.
Comme il n'est pas acceptable qu'une



Circuit Breaker Pattern

modification du code aboutisse à des coupures de service à chaque mise en production automatique, il faut mettre en place une stratégie de déploiement sans interruption de service. Différentes techniques peuvent être considérées afin de réussir à diminuer les risques et dépasser les craintes liées à un tel bouleversement dans la manière de livrer une application en production. Le but est de transformer le déploiement en un non-évènement, quelque chose de standard.

Blue/Green Deployment

Cette technique de déploiement sans interruption de service repose sur un système de double run. On maintient 2 environnements de production pleinement fonctionnels. Tous les accès utilisateurs passent par un routeur qui permet la bascule vers l'un ou l'autre de ces environnements. La version vers laquelle pointe le routeur est l'environnement Green, tandis que la partie en stand-by est appelée le Blue. La technique consiste à déployer la nouvelle version sur le Blue, faire toutes les vérifications utiles pour valider le déploiement, puis basculer le trafic du routeur.
Une fois le trafic basculé, l'environnement "nouvelle version" devient le Green, et on peut sereinement mettre à jour le second déploiement pour maintenir l'équilibre entre Blue et Green.

Canary releasing

Dès que votre infrastructure de production commence à prendre de l'ampleur, il peut devenir problématique de rester dans une approche Blue/Green Deployment.
Redonder une grappe de serveurs frontaux, un nuage de middleware et plusieurs bases de données peut rapidement couter tres cher.
Vous pourriez alors être tente par une stratégie nommée Canary Releasing.
Cela consiste à déployer la nouvelle version de son application sur un sous-ensemble des noeuds qui composent votre environnement de production. Cela peut se faire soit en augmentant légèrement le nombre de vos instances durant le déploiement, soit en mettant à jour un ensemble restreint d'instances existantes. Dans tous les cas, il faut être capable de maitriser quelle proportion de votre trafic va être dirigée vers ces instances avec la nouvelle version de votre application. Vous devrez porter une attention particulière à ces instances pour valider que la nouvelle version peut être généralisée. En cas de problème, il vous suffira de couper le trafic entrant sur ces instances pour arrêter de perturber vos utilisateurs. Une fois votre stratégie choisie, vous allez avoir besoin d'un outil pour gérer le déploiement à proprement parler de l'applicatif.

CONCLUSION

Grâce à une implication forte des équipes de développements et des équipes d'opérations, le suivi de l'avancement des projets est assuré de manière transverse fournissant ainsi une vision globale et non plus spécifique à chaque équipe. La richesse des échanges vient améliorer, du point de vue de tous, les solutions finales. Les Devs prennent en compte intuitivement les besoins spécifiques des Ops et vice versa. D'autre part, les outils d'automatisation mis en place simplifient les tâches nécessaires au delivery de bout en bout. À terme, le temps perdu entre les développements et la mise en production diminue régulièrement. La value Stream Map permet de constater cette amélioration d'itération en itération.
L'innovation et la veille technologique se développent dans la DSI. D'un point de vue managérial, le planning des releases obtient des niveaux de confiance inégalés jusqu'alors. Un rythme s'est installé, les mises en productions sont régulières et sécurisées, les interventions d'urgence se raréfient.
Attention, il ne s'agit pas d'une recette miracle! Le changement d'organisation s'est accompagné de nombreuses difficultés, la résistance au changement est toujours forte, Nous avons présenté un grand nombre d'outils destinés aux Devs comme aux Ops, qui peuvent aussi trouver leur utilité dans les équipes métiers. Tous ces systèmes ne serviront néanmoins que tres peu sans une coopération forte entre les équipes et une adhésion aux principes du DevOps.
Merci aux auteurs et aux contributeurs du TechTrends #2 DevOps de Xebia ayant permis la rédaction de cette partie.
Retrouvez ce document dans son intégralité sur http://techtrends.xebia.fr/.

Analyse orientée business de vos logs applicatifs

Dans cet article, nous allons voir comment mettre à profit nos logs applicatifs afin de faire de l'analyse orientée "business" grâce à ElasticSearch, Logstash et Kibana.

Vincent Spiewak
Consultant Java et agile chez Xebia
@vspiewak

A la tête d'un site de vente en ligne fictif, nous essayerons notamment de monitorer l'activité de nos clients, et de répondre aux questions suivantes:

- Quel est le taux de conversion (ratio recherches / ventes) ?
- Quels sont les produits les plus consultés ?
- Où nous situons-nous par rapport à nos objectifs de ventes ?

A faire chez vous

L'ensemble de la démonstration a été automatisée sur une VM Vagrant disponible sur GitHub : https://github.com/vspiewak/elk-programmez-2014.git
Vous devez installer uniquement VirtualBox et Vagrant (disponible gratuitement pour Windows, Linux et OS X). Vous trouverez le script d'installation ainsi que l'ensemble des fichiers de configuration.

Format des logs

Ce paragraphe vous présente le format des logs que nous allons exploiter **Fig.1**. Notre application génère deux types de logs, représentant respectivement une recherche ou un achat dans notre magasin de vente en ligne. Un log de vente contient l'adresse IP et le User-Agent de l'acheteur, son sexe, ainsi que les informations sur le produit acheté (marque, modèle, catégorie, options et prix).
Un log de recherche contient l'adresse IP et le User-Agent du visiteur, et quelques informations sur le produit recherché (parfois la catégorie, parfois la catégorie et la marque, etc).

Logstash

Logstash est un ETL léger permettant de collecter, transformer et charger des logs à l'aide de connecteurs d'entrées, de filtres et de connecteurs de sorties. C'est un véritable couteau suisse qui, en version 1.4.2, vous fournit 41 entrées, 50 filtres, et 55 sorties différentes.
Nous allons voir comment transformer nos lignes de logs illisibles en documents JSON structurés, et les enrichir au passage de précieuses informations (géolocalisation, version du navigateur, etc).

Premiers pas

Un agent Logstash a besoin d'un fichier de configuration pour être exécuté. Voici un exemple de configuration minimaliste : **Fig.2.**

Fig.1

```
07-10-2014 09:51:29.994 [pool-1-thread-1] INFO com.github.vspiewak.loggenerator.SearchRequest
 - id=1&ua=Mozilla/5.0 (Windows NT 5.1) AppleWebKit/534.24 (KHTML, like Gecko) Chrome/11.0.69
6.50 Safari/534.24&ip=83.112.70.237&category=Portable

07-10-2014 09:51:30.002 [pool-1-thread-1] INFO com.github.vspiewak.loggenerator.SellRequest -
 id=3&ua=Mozilla/5.0 (X11; U; Linux x86_64; en-US) AppleWebKit/534.16 (KHTML, like Gecko) Chr
ome/10.0.648.205 Safari/534.16&ip=62.23.215.100&email=client4@gmail.com&sex=F&brand=Apple&nam
e=Mac Mini&model=Mac Mini - Core i7&category=Ordinateur&options=Core i7&price=829.0
```

L'entrée standard est utilisée comme "input", la sortie standard est utilisée comme "output" (A noter que nous utilisons le codec "rubydebug" afin que le json soit formaté).
Nous pouvons lancer ensuite l'agent Logstash avec cette configuration, et écrire un message dans l'entrée standard : **Fig.3.**
Logstash a ainsi transformé notre message en un JSON formaté. Logstash ajoute automatiquement les champs @version, @timestamp et host. A noter que le champ @timestamp contient la date de réception du message par Logstash.
Le premier travail va consister à parser nos logs à l'aide du filtre grok...

Filtre Grok

Le filtre Grok nous permet, à l'aide d'expressions régulières, de découper nos logs et leur donner de la sémantique. Logstash vient avec plus d'une centaine d'expressions régulières prédéfinies (disponibles ici : https://github.com/logstash/logstash/tree/master/patterns)
Par exemple, le pattern "COMBINEDAPACHELOG" du fichier grok-patterns permet de parser les lignes de logs d'un serveur Apache.
Vous pouvez utiliser 2 syntaxes :

- %{SYNTAX:SEMANTIC} ou %{SYNTAX:SEMANTIC:TYPE}
- (?<field_name>the pattern here)

La première syntaxe utilise un pattern Logstash prédéfini (ex: INT, NOTSPACE, LOGLEVEL, ...), SEMANTIC étant le nom du champ à mapper. Vous pouvez préciser le type du champ (integer, float, string) via le paramètre TYPE. La deuxième syntaxe vous permet de définir un pattern customisé avec ou sans l'aide de plusieurs patterns Logstash.
Pour nos logs, nous utiliserons le filtre Grok suivant :

```
filter {
 grok {
   match => ["message","(?<log_date>%{MONTHDAY}-%{MONTHNUM}
-%{YEAR} %{HOUR}:%{MINUTE}:%{SECOND}.[0-9]{3}) \[%{NOTSPACE:
thread}\] %{LOGLEVEL:log_level} %{NOTSPACE:classname} - %{GREEDY
DATA:log_msg}"]
   }
}
```

Avec le filtre Grok, logstash découpe nos lignes de log correctement, ajoutant les champs log_date, thread, log_level, classname et log_msg : **Fig.4.**

Filtre date

Le filtre date va permettre de dater notre message correctement. En effet, par défaut logstash valorise le champ @timestamp avec la date courante. Dans le cas de nos logs, nous voulons utiliser la date du log (champ log_date). Nous utiliserons un pattern au format JodaTime (voir :

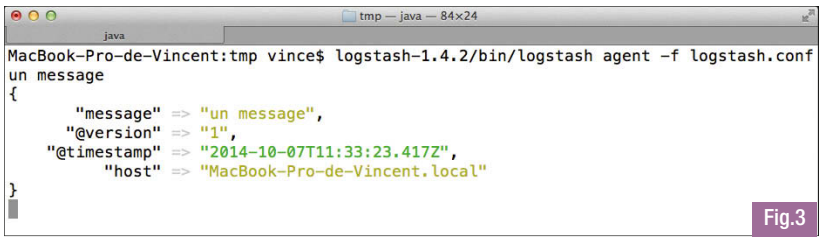


Fig.3

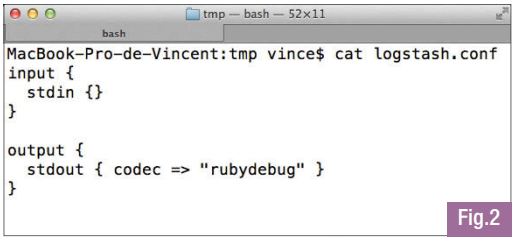


Fig.2

http://joda-time.sourceforge.net/apidocs/org/joda/time/format/DateTimeFormat.html)

```
date {
    match => ["log_date","dd-MM-YYYY HH:mm:ss.SSS"]
}
```

Filtre KV

Le filtre KV est très utile pour découper un champ au format cle1=valeur2&cle2=valeur2 (paramètres d'une requête http notamment) Nous allons utiliser ce filtre pour découper le champ "log_msg":

```
kv {
    field_split => "&"
    source => "log_msg"
}
```

Filtre mutate

Ajout d'un tag

Le filtre mutate est un filtre "couteau suisse" permettant différentes modifications sur les champs. Nous allons ajouter un tag afin de différencier nos recherches de nos ventes :

```
if [classname] =~ /SellRequest$/ {
    mutate { add_tag => "sell" }
} else if [classname] =~ /SearchRequest$/ {
    mutate { add_tag => "search" }
}
```

Conversion de type

Nous allons convertir les champs id et price respectivement en entier et nombre à virgule flottant :

```
mutate {
    convert => [ "id", "integer" ]
}
mutate {
    convert => [ "price", "float" ]
}
```

Suppression d'un champ

Toujours avec le filtre mutate, nous allons supprimer le champ log_msg que nous avons re-découpé à l'aide du filtre KV :

```
mutate {
     remove_field => [ "log_msg" ]
}
```

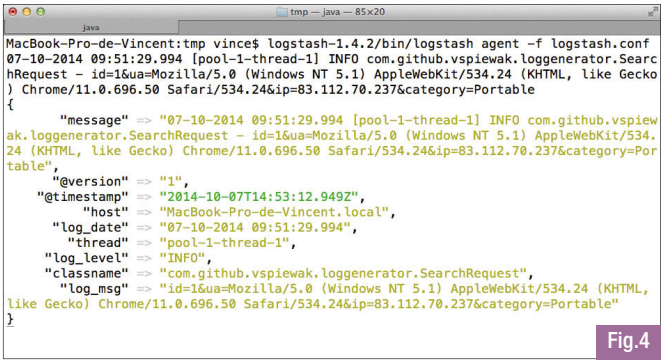


Fig.4

Split d'un champ en tableau

Pour finir avec le filtre mutate, nous allons splitter notre champ "options" afin d'avoir un tableau d'options :

```
mutate {
    split => [ "options", "|" ]
}
```

Filtre GeoIP

Le filtre GeoIP permet d'ajouter des informations de géolocalisation via une adresse IP (coordonnées gps, ville et pays notamment). Logstash utilise la base de données GeoLite de Maxmind sous license CCA-ShareAlike 3.0.

```
filter {
    geoip {
        source => "ip"
    }
}
```

Filtre UserAgent

Le filtre UserAgent permet de parser automatiquement un User Agent et d'ajouter des informations comme la version du système d'exploitation ou du navigateur :

```
useragent {
    source => "ua"
    target => "useragent"
    remove_field => [ "ua"]
}
```

Sortie Elasticsearch

Une fois notre log parsé, nous voulons que Logstash l'envoie à Elasticsearch. Rien de plus simple avec la sortie Elasticsearch, puisqu'il suffit simplement de déclarer :

```
output {
    elasticsearch {}
}
```

Note: la sortie est configurée pour se connecter sur l'Elasticsearch locale, avec un mode de transport de type "node" par défaut. Il n'y a donc rien à paramétrer. Logstash nous permet de collecter nos logs, les structurer, les enrichir et enfin de les envoyer à Elasticsearch : **Fig.5.**

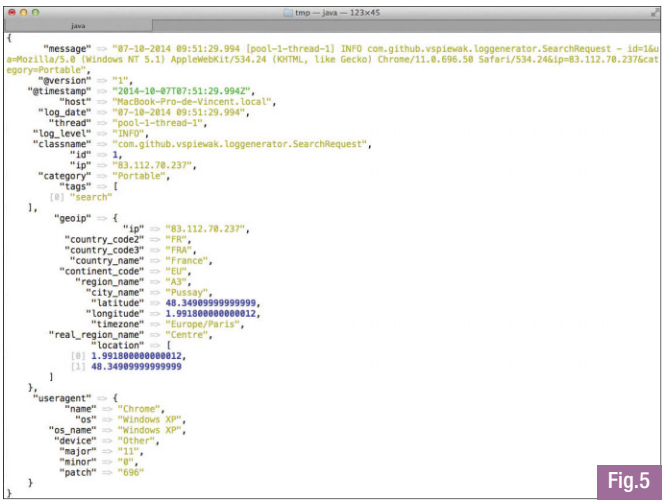


Fig.5

Elasticsearch

Elasticsearch est une base de données full-text orientée document, distribuée et offrant de la haute disponibilité notamment.
Logstash enverra nos logs dans des indices nommés "logstash-YYYY-MM-DD" (équivalent des tables dans le monde SQL traditionnel)
Pour installer Elasticsearch et le plugin “head” (offrant une interface web), rien de plus simple :

```
$ curl -O https://download.elasticsearch.org/elasticsearch/
elasticsearch/elasticsearch-1.3.4.tar.gz
$ tar xvzf elasticsearch-1.3.4.tar.gz
$ cd elasticsearch-1.3.4
$ bin/plugin --install mobz/elasticsearch-head
$ bin/elasticsearch
```

Vous pouvez surveiller l’état de santé d’Elasticsearch à l’adresse : http://localhost:9200/_plugin/head **Fig.6.**

Kibana

Kibana est une application de type “Single Page App” utilisant notamment la stack technique HTML 5, Angular JS et Twitter Bootstrap.
Dans notre exemple, nous utiliserons NGiNX, mais tout autre serveur Web capable de servir des fichiers statiques est possible :

```
curl  -O  https://download.elasticsearch.org/kibana/kibana/
kibana-3.1.1.tar.gz
tar xvzf kibana-3.1.1.tar.gz
sudo mv kibana-3.0.0milestone4 /usr/share/nginx/www/kibana
```

Vous pouvez désormais accéder à Kibana à l’url : http://localhost/kibana. **Fig.7.** La page d’accueil de Kibana dispose d’un menu en haut à droite permettant de charger, sauvegarder et partager vos dashboards via :

- Fichier JSON
- ElasticSearch
- Gist
- Permalien

Kibana offre 4 dashboards par défaut :

- Un dashboard générique Logstash (cf. photo),
- Un dashboard générique Elasticsearch,
- Un dashboard non configuré,
- Un dashboard vide.

Un dashboard se "branche" sur tous les index de votre cluster Elasticsearch, un index spécifique, ou les index dont le nom matche un pattern (cas de Logstash notamment).
L’interface reprend le système de grille de Twitter Bootstrap, décomposant les lignes en 12 colonnes, contenant des panels.

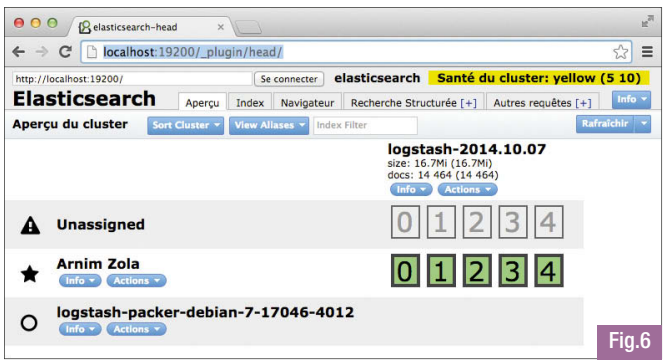


Fig.6

Il existe plusieurs types de panels à ajouter et configurer (recherche, période de temps, histogramme, camembert, carte, Markdown, ...)
L’icône “i”, disponible sur chaque panel, permet de visualiser la requête Elasticsearch.

Création d’un dashboard Kibana

Partez d’un Dashboard vierge en cliquant sur le lien "Blank Dashboard". N’oubliez pas de le sauvegarder avant de rafraîchir la page de votre navigateur ! **Fig.8.**
Ouvrez la fenêtre modale "*Dashboard settings*" en cliquant sur la roue crantée dans le coin supérieur droit :

- Onglet "*General*" : choisissez un titre et le thème ("*dark*" ou "*light*"),
- Onglet "*Index*" : sélectionnez "day" pour le champ "*Timestamping*",
- Onglet "*Controls*" : cochez toutes les cases afin de vous donner le maximum d’options,
- Fermez la fenêtre modale en cliquant sur “save”.

Vous pouvez sélectionner une période de temps à afficher dans le menu Time filter (exemple: TimeFilter > Last 15m, TimeFilter > Auto-Refresh > 5s). La barre de recherche utilise le format Lucene Query ou une expression régulière.
En cliquant sur le bouton "+" du champ recherche, créez 3 barres de recherche contenant les requêtes suivantes :

- *
- tags:”search”
- tags:”sell”

Vous pouvez épingler des recherches et leur attribuer un alias.

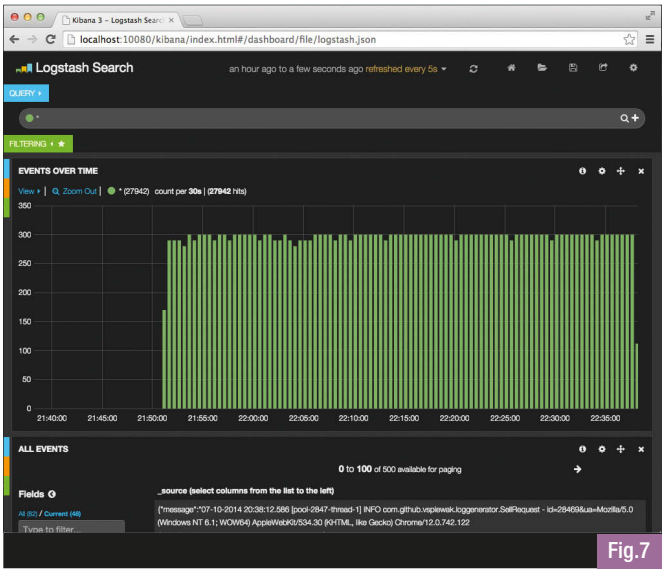


Fig.7

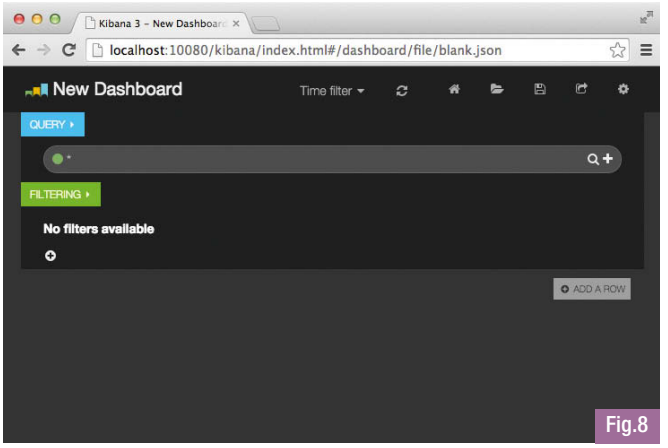


Fig.8

Cliquez sur chaque rond de couleur et sauvegardez les alias suivants :

- All
- Search
- Sell

Ligne 1 - Objectifs, Histogramme et Carte

Ajoutez une ligne en cliquant sur le bouton “Add row”, puis :

- Un panel Histogram de taille 7
- Un panel Bettermap de taille 5

Ligne 2 - Terms

Ajoutez une nouvelle ligne contenant:

- Un panel terms, field “tags”, taille 2
- Un panel terms, field “brand.raw”, length 5, taille 2
- Un panel terms, field “name.raw”, length 5, taille 2
- Un panel terms, field “geoip.city_name.raw”, length 5, taille 2
- Un panel terms, field “useragent.device.raw”, length 5, taille 2
- Un panel terms, field “useragent.name.raw”, length 5, taille 2

Ligne 3 - Données

Ajoutez une dernière ligne contenant uniquement un panel “table” de taille 12.

Drill down

En cliquant sur les différents éléments des deux dernières lignes, vous créez des filtres dynamiquement (icônes en forme de loupe ou de croix). Vous retrouverez ces filtres en dessous des barres de recherches, vous permettant de les éditer ou les désactiver. Fig.9.

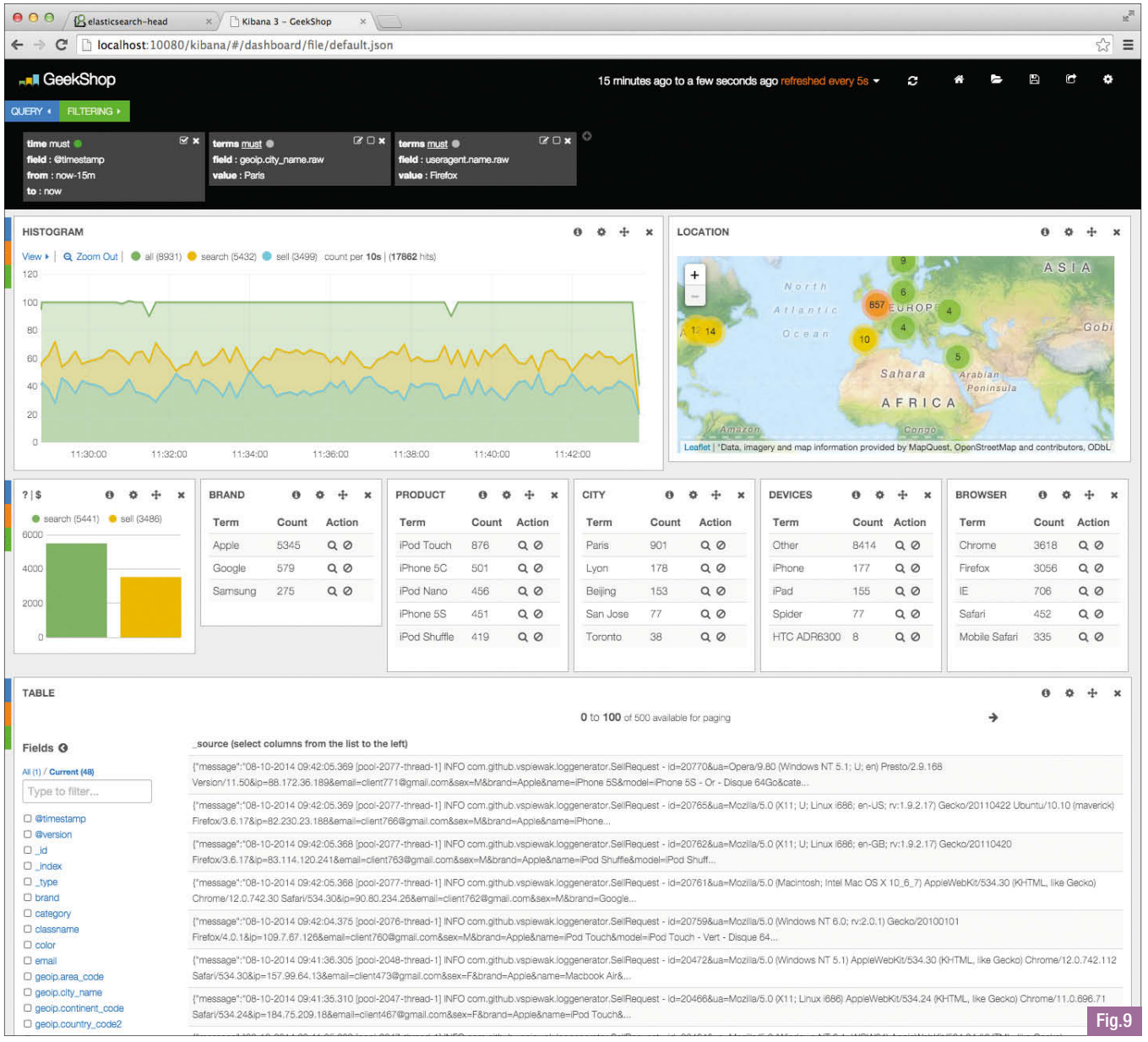


Fig.9

Test Driven Infrastructure avec Chef

Chef est un outil de gestion de configuration permettant de décrire dans un langage clair et compréhensible l'ensemble de son infrastructure ("Infrastructure as code"). Chef permet ainsi d'installer et de maintenir l'ensemble des serveurs et cela de manière automatique.

Emmanel Sciara
Consultant agile chez Xebia
@esciara

Matthieu Nantern
Consultant chez Xebia
@mNantern

Ce tutoriel montre, à travers l'installation de lighttpd qui servira une page comprenant le texte « Wonderstuff Design is a boutique graphics design agency. », un cas d'utilisation simple de Chef. Mais comme tout code il est très important de le tester afin de s'assurer de sa maintenabilité et de son évolutivité.
Cet exemple montre pas à pas comment monter un environnement pour faire du Test Driven Infrastructure avec l'outil de provisioning Chef, en suivant l'exemple du livre Test-Driven Infrastructure with Chef, 2nd Edition by Stephen Nelson-Smith.
Ce cas pratique est destiné aux développeurs débutants ou expérimentés dans la création de cookbooks Chef. Il faut à minima une compréhension des concepts introduits par Chef.
En temps normal, il faut s'assurer que les tests créés ne passent pas avant d'écrire le code qui va les faire passer.
Les outils utilisés dans cet article sont (par ordre d'apparition) :

- Ruby 1.9.x
- Bundler
- VirtualBox
- Vagrant
- Berkshelf
- Test-Kitchen
- ServerSpec
- ChefSpec

PRÉPARER SON ENVIRONNEMENT DE DÉVELOPPEMENT

Installation

L'installation complète de l'environnement de développement se déroule en 7 étapes :

- La première étape pour installer un environnement sain pour le développement de cookbooks est d'installer Ruby. Pour cela suivez l'article très complet : http://misheska.com/blog/2013/12/26/set-up-a-sane-ruby-cookbook-authoring-environment-for-chef/ . Nous allons ici travailler sur OS X.
- Installer ensuite VirtualBox en téléchargant l'application sur https://www.virtualbox.org/wiki/Downloads (version utilisée : 4.3.10)
- Installer Vagrant en téléchargant l'application sur http://www.vagrantup.com/downloads.html (version utilisée : 1.5.3). Vagrant permettra, avec l'aide de Virtualbox, de lancer automatiquement des machines virtuelles afin d'exécuter nos tests d'intégration.
- Les dépendances entre les différents cookbooks seront gérées avec Berkshelf (version 3.1.3). Pour plus d'information voir http://berkshelf.com/. Ajouter Berkshelf au fichier Gemfile que vous devez créer dans le répertoire racine du cookbook. Berkshelf sera alors automatiquement installé par Bundler lors d'un bundle install. Nous allons utiliser le même processus pour tous les autres gems et outils de ce tutoriel.

```
$ vim Gemfile
```

- Ajouter le contenu suivant :

```
source 'https://rubygems.org'

gem 'berkshelf', '3.1.3'
```

- Nous allons ensuite utiliser Test-Kitchen (version 1.2.1, http://kitchen.ci/) qui permet de créer les VM avec Vagrant et lancer les tests automatiquement. Nous utiliserons son plugin kitchen-vagrant (version 0.15.0) permettant à Test-Kitchen de piloter le cycle de vie des VM en utilisant Vagrant. Ajouter Test-Kitchen au Gemfile ainsi que son plugin kitchen-vagrant :

```
gem 'test-kitchen', '1.2.1'
gem 'kitchen-vagrant', '0.15.0'
```

- Il ne reste plus qu'à installer tous les composants avec Bundler :

```
$ bundle install
```

Création du squelette de cookbook

Créer le fichier metadata.rb dans le répertoire racine :

```
$ vim metadata.rb
```

Contenu de metadata.rb :

```
name "wonderstuff"
version "0.1.0"
```

Créer le squelette de cookbook avec berks init (ne pas écraser le Gemfile que nous venons de créer) :

```
$ bundle exec berks init
```

Terminer le squelette :

```
$ mkdir recipes
$ touch recipes/default.rb
```

LANCER DES TESTS À L'INTÉRIEUR D'UN NOEUD AVEC BUSSER

Busser est un framework de setup et d'exécution de tests sur les nœuds créés par Test-Kitchen. Il utilise une architecture de plugin permettant d'ajouter le support de différents outils/stratégies de test tels que MiniTest, Cucumber, Bash, etc.
L'installation de Busser sur le nœud et l'exécution des tests sont

complètement prises en charge par Test-Kitchen. Par défaut, Test-Kitchen va regarder dans le répertoire test/integration pour trouver les fichiers de test à lancer. Ce répertoire est organisé comme suit :

- Directement sous test/intégration se trouveront des répertoires dont les noms définiront les suites de tests. Les suites de tests à lancer sont déclarées dans le .kitchen.yml sous suites: - name: <nom de la suite>. Pour plus d'informations sur .kitchen.yml se référer à http://docs.opscode.com/config_yml_kitchen.html.
- Sous chaque répertoire de suites de tests se trouveront des répertoires dont les noms sont ceux des plugins Busser utilisés.
- Sous chaque répertoire de plugin Busser se trouveront les fichiers de tests à lancer sur les nœuds.

Organisation des répertoires de suites de tests à lancer par Busser :

```
test/integratio
  |_ do_something <-- suite name (more on this later)
     |_ cucumber <-- busser (AKA the "tester")
        |_ my_test_files <-- a test
```

Pour plus d'informations sur Busser, se référer à https://github.com/test-kitchen/test-kitchen/wiki/Getting-Started#structure.

ECRITURE D'UN TEST D'INTÉGRATION UTILISANT SPECSERVER

Introduction à ServerSpec

ServerSpec permet d'écrire des tests RSpec permettant de tester que vos serveurs sont correctement configurés.
ServerSpec teste l'état des serveurs par accès SSH. Pas besoin d'y installer un agent. De plus, Busser se charge entièrement de tout le cycle d'exécution des tests sur les noeuds créés par Test-Kitchen.
Plus d'informations disponibles sur http://serverspec.org/.

Configuration de ServerSpec

ServerSpec a besoin de savoir le type d'OS sur lequel il va effectuer les tests avant de lancer ces derniers. Pour cela, créons un fichier de support spec_helper.rb qui fera ce travail. Il servira de référence ensuite dans tous les fichiers de tests effectués :

```
$ mkdir test/integration/default/serverspec
$ vim test/integration/default/serverspec/spec_helper.rb
```

Contenu de spec_helper.rb :

```
# encoding: UTF-8
require 'serverspec'
require 'pathname'

set :backend, :exec
```

Ecriture du test ServerSpec

Créez les répertoires qui vont permettre la prise en charge des tests par Busser et créez le fichier contenant le test.

```
$ mkdir test/integration/default/serverspec/localhost
$ vim test/integration/default/serverspec/localhost/readable
_services_spec.rb
```

Contenu de fichier readable_services_spec.rb :

```
# encoding: UTF-8
require 'spec_helper'

describe 'Wonderstuff Design' do

  it 'should install the lighttpd package' do
    expect(package 'lighttpd').to be_installed
  end

  it 'should enable and start the lighttpd service' do
    expect(service 'lighttpd').to be_enabled
    expect(service 'lighttpd').to be_running
  end

  it 'should render the Wonderstuff Design web page' do
    expect(file('/var/www/index.html')).to be_file
    expect(file('/var/www/index.html')).to contain 'Wonderstuff
Design is a boutique graphics design agency.'
  end

end
```

Vous pouvez vérifier que les tests se lancent et échouent en lançant la commande :

```
$ bundle exec kitchen verify ubuntu
```

ECRITURE D'UN TEST UNITAIRE CHEFSPEC

Introduction à ChefSpec

ChefSpec est un framework de tests unitaires permettant de tester des cookbooks Chef. Il simplifie l'écriture de tests permettant d'obtenir un feedback rapide sur les changements de cookbooks sans avoir besoin de machine virtuelle ni de serveurs dans le Cloud.
ChefSpec est une extension de Rspec. Il lance votre cookbook localement en utilisant Chef Solo sans avoir besoin de faire converger un noeud. Cela a deux avantages principaux :

- C'est très rapide !
- Vos tests peuvent faire varier les attributs des noeuds, OS, les résultats de recherche pour vérifier le comportement des cookbooks dans diverses conditions.

Plus d'information disponible sur https://github.com/sethvargo/chefspec ou encore http://docs.opscode.com/chefspec.html.

Installation de ChefSpec

Ajouter ChefSpec au Gemfile :

```
gem 'chefspec', '4.0.1'
```

Lancer l'installation du Gem :

```
$ bundle update
```

Configuration de ChefSpec

Les tests ChefSpec se trouvent dans le répertoire spec. On peut dire à ChefSpec de faire les tests en simulant un OS spécifique. Nous allons faire cela dans cet exemple particulier en nous cantonnant à Ubuntu 12.04. Pour cela, nous allons spécifier l'OS de manière globale en utilisant le fichier de support spec_helper.rb. Il est également possible de lancer les tests en simulant différents OS grâce à Rake. Pour plus d'information sur la spécification de l'OS cible de ChefSpec, se référer à https://github.com/sethvargo/chefspec#configuration .
Créez le répertoire spec et le fichier spec_helper.rb :

```
$ mkdir spec
$ vim spec/spec_helper.rb
```

Contenu de spec_helper.rb :

```
# encoding: UTF-8
require 'chefspec'
require 'chefspec/berkshelf'

RSpec.configure do |config|
  config.platform = 'ubuntu'
  config.version = '12.04'
end
```

Ecriture du test ChefSpec

Créez les répertoires et le fichier contenant le test.

```
$ mkdir -p spec/unit/recipes/
$ vim spec/unit/recipes/default_spec.rb
```

Contenu de fichier default_spec.rb :

```
# encoding: UTF-8
require 'spec_helper'

describe 'wonderstuff::default' do
  let(:chef_run) do
    runner = ChefSpec::Runner.new(
      log_level: :error
    )
    Chef::Config.force_logger true
    runner.converge('recipe[wonderstuff::default]')
  end

  it 'installs the lighttpd package' do
    expect(chef_run).to install_package('lighttpd')
  end

  it 'creates a webpage to be served' do
    expect(chef_run).to render_file('/var/www/index.html').
with_content('Wonderstuff Design is a boutique graphics
design agency.')
  end

  it 'starts the lighttpd service' do
    expect(chef_run).to start_service('lighttpd')
  end

  it 'enables the lighttpd service' do
    expect(chef_run).to enable_service('lighttpd')
  end
end
```

Vérifiez que les tests se lancent et échouent en lançant la commande :

```
$ bundle exec rspec -fd
```

ECRITURE DU COOKBOOK

Maintenant que tous les tests sont en place, il ne reste plus qu'à écrire le cookbook.

```
$ vim recipes/default.rb
```

Contenu de fichier default.rb :

```
# encoding: UTF-8
#
# Cookbook Name:: wonderstuff
# Recipe:: default
#
# Copyright (C) 2014 Emmanuel Sciara
#
# All rights reserved - Do Not Redistribute
#

package 'lighttpd'

service 'lighttpd' do
   action [:enable, :start]
end

cookbook_file '/var/www/index.html' do
   source 'wonderstuff.html'
end
```

Cette recette utilise un fichier html comme ressource qu'il copie sur le serveur.

```
$ mkdir -p files/default/
$ vim files/default/wonderstuff.html
```

Contenu de fichier wonderstuff.html :

```
<html>
  <body>
    <p>Wonderstuff Design is a boutique graphics design agency.</p>
  </body>
</html>
```

Donnez une adresse IP privée à votre VM vagrant afin de pouvoir accéder à la page html du serveur. Pour cela modifier le fichier .kitchen.yml :

```
$ vim .kitchen.yml
```

Remplacez la portion du fichier .kitchen.yml comme ci-dessous :

```
suites:
  - name: default
    driver:
      network:
        - ["private_network", {ip: "192.168.33.40"}]
    run_list:
      - recipe[wonderstuff::default]
    attributes:
```

Détruisez la VM existante pour que le changement d'IP soit pris en compte :

```
$ bundle exec kitchen destroy ubuntu
```

Et voilà ! Voyez les tests serverspec passer en lançant la commande suivante :

```
$ bundle exec kitchen verify ubuntu
```

et les tests chefspec avec la commande :

```
$ bundle exec rspec -fd
```

Vous pouvez voir la page sur http://192.168.33.40/

Premiers pas avec XL Deploy

XL Deploy est une solution éditée par XebiaLabs destinée aux Devs et aux Ops. Elle permet de prendre en charge la phase délicate du déploiement de l'application et de sa configuration dans les différents environnements.

Benoit Moussaud
Technical Director Southern Europe - XebiaLabs
@bmoussaud

La première spécificité de la solution XL Deploy est qu'elle est basée sur le modèle UDM (Unified Deployment Model). Ce modèle stipule que pour avoir une application déployée (DeployedApplication), il faut créer un déploiement qui prend en entrée :

- Un **package** sous la responsabilité des Devs. Il regroupe l'ensemble des éléments qui constituent une version d'une application. Ces éléments sont :
  - Des *artefacts*, les fichiers ou les répertoires de fichiers : .exe, .war, fichiers SQL, fichier de configuration .properties etc.
  - Des *ressources*, les éléments de configuration tels que les datasources, virtualhost etc.
  - Des *méta-données* telles le nom de l'application et sa version.

Note : Le package doit toujours être complet, avec l'ensemble des éléments.

- Un **environnement** sous la responsabilité des Ops. Il est caractérisé par :
  - Un ensemble de *containers,* des éléments d'infrastructure ou de middleware décrits du point de vue du déploiement (machine, serveur d'application, serveur web, base de données...) sur lesquels on va déployer l'application,
  - Un ensemble de *dictionnaires* qui décrivent la configuration à appliquer. Exemple : username, password, répertoire, ports, placeholders à remplacer dans les fichiers de configuration.

La seconde spécificité de XL Deploy est la génération automatique des plans de déploiement. Basé sur la spécification de ce déploiement (un package, un environnement, des dictionnaires), sur l'état initial (comme une précédente version déjà déployée), et l'état cible (je veux déployer cette version sur cet environnement avec cette configuration), le moteur « Autoflow » de XL Deploy calcule et applique les différentes étapes à effectuer sur les différents middlewares afin de refléter le nouvel état. La connaissance des actions à effectuer est définie dans le serveur XL Deploy sous forme de règles de déploiement qui seront appliquées quel que soit l'environnement (l'ensemble des équipes partage donc les mêmes règles de déploiement). Une règle s'exprime de la façon suivante : si un élément du package est créé (modifié ou supprimé) alors il faut effectuer les actions suivantes.

Exemple : si un fichier WAR ciblé sur un serveur Tomcat est modifié, alors il faut

- Arrêter le serveur Tomcat,
- Supprimer l'ancienne version du fichier WAR,
- Déposer le nouveau fichier WAR,
- Démarrer le serveur Tomcat.

XebiaLabs propose des ensembles des règles prêtes à l'emploi pour les principaux middleware Java (IBM WebSphere, Oracle Weblogic, Apache Tomcat,RedHat JBoss), le monde Microsoft (IIS, BizTalk, Services Windows), les serveurs Web, les bases de données, des ESB (WebSphere MQ, Oracle Service Bus, Tibco) et les load-balancers (F5, NetScaler).

La dernière spécificité d'XL Deploy est sa capacité à se connecter aux machines distantes en utilisant les protocoles standard (SSH dans le monde Unix et WinRM dans le monde Windows). Il n'est donc pas nécessaire d'installer et de configurer un agent propriétaire sur l'ensemble du parc de serveurs. Nous allons maintenant décrire les différentes actions à réaliser pour déployer une application Java (une application Microsoft, par exemple .NET, suivra les mêmes principes).

Création d'un package

Tout d'abord il faut créer le package de notre application. Nous allons partir d'une application Java classique. Elle est composée de 2 fichiers War (PetClinic.war et PetClinic-Backend.war) – type jee.War, une définition d'une DataSource - type tomcat.DataSourceSpec et d'un répertoire qui contient des fichiers de configuration – type tomcat.ConfigurationFolder. Le fichier *Manifest* correspondant à cette description est le suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<udm.DeploymentPackage version="1.0-20140829-182124" application
="xldeploy-petclinic-tomcat">
  <deployables>
    <jee.War name="petclinic" file="PetClinic.war" />
    <jee.War name="petclinic-backend" file="PetClinic-Backend.war" />
    <tomcat.DataSourceSpec name="petclinicDS">
<jndiName>jndi/pet</jndiName>
      <username>{{DATABASE_USERNAME}}</username>
      <driverClassName>com.mysql.jdbc.Driver</driverClassName>
      <password>{{DATABASE_PASSWORD}}</password>
      <url>{{DATABASE_URL}}</url>
    </tomcat.DataSourceSpec>
    <tomcat.ConfigurationFolder name="configuration" file="config" />
  </deployables>
</udm.DeploymentPackage>
```

Le package doit être indépendant de l'environnement cible. Les éléments paramétrables sont déclarés en utilisant un *placeholder* avec le format *mustache* {{..}}. Dans l'exemple ci-dessus le username, le password et l'url de la datasource sont respectivement gérés par les placeholder DATABASE_USERNAME, DATABASE_PASSWORD et DATABASE_URL. En revanche le nom JNDI est le même quel que soit l'environnement; il peut donc y être déclaré dans le *Manifest* du package.

La structure du package est donc la suivante :

```
├── PetClinic-Backend.war
├── PetClinic.war
├── configuration
│   └── config
│       ├── log4j.properties
│       ├── petclinic-backend.properties
│       └── petclinic.properties
└── deployit-manifest.xml
```

Ce répertoire est ensuite transformé en une archive (.dar) et importé dans XL Deploy pour y être stocké dans son référentiel. **Fig.1.**

L'application xldeploy-petclinic-tomcat version 1.0-20140829-182124 est importée. Afin de faciliter la création des packages et leur import dans XL Deploy, XebiaLabs propose des intégrations avec les outils de *build* tels que Maven ou MS-Build et les serveurs d'intégration continue comme Jenkins ou TFS.

Définition d'un environnement

Avant de définir l'environnement cible, il faut d'abord décrire l'infrastructure (du point de vue du déploiement). Elle sera composée d'un host 'test-machine' – type overthere.SshHost sur lequel est installé un serveur Tomcat 'tomcat.test' – type tomcat.Server et son host virtuel par défaut tomcat.test.vh – type tomcat.VirtualHost Fig.2.
Ensuite il faut définir un dictionnaire qui portera les valeurs de configuration pour cet environnement.
Pour les données sensibles, XL Deploy propose un autre type de dictionnaire de type 'udm.EncryptedDictionary' dans lequel le mot de passe de la datasource sera défini.
L'environnement 'Test' sera donc composé des 3 *containers* et des 2 dictionnaires qui viennent d'être créés.

Déploiement d'une application

Le déploiement est la mise en relation entre une version d'une application (xldeploy-petclinic-tomcat/1.0-20140829-182124) et un environnement (Test). La phase de mapping va associer chaque élément du package sur un ou plusieurs containers de l'environnement Fig.4.

> **Remarque :** cette modélisation montre que le déploiement concerne un artefact de type 'jee.War 'sur un container de type 'tomcat.VirtualHost' bien loin d'une vision infrastructure où le fichier 'petclinic.war' serait associé à une machine 'test-machine'.

Les propriétés Url, Username et Password de la datasource 'petclinicDS' sont entièrement configurées avec les différentes valeurs proposées par les 2 dictionnaires. Basé sur cette spécification, XL Deploy va générer la tâche de déploiement initiale suivante : Fig.5. Lors d'une mise à jour avec une nouvelle version, XL Deploy va analyser les éléments modifiés (par exemple un seul des 2 wars) et générer le plan suivant.
Si en même temps, le username de la datasource a été modifié dans le dictionnaire, XL Deploy va le détecter et inclure la re-création de la datasource dans le plan de déploiement Fig.6.
L'exécution de ces tâches par le moteur XL Deploy consiste pour chacune des étapes (Step) à se connecter sur la machine distante en utilisant Ssh ou WinRM et à exécuter la commande comme le ferait un opérateur.

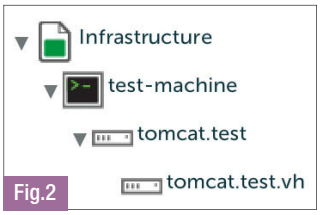


Fig.2

Définition de l'infrastructure

Et mon intégration continue ?

Quel que soit le moteur d'intégration continue choisi (Jenkins, Bamboo, TFS), les équipes ont la plupart du temps intégré une partie déploiement - appelée à la suite de la tâche de Build- via l'utilisation de script custom ou de plugins. Si effectivement cela remplit la fonction de déploiement, celle-ci n'est pas utilisée par l'ensemble des acteurs de la chaîne (de Dev à Ops) et elle n'intègre pas les spécificités des différents environnements (changement de topologies) ou les besoins propres aux plateformes de production (exemple intégration de fonction de sauvegarde ou de gestion des load-balancers). XL Deploy propose une intégration avec les différents serveurs d'intégration continue en proposant les fonctions suivantes : constitution du package de déploiement, import du package, et déclenchement du déploiement sur l'environnement idoine. Rapidement les équipes mettent en place du déploiement continu : le développeur « committe » son code, l'intégration continue se déclenche et construit l'application, lance les tests unitaires et déploie l'application. De cette manière, la procédure de déploiement est testée plusieurs fois par jour coté Devs et elle est rendue plus robuste pour les équipes Ops qui déclencheront leurs déploiements avec l'interface graphique ou l'interface ligne de commande (CLI) via le même outil.
XL Deploy est une solution transverse qui prend non seulement en charge le délicat sujet du déploiement des applications mais permet également aux équipes Devs & Ops de partager un vocabulaire commun (Environnement, Application, Version et Configuration avec les dictionnaires) et ainsi de créer entre les deux équipes une coopération efficace, qui est souvent une des clés de l'approche DevOps !

XebiaLabs the deployment automation company

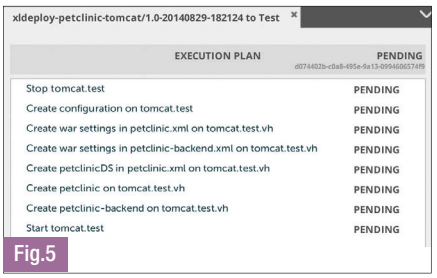


Fig.5

Tache de déploiement initial

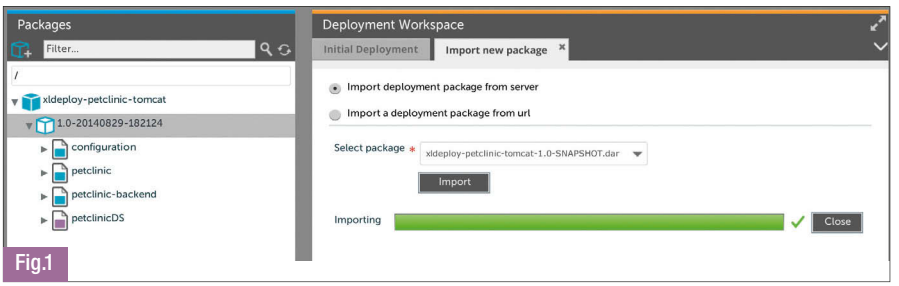


Fig.1

Import d'un package dans XL Deploy

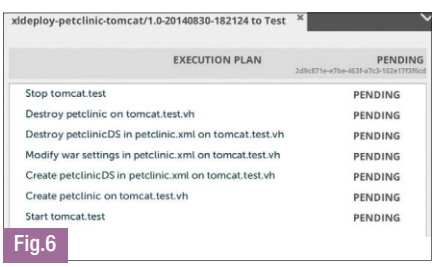


Fig.6

Tache de déploiement (War et Datasource modifiés)

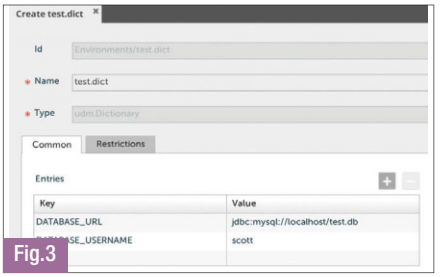


Fig.3

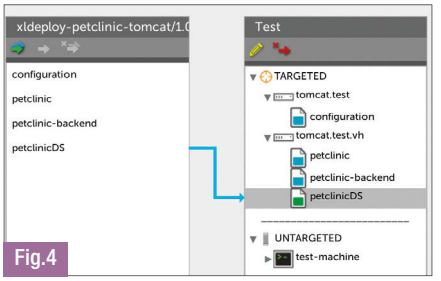


Fig.4

Figure 3 Définition des dictionnaires

Mettez vos builds dans le Cloud avec Visual Studio Online

Visual Studio Online (Team Foundation Server en mode SAAS) prend de plus en plus ses marques auprès des sociétés et des développeurs indépendants. Sa "gratuité", sa facilité de mise en place en font une solution de choix dans bien des contextes.

Jeremy LANDON
infinitesquare

Visual Studio Online offre entre autres : un contrôle de source, un gestionnaire de projet, un gestionnaire de test... mais aussi un serveur de build ! Souvent délaissé, le serveur de build est devenu une plus-value énorme lors de la réalisation d'un projet informatique. Le but du serveur de build ? Permettre d'automatiser les actions qu'aurait fait manuellement un développeur, testeur et/ou opérationnel. Cela se traduit le plus souvent par de la génération d'applications (avec lancement des tests) ou du déploiement.

Des bénéfices immédiats

Les builds peuvent être lancées manuellement, à fréquence régulière (pour du déploiement en environnement de dev par exemple ?) ou en intégration continue (à chaque envoi de source sur le serveur par un développeur), cette dernière devrait être aujourd'hui présente sur la quasi-totalité des projets ! La mise en place est simple, elle vérifie que l'application compile, répond aux exigences et détecte les régressions en lançant les tests et potentiellement déploie directement dans un environnement de test ! La plus-value est colossale et au vu de la facilité d'implémentation dans un cadre de vérification, il serait vraiment dommage de s'en priver !

Une solution adaptée pour tous

Après avoir compris l'utilité d'avoir un serveur de build, il faut désormais en posséder un. De manière générale, le serveur de build est installé manuellement, et donc ce même serveur sollicite une machine. Pour une entreprise, le problème peut ne pas se poser, mais pour un usage personnel, l'implémentation est tout de même un peu lourde... Avec Visual Studio Online ce tracas est de l'histoire ancienne.

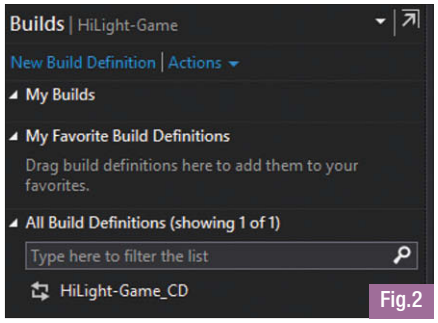


Fig.2

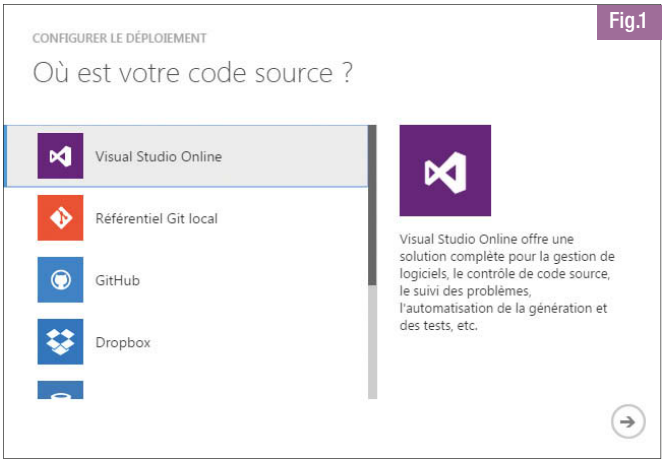


Fig.1

Concrètement pour chaque compte Visual Studio Online ouvert Microsoft offre 60min de build gratuit par mois ! On dispose d'un serveur de build dans Azure qui nous permettra de générer nos applications. 60min par mois pour un indépendant ça peut être suffisant. Pour une entreprise, cela ne couvre la plupart du temps même pas les besoins journaliers ! En effet les builds sont des ressources dites partagées: cela signifie que le temps de build est consommé par l'ensemble des développeurs. Après les 60min, trois possibilités s'offrent à nous :

- Attendre le mois suivant pour réaliser vos builds,
- Passer sur un serveur de build on-premise,
- Passer en mode payant en étant facturé à la minute de build pour s'adapter aux besoins de chacun.

On l'aura compris, cette solution offerte par Visual Studio Online n'est pas favorable dans tous les cas. Heureusement Microsoft laisse de la liberté et cette machine de build reste « optionnelle », il est tout à fait possible d'utiliser son propre serveur de build (placé dans Azure par exemple ?) qui sera mappé sur notre compte Visual Studio Online.

Déployer sur Azure n'aura jamais été aussi simple

Azure nous propose de pouvoir se provisionner de la ressource en quelques minutes et selon nos besoins. Entre autres, il propose de fournir des sites Web directement prêts à l'emploi. Il suffit de le déployer : directement via Visual Studio, via du PowerShell en utilisant le SDK Azure, ou plus simplement en réalisant une build ! Chaque site Web Azure peut être lié à un contrôleur de source (Visual Studio Online, GitHub, BitBucket...) Fig.1.

A quoi cela sert-il ? En prenant le cas de Visual Studio Online, il suffira de sélectionner le projet du site Web, et, automatiquement, Azure nous créera une build qui apparaîtra quelques secondes plus tard dans la liste des builds du projet. Cette build sera :

- En intégration continue,
- Préconfigurée,
- Prête à l'emploi ! Fig.2

En bref, réaliser une build pour un site sur Azure est d'une simplicité enfantine, il y a désormais aucune excuse pour ne pas en faire !

Gagnez en efficacité

Avec Visual Studio Online, Microsoft étend le principe du SAAS à son maximum en nous offrant une machine de build déjà prête à l'emploi.

Microsoft comme à son habitude offre une synergie maximale avec ses autres outils, et la génération de build de déploiement pour les sites Web Azure est d'une efficacité redoutable. Productivité, qualité et confort sont donc au rendez-vous facilement et surtout rapidement, alors aucune raison de s'en priver !

DevOps avec Visual Studio, Azure, TFS, VSO

Les processus de développement de logiciels évoluent constamment pour s'adapter aux nouvelles contraintes de chacun. Ces évolutions passent par des nouvelles méthodologies et des nouveaux outils ayant pour but d'aider et améliorer la qualité et productivité des trois acteurs d'un projet de développement logiciel à savoir le métier, le développement et l'opérationnel.

Vivien FABING
Jérémy LANDON
Consultants ALM
INFINITE SQUARE

- Le métier, dont le rôle est d'apporter une réponse à une problématique ou à un besoin utilisateur,
- Le développement dont la mission est dans un premier temps de répondre au plus vite aux demandes du métier, tout en gardant une bonne qualité de livraison (fonctionnelle et technique). Cela passe par des livraisons fréquentes de nouvelles fonctionnalités,
- Enfin, les opérations, qui mettent en place le développement en production pour répondre au besoin métier initial; elles garantissent la stabilité et la disponibilité du produit.

La communication étant l'une des clés de la réussite d'un projet informatique, ces trois acteurs sont (ou devraient être) en communication permanente. Malheureusement avec trois équipes complètement différentes et trois langages propres et parfois uniques, des frontières naturelles se créent empêchant une bonne communication et une bonne compréhension des besoins de chacun.
Les conséquences sont multiples et parfois désastreuses dans un projet informatique :

- Un développement qui ne répond plus au besoin du métier,
- Un développement qui se soucie peu de la production,
- Des retours des opérations rarement complets pour le développement,
- Une vraie difficulté à créer des cycles de développement fluides,
- Une mise en production tardive ou bancale...

La liste est malheureusement encore longue.
Pour pallier en partie ces problèmes, les désormais très populaires méthodes agiles sont apparues. Celles-ci permettent, entre autres, une meilleure communication entre le métier et le développement, mais pour la communication entre le développement et l'opérationnel il y a DevOps !

QU'EST-CE QUE DEVOPS ?

Le terme DevOps (contraction des mots **Dév**eloppement et **Opérations**) est apparu en 2009. C'est avant tout une idée selon laquelle chaque composante d'une organisation doit collaborer efficacement afin d'atteindre un même objectif. La démarche DevOps est venue d'un constat : la communication et la collaboration sont l'une des clés de réussite d'un projet, et la séparation entre le développement et l'opérationnel était souvent la source d'échecs.
DevOps est aujourd'hui un mouvement, une culture et une extension des démarches agiles ayant pour objectif d'obtenir une cohésion du système d'information sur les besoins de l'entreprise. Concrètement, là où les démarches agiles tentent de concilier le métier et le développement vers un même discours et objectif, DevOps a pour objectifs d'améliorer la communication et la collaboration entre le développement et l'opérationnel. Cet objectif est atteint par l'utilisation de nouveaux outils, d'une meilleure communication et collaboration.

Développement/Opérationnel : un objectif final commun, mais des cultures différentes.

Le problème de collaboration et de communication entre le développement et l'opérationnel est connu et récurrent, mais finalement logique.
Le rôle du développement est de livrer de nouvelles fonctionnalités, maximiser le changement et innover : pour ce faire la réponse aux exigences et souvent faites par la production d'un livrable qui n'a pas connaissance des contraintes opérationnelles.
L'opérationnel cherchera quant à lui plutôt à obtenir une stabilité et limiter les risques (toute nouveauté est un risque à prendre en compte).
Les deux cultures sont opposées et créent des conflits : si le développement réalise des changements, l'opérationnel tentera de les minimiser pour limiter le risque d'instabilité. Un conflit dû à la volonté de combler les besoins de chacun, mais qui, au final, empêche le bon déroulement de l'objectif commun. Car finalement développement et opérationnel partagent le même objectif : fournir une solution stable et fonctionnelle, le tout dans un délai imparti. En bref, satisfaisante et répondant au mieux au besoin du client final.
DevOps a pour objectif de répondre aux besoins de chacun : la rapidité pour le développement tout en maintenant une bonne stabilité. Pour ce faire trois piliers sont mis en avant :

- L'infrastructure-as-code : le but est de *coder* la création des infrastructures, cette industrialisation permet d'obtenir des tâches répétables facilement (ex : création d'un environnement de test), d'améliorer la productivité et la fiabilité.
- Livraison continue : pour apporter au plus vite de la valeur ajoutée aux produits et diminuer le Time-To-Repair (TTR : délai de réparation d'un dysfonctionnement) par un principe simple : plus un bug est découvert tôt, plus il sera simple à corriger.
- Collaboration : donner accès à chacun à différentes données qui leur seraient utiles. Ex : fournir rapidement et sur demande une infrastructure de test tout en maîtrisant les coûts, ou encore accéder facilement aux rapports d'erreurs pour développement.

Tout comme la démarche Agile, la démarche DevOps repose sur différents outils, mais surtout sur une acceptation aux changements des différents acteurs. Il peut être compliqué et couteux de faire une chaîne DevOps par soi-même.
La gamme Visual Studio ALM propose déjà beaucoup d'outils depuis 2005, notamment pour fluidifier la communication Métier > développeurs > testeurs. En 2014, voici les nouveautés pour mettre en place rapidement et efficacement une démarche DevOps !

DevOps et l'Agilité

De nos jours, lorsque l'on est confronté au développement d'un logiciel informatique, il est devenu quasi impossible de ne pas avoir entendu parler des méthodes agiles.
DevOps est quant à lui un terme plus récent, mais qui commence petit à petit à se faire sa place parmi les concepts fondamentaux améliorant le cycle de vie du développement d'une application.
Inutile de tergiverser indéfiniment, ces 2 concepts sont loin d'être incompatibles et l'on pourrait même dire qu'ils sont deux des chaînons d'une suite logique visant à améliorer la qualité d'un logiciel.

Dans la continuité de l'Agilité

L'agilité s'efforce de mettre tout en œuvre dans un seul et unique but : la satisfaction du client.

Pour cela, l'expérience montre que l'adoption de cycles de développement courts permet de réduire le délai entre le moment où une fonctionnalité est développée, et le moment où le client constate son bon fonctionnement. La réduction de ce délai évite le malheureusement bien connu « effet tunnel » qui prend souvent la forme d'une période de développement de plusieurs mois voire années, aboutissant généralement à un écart entre ce que l'équipe de développement aura construit et ce que le client imaginait initialement.

Cependant, l'adoption de ces cycles de développement courts augmente également la fréquence d'exécution de phases qui auparavant ne nécessitaient d'être exécutées qu'une ou deux fois : La phase de recette (tests), et la phase de mise en production (déploiement).

Ces deux phases étaient réalisées manuellement, généralement par une poignée de personnes dédiées voire une petite équipe. Dans le cas de la phase de déploiement, l'opérateur devait bien souvent suivre un processus détaillé sous forme de documentation. Cette opération s'avère bien souvent longue, fastidieuse et conduit à de nombreuses erreurs (erreurs manuelles, changement de processus de livraison non notifié aux opérateurs, etc.).

Dans un contexte Agile où il faut dérouler cette phase toutes les 2 semaines voire toutes les semaines, comment réduire le temps de déploiement et les risques dus au caractère manuel de l'action ? Une seule solution : l'automatisation. C'est particulièrement sur cette plateforme commune d'automatisation que vont pouvoir se retrouver développeurs et opérateurs : automatisation de la remontée d'informations, automatisation du processus de déploiement, etc.

DU SUIVI POUR PLUS DE MAÎTRISE

Le principal manque d'efficacité dans la collaboration entre le développement et l'opérationnel est lié à la récolte d'incidents de production. Que faire lorsqu'un incident arrive en production ? Quelles informations possèdent les développeurs ? Et comment détecter l'incident ? Microsoft apporte une solution pour répondre à ces questions : System Center Operations Manager et Application Insights.

System Center Operations Manager (SCOM) : la communication entre IT et développeurs

SCOM est un outil destiné aux IT offrant une surveillance de performance et d'événements de systèmes Windows. Gâce à SCOM, l'IT peut savoir à tout moment quand une application à subi un dysfonctionnement et suivre différentes données de performance sur celle-ci Fig.1.

Le point fort des solutions Microsoft est la synergie qu'elles apportent les unes aux autres. A la détection d'un dysfonctionnement, l'IT peut en quelques clics envoyer directement un rapport (**Work Item** de type Operationnal Issue) aux développeurs via Team Foundation Server (de façon totalement transparente)! Le Work Item créé contiendra entre autres un rapport d'erreurs auto généré (exception, heure, application cible ect.) mais surtout **l'Intellitrace** ! Pour rappel, l'intellitrace permet de récolter différentes informations durant l'exécution d'un programme dont l'historique d'exécution, permettant ainsi au développeur de se retrouver dans le même contexte (valeur des différentes variables, liste des exceptions, call stack détaillée...) que lors de la levée de l'exception.

Néanmoins la collecte d'information par Intellitrace a un coût non négligeable sur les performances d'une application, c'est pourquoi cette fonctionnalité est désactivable : **Monitoring -> Tasks -> Start/Stop Intellitrace Collection**, permettant ainsi de reproduire un bug dans un laps de temps réduit afin de ne pas affecter tous les utilisateurs. L'envoi du rapport de bugs ajoute le suivi d'état des alertes, l'IT devient (sans s'en rendre compte) désormais un signaleur de dysfonctionnement des plus efficaces et pertinents pour les développeurs Fig.2.

Grace à l'intégration de TFS dans SCOM, Microsoft offre un moyen de communication entre le développement et l'opérationnel simple et efficace. Cette fonctionnalité est aussi disponible pour les applications Java. Avec SCOM, les bugs sont détectés plus vite, signalés tout aussi rapidement; le développement profite de la signalisation rapide pour corriger le bug plus facilement, l'opérationnel profite de cette remontée pour avoir une application toujours plus stable. En bref tout le monde profite de SCOM et de sa synergie parfaite avec TFS. Dans la même optique Microsoft a sorti en Novembre 2013 Application Insights (uniquement pour Visual Studio Online).

Application Insights : tracer son application pour une meilleure réactivité

SCOM est capable de tracer les différents incidents de notre application, mais la dépendance à l'opérationnel est forte. Comment avoir un suivi temps réel des incidents en environnement de production sans impacter les performances ?

Cette tâche n'est pas simple à mettre en place sans outils appropriés.

Bien sûr chaque projet est accompagné (ou devrait être accompagné) de son log applicatif, faut-il encore pouvoir accéder simplement à ces logs qui sont très généralement dans un environnement accessible seulement par l'opérationnel, et disposer d'un contexte de reproduction simple.

Application Insights a pour objectif de mettre à disposition les incidents de production, mais cette fois-ci directement aux développeurs. Des solutions existent pourtant déjà, l'une des plus populaires étant BugSense.

Là où Application Insights se démarque, c'est qu'il ne se contente pas d'un simple log d'erreur, mais il peut tout aussi bien fournir **l'Intellitrace** !

En dehors de ça, Application Insights propose également d'autres fonctionnalités

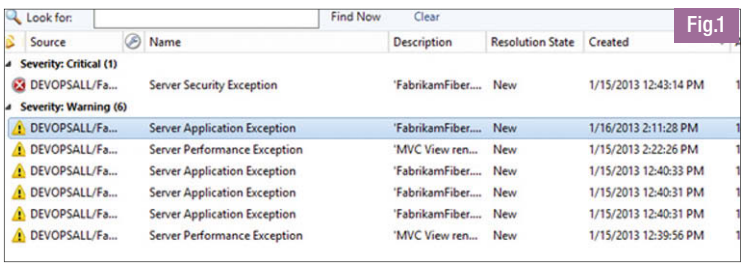


Fig.1

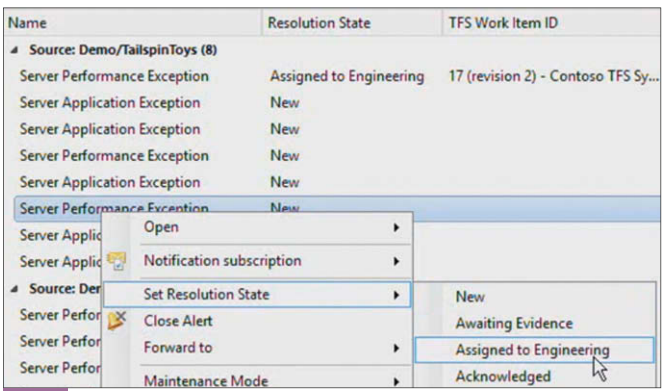


Fig.2

intéressantes qui étaient jusque-là manquantes dans l'outillage Microsoft telles que les analyses de disponibilité ou les statistiques d'utilisation.
Pour les analyses de disponibilité, il est possible en quelques clics de configurer Application Insights pour surveiller une application Web. Application Insights va ainsi, à intervalles réguliers, effectuer plusieurs requêtes sur l'applicatif et enregistrer le temps de réponse, proposant ainsi dans le temps des graphiques d'analyses de performances. Application Insights est disponible pour les applications .NET (Asp.net, Windows 8, Windows Phone) et Java !
Pour les statistiques d'utilisation, Application Insights propose un SDK qui permet d'intégrer des marqueurs dans un applicatif Web, une application Windows 8 ou une application Windows Phone. Ces marqueurs s'occupent d'analyser dans le détail l'utilisation de l'applicatif et de proposer des informations détaillées dans l'interface d'analyse centrale. Par exemple, pour une application mobile, les informations récoltées sont très nombreuses : nombre d'utilisateurs, détail des pages vues, chemin de navigation des utilisateurs, fidélisation des utilisateurs (combien de temps avant qu'un utilisateur revienne sur l'application), type, langue et version du téléphone utilisé, mode de connexion (Wifi, GSM, offline... car les traqueurs fonctionnent en offline et consolident à la découverte du réseau).

Azure : plus de souplesse pour tous

DevOps, c'est aussi la possibilité de disposer plus simplement et plus rapidement d'environnements de développement ou de tests pour reproduire la production, ou simplement pour développer.
Aujourd'hui, disposer rapidement d'un environnement de développement ou de test n'est pas une mince affaire. Sauf quelques exceptions, les délais IT pour commander et mettre en place un environnement de développement sont souvent conséquents. Cela est normal, le développement a besoin d'une machine éphémère, et l'IT a besoin de rationnaliser et maîtriser les coûts.
C'est là où le Cloud, apporte de la souplesse intéressante au développement et à l'opérationnel.
Besoin d'une machine virtuelle pour tester une fonctionnalité sur un serveur SharePoint ? Azure est capable de fournir ce genre de ressource en l'espace de quelques minutes voir secondes. Azure propose, de base, différentes images de machine : Biztalk, SQL Server, Ubuntu, CentOS, Oracle... et même des machines avec **Visual Studio 2014** (pour tester le nouveau Visual Studio sur sa tablette par exemple ?). Un gros avantage également est que Microsoft offre avec tout abonnement MSDN de 75€ (MSDN Premium) à 115€ (MSDN Ultimate) de crédit sur Azure. De quoi réaliser sans restriction plusieurs tests. Une charge en moins pour l'opérationnel et un bonheur pour les développeurs qui ont un environnement sans aucune connaissance IT et en un claquement de doigt.
Pour le développeur, l'utilisation du Cloud est de plus en plus simplifiée, avec une gestion des ressources Azure directement depuis Visual Studio, sans jamais se connecter sur le portail Azure et avec des connaissances très réduites sur Azure.
Par exemple, déployer un applicatif Web sur un serveur de développement IIS dans Azure peut s'effectuer en quelques clics depuis Visual Studio 2013/2014 grâce à la notion de Website Azure.
Azure n'est pas réservé qu'aux développeurs, l'IT y trouve aussi un grand avantage : besoin d'un environnement de test pour déployer en intégration continue, sans pour autant acheter une nouvelle machine ? Un administrateur système a la possibilité de provisionner à la demande, en quelques minutes, un nouvel environnement composé d'une ou plusieurs VM. Avec un peu de scripting, cet environnement peut même être configuré pour ne rester actif que dans les périodes de temps pour lesquelles l'équipe l'utilise, et être par exemple éteint automatiquement la nuit, le week-end et tout simplement quand le projet est terminé.
Enfin, besoin de reproduire temporairement un environnement de pré-production composé de nombreuses machines pour reproduire un incident de production? Azure et le Cloud permet de disposer de ressources illimitées de manière éphémère, sans doubler son investissement initial en termes de machines physiques.
L'IT Pro chargé de gérer un abonnement va donc faire le nécessaire pour simplifier la création et la reproduction d'environnements de développement, test, production, souvent en créant ses propres scripts. Les ressources informatiques pour faire tourner les applications devenant ainsi une « simple configuration ».
Un des piliers de DevOps étant l'infrastructure-as-code, il devient très intéressant de réaliser du scripting pour des tâches redondantes, ou vouées à être automatisées. Microsoft fournit un SDK PowerShell permettant de jouer avec toutes les fonctionnalités du service. Le SDK est disponible sur azure.microsoft.com, un petit d'exemple d'utilisation avec la création d'une machine virtuelle : Fig.3.
Simple, court, compréhensible facilement, c'est tout ce qu'on demande à un bon SDK. Grace à Azure, l'IT se retrouve facilement avec une charge en moins sur les épaules, et la puissance et la flexibilité d'Azure leur facilite grandement la gestion de leurs environnements.

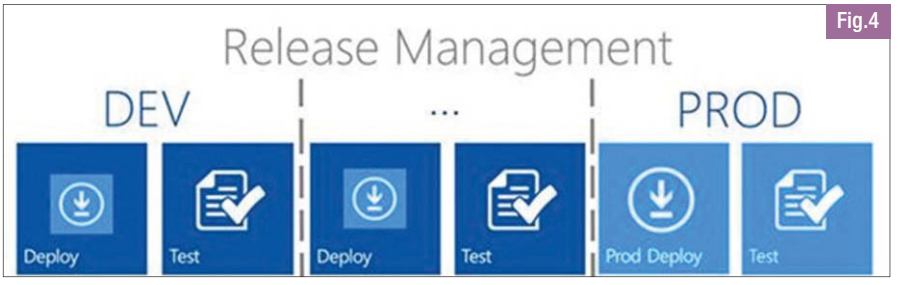
Release Management : l'orchestrateur de livraison

L'un des principaux buts du mouvement DevOps est l'industrialisation du processus de livraison pour permettre de gagner en productivité et en qualité. Auparavant, il n'existait pas de solution Microsoft pour répondre à ce besoin, désormais il y a Release Management ! Mais avant de parler du produit, qu'est-ce que Release Management ?
Release Management est la gestion des mises en production, il a pour but de planifier les déploiements d'application tant sur le plan matériel que logiciel. Release Management peut être quelque chose de vaste et complexe à mettre en place sans outil.
Heureusement des solutions existent et gagnent en maturité d'année en année.
Release Management n'est pas qu'un outil, il implique aussi une nouvelle façon de gérer la livraison de son projet informatique. Qui est

Fig.3

```
New-AzureVMConfig -Name $vmName -InstanceSize "Small" -Image $nomDeLimage
    | Add-AzureProvisioningConfig -Windows -AdminUserName $username -Password $password
    | New-AzureVM -ServiceName $svcName -Location "West US"
```

Fig.4



concerné par Release Management ? La réponse est : tout le monde.
Exemple avec un cas simple, mais pourtant très fréquent, composé d'un environnement de développement, un autre de Quality Assurance, et, enfin un dernier de production. Chacun de ces environnements possède ses propres contraintes de déploiement.
L'environnement de développement n'aura que peu de contraintes sur le système de backup mis en place, contrairement à l'environnement de production.
Par ailleurs, un système permettant de garder une disponibilité doit être mis en place lors de la migration sur la production.
Toutes ces étapes dans la livraison possèdent leurs contraintes et leur propre workflow de déploiement.
Elles possèdent aussi leurs propres critères de validation et leurs propres validateurs. Il y a donc beaucoup de contraintes et de variables pour assurer une bonne automatisation de la livraison.
Un orchestrateur devient donc vite indispensable pour fournir une livraison stable et surtout modulable et évolutive Fig.4.
Release Management est issu du produit InRelease, une solution développée par la société InCycle. La solution a été rachetée par Microsoft courant 2013 afin d'en faire son produit de référence pour la mise en œuvre des scénarios de livraison.
Disponible depuis l'année dernière, le nouvel outil Release Management for Visual Studio a probablement sa place dans votre usine logicielle et permettra à coup sûr d'améliorer la qualité de vos processus de déploiement avec simplicité et efficacité.
Release Management fonctionne avec Team Foundation Server 2010, 2012 et 2013; il permet de façon simple et entièrement personnalisable de modéliser et réaliser les processus de livraison, ainsi que d'effectuer des suivis d'approbation adaptés à chaque phase de livraison.
Ces processus sont semblables au Workflow standard, et sont donc faciles à prendre en main. Avec Release Management il est donc aisé d'adapter son processus de déploiement en fonction de l'environnement cible (backup, chaines de connexions, plateforme de déploiement etc.). Le but final étant d'automatiser au maximum le processus de livraison afin de gagner en productivité et en fiabilité.
La solution Release Management est décomposée en 3 modules distincts à savoir :

- Un agent de déploiement : Release Management Agent,
- Un client WPF pour modéliser les processus de livraison via un système très proche de la réalisation de Workflow. Et une interface Web pour interagir avec la requête d'approbation mise en place : Release Management Client,
- Le point central de Release Management, contenant la base de données SQL exposant différents services utilisés par les autres composants : Release Management Server.

Chacun des modules étant indépendant, la nouvelle solution de Microsoft se veut flexible.

À qui s'adresse cet outil ?

La quasi-totalité des applications sont concernées par les besoins auxquels répond Release Management. Toute application a besoin de tests, d'être validée par plusieurs intervenants, des intervenants avec différents profils, le tout bien orchestré et suivi pour éviter les erreurs humaines et avoir du reporting.
Release Management est aujourd'hui une valeur sûre pour grandement améliorer la collaboration et la fluidité du déploiement des applications.

Conclusion : Dans la pratique, est-ce que DevOps va révolutionner ma vie ?

Après l'Agile, DevOps. Derrière ce mouvement, toujours le même objectif : gagner en collaboration et en communication pour fournir toujours plus de qualité. Le mouvement DevOps est encore « jeune », mais a déjà ces principes bien ancrés.
Joignant processus et outillage, DevOps tend à devenir un standard dans la réalisation des projets informatiques.
Néanmoins, tout comme l'Agile, l'outillage ne fait pas tout; il est même inutile sans un changement de mentalité et de façon de travailler.
Aujourd'hui les outils sont à disposition : complets, simples et abordables. Le DevOps est à portée de main ! Lancez-vous !

Pré-requis pour réussir la mise en oeuvre d'un processus de *'Continuous Delivery'*

Cyrille Le Clerc,
Product Manager, **CloudBees**

Actuellement, les entreprises se montrent de plus en plus séduites par la méthodologie de *continuous delivery* (livraison continue) qui permet de disposer de manière immédiate de logiciels de qualité. Le *continuous delivery*, dont la particularité est de maintenir constamment le logiciel prêt à la mise en production, peut être considéré comme une évolution naturelle des pratiques d'intégration continue et de développement logiciel agile ; cependant, les enjeux culturels et opérationnels liés à la mise en œuvre du *continuous delivery* sont infiniment plus importants. Aux yeux de la plupart des entreprises, le *continuous delivery* ne nécessite que l'adaptation et l'extension des processus de développement logiciel existants. Or, les rôles, les relations et les responsabilités des individus de l'entreprise peuvent être impactés. Les outils utilisés pour livrer, mettre à jour et maintenir les logiciels doivent être capables de supporter parfaitement l'automatisation et la collaboration, tout en réduisant au minimum les délais et assurant la diffusion de cycles de feedback rigoureux au sein de l'entreprise. Même si de tels changements peuvent constituer un enjeu de taille pour les entreprises soumises à des contraintes règlementaires et opérationnelles, il existe aujourd'hui un certain nombre de bonnes pratiques permettant de progresser dans des conditions optimales.
Voici donc 7 prérequis que les entreprises devraient prendre en compte pour réussir leur transition vers le *continuous delivery*.

1 S'assurer que les équipes de développement, d'assurance qualité et de production communiquent entre elles et ont défini des objectifs communs

Alors que l'intégration continue ne concerne que les développeurs, le *continuous delivery*

adresse les phases de test réalisées par les responsables d'assurance qualité et les déploiements vers les environnements de préproduction et de production gérés par les responsables opérationnels de production. Ceci constitue une évolution majeure du développement logiciel et, pour réussir la transformation d'une plate-forme d'intégration continue en plate-forme de continuous delivery, il est essentiel d'associer à sa gouvernance les équipes d'assurance qualité et les équipes de production autant que les développeurs. La collaboration et la communication sont aujourd'hui des composantes essentielles d'un développement logiciel réussi et dans un environnement de continuous delivery, elles jouent un rôle central.

2 Implémenter l'intégration continue avant d'entamer la phase de *continuous delivery*

Le *continuous delivery* est une extension de l'intégration continue. Ainsi un prérequis pour aborder le *continuous delivery* consiste à s'assurer que l'intégration continue est déjà en place et opérationnelle, avec la gestion du contrôle de source, les builds automatisés et les tests unitaires du logiciel.

3 Automatiser tout ce qui peut l'être

Le *continuous delivery* implique la répétition automatique de nombreuses tâches telles que la création d'applications et de packages, le déploiement des applications et des configurations, la réinitialisation des environnements et des bases de données. Avec le *continuous delivery*, toutes ces tâches devraient être automatisées avec des outils et des scripts et gérées sous gestion de version permettant que chaque étape puisse être auditée et reproduite.

4 Faire partager les outils et les procédures par toutes les équipes

Le *continuous delivery* vise à valider les procédures et l'automatisation du déploiement utilisées dans l'environnement de production. Pour atteindre cet objectif, ces procédures et ces automatisations doivent être utilisées le plus tôt possible, de manière à ce qu'elles aient été testées à fond lorsqu'elles seront utilisées pour déployer le logiciel en production. La plupart du temps, les mêmes outils peuvent être utilisés dans tous les environnements, c'est-à-dire l'intégration, la préproduction et la production. Les scripts d'automatisation devraient être gérés dans des référentiels de code source partagés, de manière à ce que chaque équipe – développement, assurance qualité et production – puisse améliorer les outils et les procédures. Les mécanismes tels que les *Pull Requests* (c. f. GitHub) peuvent aider à la gouvernance de ces outils et scripts partagés.

5 Rendre les applications « production friendly » pour les équipes d'exploitation : pour des déploiements sans stress

Les procédures de déploiement et de *rollback* des applications devraient être simplifiées de manière à ce que le déploiement en production devienne un non-événement. Une étape-clé pour y parvenir est de réduire le nombre de composants et de paramètres de configuration déployés. La facilité des *rollbacks* est essentielle pour que les équipes soient plus audacieuses dans leur déploiement de nouvelles fonctionnalités ; en cas de problème, il suffit de revenir en arrière. Les « *feature toggles* » — activation de fonctionnalités par configuration — aident à découpler le déploiement de binaires de l'activation des fonctionnalités : un *rollback* peut alors simplement être réalisé par la désactivation d'une fonctionnalité, grâce à un paramètre de configuration. Il est important de bien maîtriser les évolutions des schémas de bases de données, car ceci peut rendre les déploiements et les *rollbacks* beaucoup plus complexes. Le modèle de design *schema-less* des bases de données NoSQL offre une grande flexibilité en transférant la responsabilité du schéma de la base de données vers le code. Ce concept peut aussi être appliqué aux bases de données relationnelles traditionnelles.

6 Rendre l'infrastructure « project friendly » : responsabiliser les individus et les équipes

Les infrastructures devraient fournir tout l'outillage (les GUIs — interfaces graphiques), les APIs (interfaces de programmation d'applications) et les SDK (kits de développement logiciel) ainsi que la documentation nécessaire pour responsabiliser les équipes de développement et d'assurance qualité, et leur permettre ainsi de travailler en toute autonomie. Parmi ces tâches figurent :

- Le déploiement de la version de l'application de leur choix dans un environnement Donné,
- La gestion des paramètres de configuration (lecture, modification, export, import),
- La gestion des bases de données (création et restauration de *snapshots* des données),
- La possibilité de lire, rechercher et créer des alertes sur les logs des applications.

Les plateformes de Cloud public, et en particulier les plates-formes IaaS et PaaS sont des exemples de plates-formes « *project friendly* ».

7 Garantir que les versions d'applications sont prêtes à entrer en production

L'un des principaux objectifs du *continuous delivery* est de permettre au *product owner* de décider de déployer en production toute version de l'application qui a passé avec succès le processus de continuous delivery — et pas uniquement les versions livrées à la fin de chaque itération, identifiée par un numéro de version « lisible par les humains ». Atteindre cet objectif nécessite un certain nombre de changements dans la manière dont l'application est conçue :

- Les fonctionnalités n'ayant pas encore été validées par les équipes d'assurance qualité ne devraient pas être montrées aux utilisateurs finaux. Les patterns de *feature toggles* et de *feature branches* sont deux moyens de le faire.
- Les outils de build devraient évoluer pour passer du concept de versions sémantiques (e.g. 1.3.6) séparées par des versions « *snapshot* » intermédiaires non identifiées (e.g. 1.3.7-SNAPSHOT) à un concept flot de versions non sémantiques, mais chacune uniquement identifiée pour permettre de déployer en production n'importe laquelle de ces versions. Subversion aide à mettre en œuvre ce concept grâce à ses numéros de versions ordonnés. Git rend la tâche plus complexe à cause de ses « *hashs* » de commit non ordonnés ; un outillage peut aider à rendre ces identifiants de version plus « lisibles » pour les humains.

Conclusion

De tout ce qui précède, on comprend que *continuous delivery* est bien plus qu'un simple ensemble d'outils — en fait, c'est un processus qui a un impact sur les individus et sur la culture de l'entreprise.

La technologie, les individus et les processus doivent s'accorder pour que la mise en œuvre du *continuous delivery* réussisse dans une entreprise, une approche collaborative étant l'une des conditions essentielles du succès. Ces sept prérequis constituent de bonnes pratiques qui vous conduiront avec succès sur la voie du *continous delivery.*

Offre Spéciale Fêtes

1 an de Programmez! (11 n°)

+ Clé USB Programmez!

(tous les n° depuis le n°100)

+ Accès aux archives

+ Les anciens numéros papier disponibles

TOUT **PRO**GRAMMEZ! le magazine du développeur pour

~~112,90€~~ *

79€ **

08-22-13 © esenkartal

* *prix selon les stocks des numéros disponibles*
** *tarif valable uniquement pour la France métropolitaine*

Offre spéciale valable jusqu'au 31 janvier 2015.

Toutes nos offres sur www.programmez.com

Oui, je profite de l'offre Spéciale Fêtes

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

☐ *Abonnement 1 an au magazine + Clé USB + Accès aux archives + Les anciens numéros papier disponibles*

☐ M. ☐ Mme ☐ Mlle Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

Tél : *(Attention, e-mail indispensable pour les archives sur internet)*

E-mail : @

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

☐ Je souhaite régler à réception de facture

* *Tarifs France métropolitaine*

PROG 180 *Offre limitée, valable jusqu'au 31 janvier 2015*

Epilogue : entre syndrome “push to prod” et DevOps m’a tuer

Le DevOps peut être vu comme une ultime évolution des méthodes agiles, de la gestion du cycle de vie et en cherchant à rapprocher le développeur, de l’exécution, du développement continu, du déploiement continu et de la fin de la « version ». Mais comme vous l’avez lu dans ce dossier, pour atteindre le Graal du DevOps, il faut l’ADN. Mais à tous les niveaux, que vous soyez seul ou en équipe, il est possible de mettre en œuvre du DevOps, aussi bien dans les méthodes que les outils. Mais attention aussi à ne pas tomber dans le DevOps marketing et de mettre du DevOps sur tout et n’importe quoi.

Utilisez-vous des pratiques DevOps ?

Chaque jour ou presque : 28 %
Uniquement quelques outils : 10 %
C’est quoi ? : 62 %
(sondage express Programmez ! / novembre 2014, 100 votes)

Si le terme DevOps est partout, la preuve en 1 an, nous avons publié deux dossiers et un livre blanc sur ce sujet, ce n’est pas pour autant que le développeur le comprend ou l’utilise. Vous êtes 62 % à vous demander ce que c’est et ce qu’il y a derrière ce mot. Mais comme DevOps a une approche très modulaire, vous pouvez faire du DevOps (tout du moins une approche à la DevOps) sans le savoir réellement. Il y a un décalage important entre les entreprises et les équipes techniques et de production qui l’utilisent et les autres (PME, indépendants, étudiants, etc.).

Et si on dépassait le syndrome « push to prod »

En tant que développeur, vous avez sans doute déjà vécu cette peur quand il s’agit de déployer, de passer le projet que vous avez codé, en production... Ah le « push to prod ». DevOps, l’intégration continue, le développement continu, le déploiement continu doivent justement éviter ce « syndrome » et ne plus forcément envoyer en déploiement / production le projet d’un coup. Mais au contraire, être sur une approche au « fil de l’eau », bref en mode continu. Fini les frontières étanches entre le développement, le déploiement, la production. Fini (ou presque) la notion de « release ». Pour arriver à cette extrémité, la notion de DevOps, de développement continu doit être poussé à l’extrême comme on peut l’avoir dans des projets cloud, Netflix, Facebook, Twitter, etc. Ce sont des centaines de « releases » de code qui sont faites chaque année...
DevOps peut donc être un moyen d’éviter le syndrome « push to prod » (d’autres méthodes le permettent aussi, comme dit plus haut).

DevOps va-t-il tuer le développeur ?

Le développeur doit rester développeur. Donc, il doit continuer à coder, à faire ce qu’il sait faire. Il peut intégrer d’autres compétences et profils en dehors de la technique mais ces éléments ne doivent pas supplanter le code. Par contre, oui, connaître les processus de déploiements, de productions sont des « plus » et être ainsi plus réactif et comprendre ce qu’il se passe après le codage.
Jeff Knupp, développeur python, a publié un post très intéressant sur son blog (http://goo.gl/fuVWV9). DevOps est l’occasion de rapprocher développeur, les responsables des opérations (déploiement, production notamment), et les responsables qualités (cellule qualité, testeurs). L’accélération de la publication des logiciels et des différentes versions bouleverse le modèle traditionnel de développement (que ce soit le développement en V ou en agile). Le développeur, à cause du rythme toujours plus élevé et des agendas de plus en plus serrés, assume de plus en plus de rôles (développeur, qualité logicielle, responsable de release / version...). Jeff rapproche cette frénésie de charger la barque des responsabilités du développeur à la notion de développeur « full-stack » qui fait tout ! Les géants du web, les startups ont favorisé l’apparition de ce développeur full-stack. Mais le contexte n’est pas le même. Dans une startup, les ressources sont souvent limitées. Mais faut-il faire de même dans les grandes équipes ? Pas sûr. Jeff conclut ainsi : **laissez les développeurs écrire du code !**
Que l’on soit d’accord ou non, notamment sur les arguments avancés, avec Jeff, cette réflexion est importante. Il faut pouvoir trouver un juste milieu. Mais il ne faut pas croire que DevOps va tout résoudre et qu’il est adapté à tout. Le risque (réel) est que les entreprises, les SSII, etc. ne jurent que par DevOps et l’imposent partout, pour tout, au risque de surcharger le développeur qui passera de moins en moins de temps à coder.
DevOps ne doit pas être une obsession ni un objectif absolu. Il peut cependant vous aider à mieux structurer les développements, le cycle de vie des applications, à améliorer les pratiques en équipe.

« DevOps a besoin de devenir une bonne pratique dans votre entreprise si vous voulez réussir dans la nouvelle économie orientée application »

Voilà une phrase choc du rapport « How to survive and thrive in the application economy », publié en septembre 2014 par l’éditeur CA. Selon ce rapport, DevOps accélère la disponibilité (des applications), améliore la qualité des applications. Selon cette étude, les entreprises françaises sont en retard dans l’économie des applications. Cette étude évoque un chiffre étonnant « 95 % des entreprises françaises planifient actuellement la mise en application de l’approche DevOps afin d’accélérer la livraison d’applications ». Nuançons ce résultat car l’étude porte sur 13 pays dont la France et 1425 personnes interrogées (responsables informatiques et métiers). D’autre part, nous ne croyons pas que 95 % des entreprises savent ce qu’est DevOps et nombre d’entre elles ne savent même pas que cela existe. Peut-on dire que DevOps est à la mode comme l’eXtreme Programming ou Scrumm l’était il y a quelques années ?
Lisez cet article très pertinent chez OVH et comment DevOps est utilisé au quotidien :
https://www.ovh.com/fr/a1075.devops-profil-pointu-poste-ovh
Quelques éléments clés de cet article :
DevOps est à cheval entre ces deux mondes (développeurs, sysadmins)
Un devops est capable de développer et d’administrer. C’est un développeur opérationnel.

Les Frameworks Front-End Web

2ere partie

Hugo Carnicelli
Consultant chez SoftFluent

Utiliser un Framework Front End lors d'un développement web permet de gagner un temps considérable sur la mise en place d'une interface utilisateur. Une foule de composants réutilisables sont la plupart du temps déjà disponibles.

Ink présenté comme « l'OutSider »

Le Template de base est un peu plus complexe que les autres concurrents. On remarque deux feuilles de style et quatre fichiers JavaScript. On notera la présence de Font Awsome, l'une des plus exhaustives des polices d'icônes. De plus, on retrouve une « div » de classe « ink-grid » qui fait office de conteneur de grille responsive :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial
-scale=1.0, maximum-scale=1.0, user-scalable=0">
  <title>Horloge m&eacute;t&eacute;o</title>
  <link rel="stylesheet" type="text/css" href="css/ink-flex.
min.css">
  <link rel="stylesheet" type="text/css" href="css/font-awesome.
min.css">
</head>
<body>
  <div class="ink-grid ink-form">
    <!-- SUITE DU CONTENU A INSERER ICI -->
  </div>
  <script type="text/javascript" src="js/modernizr.js"></script>
  <script type="text/javascript" src="js/holder.js"></script>
  <script type="text/javascript" src="js/ink-all.min.js"></script>
  <script type="text/javascript" src="js/autoload.js"></script>
</body>
</html>
```

Dès la réalisation du bloc de recherche, on s'aperçoit d'un défaut majeur : il n'est pas prévu d'ajouter des marges afin de centrer le contenu de la grille comme c'est le cas pour Bootstrap et Foundation. La seule parade possible est de placer deux colonnes de 10% via la classe « all-10 », pour la marge gauche, et 80% via la classe « all-80 », pour le contenu, afin de réaliser une marge de chaque côté. La classe faisant office de ligne est cette fois « column-group » :

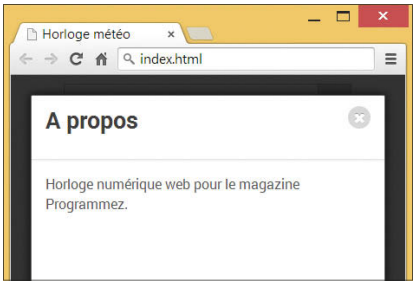
```
<div class="column-group first">
  <div class="all-10"> </div>
  <div class="all-80">
    <div class="control-group">
      <div class="control all-100 append-button">
        <span>
          <input type="text" id="ville" value="Paris"></span>
        <button class="ink-button"><i class="fa fa-search"></i>
</button>
      </div>
    </div>
  </div>
</div>
```

Pour l'horloge là encore, il faut ajouter une colonne vide afin de réaliser une marge. Pour le reste, l'implémentation est sans surprise :

```
<div class="column-group middle">
  <div class="all-10"> </div>
  <div class="all-80 clock">
    <div class="column-group clock-inner">
      <div class="all-50 degree">
        12&deg;
      </div>
      <div class="all-50 align-right">
        <i class="fa fa-cloud weather"></i>
      </div>
    </div>
    <div class="column-group clock-inner">
      <div class="all-50">
        10/01/2015
      </div>
      <div class="all-50 align-right">
        10:42
      </div>
    </div>
  </div>
</div>
```

Le bouton « A propos », représenté sous la forme d'un « a » orné d'une classe « ink-button », est aligné à droite grâce à la classe « push-right ». Quant à la pop-up, elle n'est ni la plus complexe ni la plus simple des trois. Elle offre visiblement tout autant de possibilités que ses concurrentes. Point intéressant, la mise en page de celle-ci se situe entre Bootstrap et Foundation du point de vue de l'overlay. Elle s'affiche pratiquement pleine page, laissant entrevoir l'overlay, et ce tout autant du point de vue classique que mobile :

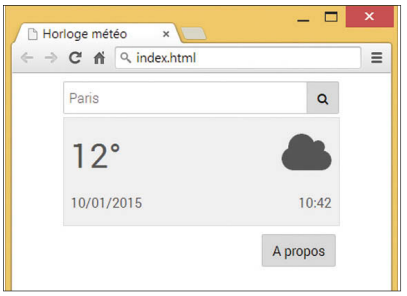
```
<div class="column-group first">
  <div class="all-10"> </div>
  <div class="all-80">
    <a href="#" id="aproposTrigger" class="ink-button push-right">
A propos</a>
  </div>
</div>
<div class="ink-shade fade">
  <div id="apropos" class="ink-modal fade" data-trigger="#apropos
Trigger">
    <div class="modal-header">
      <button class="modal-close ink-dismiss"></button>
      <h2>A propos</h2>
    </div>
    <div class="modal-body" id="modalContent">
      Horloge num&eacute;rique web pour le magazine Programmez.
    </div>
  </div>
</div>
```



En revanche, un clic sur l'overlay ne permet pas de fermer la pop-up. Cela peut s'avérer relativement perturbant puisque la plupart des plugins équivalents proposent cette fonctionnalité. Point noir cependant, il n'existe pas de classe CSS permettant de mettre en valeur un bloc comme cela était le cas pour les deux autres. Nous allons donc ajouter aux classes habituelles deux nouvelles classes afin de s'y substituer :

```
<style type="text/css">
  .first {
    margin-top: 10px;
  }
  .middle {
    margin-top: 20px;
  }
  .clock {
    border: 1px solid #DDD;
    background: #f0f0f0;
  }
  .clock-inner {
    margin: 16px 8px;
  }
  .degree {
    font-size: 40px;
  }
  .weather {
    font-size: 56px;
  }
</style>
```

Le résultat final au format mobile ressemble à cela :



Conclusion

Voici un tableau visant à comparer de manière littérale différents aspects des Framework exposés ci-contre. Bootstrap a été très utilisé dernièrement, trop si l'on peut dire. Selon le site trends.builtwith.com, environ 1 site sur 30 utilise Bootstrap. Il est devenu aisé de reconnaitre ce Framework comme brique graphique au sein d'un site web. Ce défaut est généralement un point négatif dans la mise en œuvre d'une identité graphique. De plus, il n'est pas le meilleur concernant la plupart des critères cités ci-dessus. Restant cependant une valeur sure, je peux recommander sans réserve l'utilisation de ce Framework pour des projets de type Back Office. Il comporte un ensemble de composants clé en main répondant à la plupart des besoins lors de la conception d'une interface d'administration. Foundation se veut plus simple d'utilisation et est conçu afin d'intégrer une charte graphique dès les premières phases de conception. On notera cette phrase, issue de la FAQ reflétant parfaitement la ligne éditoriale se distinguant de Bootstrap : Nous avons intentionnellement laissé nos styles spartiates. Nous ne voulions pas nous retrouver dans un univers où tous les sites ressemblent à Foundation – c'est pour cela que nous avons mis de côté beaucoup de styles complexes pour que vous puissiez ajouter les vôtres facilement sans avoir à réécrire un ensemble de styles inclus. De plus, Foundation compte quelques références intéressantes à son actif comme le Store officiel HTC ou encore le site Pixar Projection. Je le recommande pour des projets de type Front Office sans hésiter. Il nécessitera en revanche un travail plus important lors de sa mise en place. Le petit dernier, Ink, est plus complet que ses concurrents sur plusieurs aspects importants tels que la compatibilité navigateurs. Cela peut s'avérer être un choix déterminant. Malgré cela, il manque peut-être encore de maturité et de références pour pouvoir être recommandé au même titre que les deux autres. Il peut toutefois être un choix intéressant en cas d'utilisation de Node.js ou pour un projet qui nécessite des composants JavaScript plus poussés et plus nombreux. Il reste un Framework très prometteur à surveiller de près. Pour aller plus loin, n'hésitez pas à consulter les documentations des différents Framework :

http://getbootstrap.com/
http://foundation.zurb.com
http://ink.sapo.pt/

Vous trouverez également des tableaux de compatibilité avec les différents navigateurs :

http://getbootstrap.com/getting-started/#support
http://foundation.zurb.com/docs/compatibility.html
http://ink.sapo.pt/ui-elements/

	Bootstrap	Foundation	Ink
Poids gzip	49.3 KB + 32.9 KB*	48.6 KB + 29.6 KB* + 35.6 KB**	412 KB
Affichage	330 ms	290 ms	490 ms
Police d'icône	200 icones	283 icônes externes	369 icones
Grille	Complète	Complète, ne nécessite pas de conteneur	Sans possibilité d'offset
Popup	Largeur limitée	Pleine page au format mobile	Overlay non réactif (voir partie B étape 6)
Composants JS	12	18	22
Pré-processseur CSS	LESS / SASS	SASS	SASS
Compatibilité navigateurs	8+ ordinateur + mobile ordinateur ordinateur + mobile ordinateur	9+ ordinateur + mobile ordinateur + mobile ordinateur + mobile ordinateur + mobile	8+ ordinateur + mobile ordinateur + mobile ordinateur + mobile ordinateur + mobile

*jQuery 1.11 et 2.1 **Zurb Icon Fonts 3

Connectez vos objets Arduino, Spark.IO, .NET Microframework, Galileo à Microsoft Azure 2e partie

L'Internet des objets (Internet of Things, ou IoT) est en plein boom. Entre les coûts des processeurs et les coûts des composants électroniques vraiment très faibles, plus une faible consommation électrique, il est devenu possible de créer de nombreux objets plus ou moins intelligents. Ajoutez à cela des connexions devenues possibles grâce à du Bluetooth Low Energy BLE, des composants Wifi ou GPRS à prix réduits, ces nouveaux objets peuvent se connecter à Internet ou entre eux.

Laurent **Ellerbach**
Audience Marketing - Microsoft EMEA - DX

Cloud et infrastructure

Pour le stockage des données, je me suis naturellement tourné vers Azure. De nombreux éléments penchent en sa faveur : gratuit pour le stockage des données jusqu'à 20Mo, gratuit pour la publication de données à travers les Mobile Services, 10 sites web gratuits que ce soit en APS.NET ou PHP ou autre framework pour une consultation des données postées. Cela répond donc parfaitement à mon besoin de hobbyiste. Azure possède pour des besoins plus avancés également tout ce qui est nécessaire pour une infrastructure IoT comme de l'analyse et prédiction de données avec le Machine Learning, un service spécialement dédié pour les volumes de données intensifs par de nombreux objets. Le tout avec un coût très avantageux. Azure est interopérable, supporte la plupart des technologies telles que Linux, Windows, ASP.NET, PHP, Node.JS, Java et bien d'autres. A noter que même pour l'accès aux services gratuits de Azure, il est nécessaire de rentrer lors de l'inscription une carte bancaire. Cette carte n'est utilisée que pour la vérification de l'identité. Si vous voulez accéder à des services payants, elle sera alors utilisée.

Une fois enregistré, le paramétrage des Azure Mobile Services se fait assez simplement.

Etape 1 : créer un Mobile Service Fig.6.

Etape 2 : créer une table Fig.7.

L'intérêt de ces mobiles services est leur grande flexibilité. Pas besoin de créer la structure de la base de données, cela se fait automatiquement. Tous les accès à Azure se font sur des protocoles standards ouverts. Ajouter un enregistrement se fait également très simplement en HTTP ou HTTPS avec des API REST.

Pour accéder à Azure, il est donc nécessaire d'avoir une pile IP et des sockets dans le framework de son objet. Voici le code nécessaire pour écrire dans mon service Azure depuis mon Arduino ou équivalent :

```
TCPClient client;
byte AzureServer[] = { 12, 34, 56, 78 };
```

```
String writeJsonWind(struct wind wd) {
  // Création d'un JSON simple;
  String datastring = "{\"sensorID\":";
  datastring += String(wd.ID);
  datastring += ",\"channel\":";
  datastring += String(wd.channel);
  datastring += ",\"instSpeed\":";
  datastring += String(wd.instantSpeed);
  datastring += ",\"averSpeed\":";
  datastring += String(wd.averageSpeed);
  datastring += ",\"direction\":";
  datastring += String(wd.direction);
  datastring += ",\"batterylife\":";
  datastring += String(wd.bat);
  datastring += "}";
  return (datastring);
}

void sendData(String thisData) {
  // Crée un connexion sur le port 80
  // L'adresse IP est celle du service Mobile Services
  if (client.connect(AzureServer, 80))
  {
    //Serial.println("Connected to Azure Server");
    // Créer la requête REST en POST
    // Nomdelatable représente le nom de votre table
    client.print("POST /tables/nomdelatable/");
```

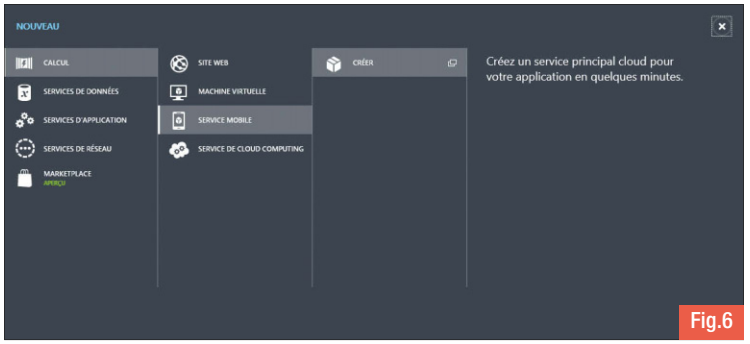


Fig.6

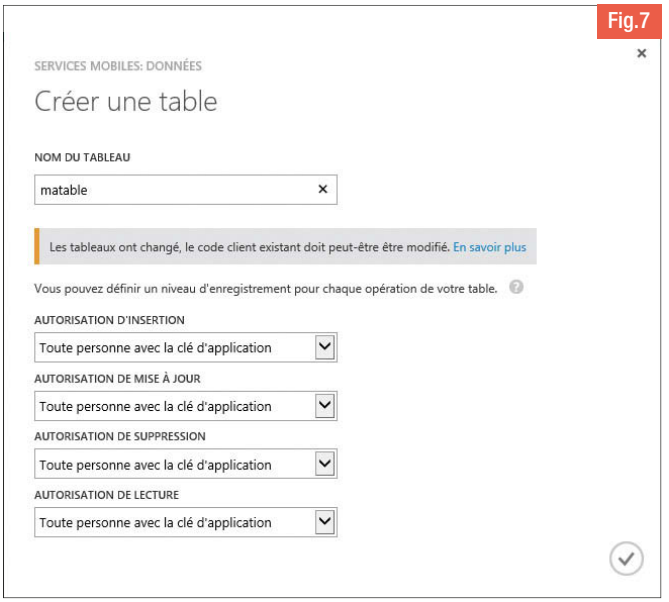


Fig.7

```
    client.println(" HTTP/1.1");
    // la clé est la clé de l'application
    client.println("X-ZUMO-APPLICATION: 123456789abcdef12345
6789abcdef12");
    // le host est le nom du service Azure
    client.println("Host: nomdumobileservice.azure-mobile.net");
    client.print("Content-Length: ");
    client.println(thisData.length());
    client.println("Connection: close");
    client.println();
    // et les données
    client.println(thisData);
  }
  else {  // En cas d'erreur clôt la connexion:
    client.stop();
  }  }

// L'envoi des données se fait ensuite simplement
// en créant le JSON et en postant les données
String dataString = writeJsonWind(myWind);
sendData(dataString);
```

Une fois un socket ouvert sur un port (ici 80), il est possible de communiquer, ici en HTTP. Le protocole http est très simple, il consiste en une entête au format texte suivi d'un contenu au format texte (ou compressé). HTTPS est équivalent mais avec un cryptage en plus. Dans mon cas, ce qui est envoyé va donc ressembler à cela :

```
POST /tables/nomdelatable/ HTTP/1.1
X-ZUMO-APPLICATION: 123456789abcdef123456789abcdef12
Host: nomduservice.azure-mobile.net
Content-Length: 88
Connection: close
{"sensorID":22,  "channel":5,  "instSpeed":12,"averSpeed":5,
"direction":2,"batterylife":90}
```

Ce qui est envoyé dans ce cas est un JSON qui a l'avantage d'être moins bavard que du XML tout en restant lisible par un humain. La création de la chaîne est assez simple à faire à l'aide d'un string dans le framework Arduino. Il est un peu consommateur en ressources mais simplifie la vie. La même chose peut être faite avec un simple tableau de « char » bien entendu. Ce qui est renvoyé ressemble à cela :

```
HTTP/1.1 201 Created
Cache-Control: no-cache
Content-Length: 133
Content-Type: application/json
Location: https://nomduservice.azure-mobile.net/tables/weather
//931CFDDE-AB7F-4480-BA28-F1D5C611398B
Server: Microsoft-IIS/8.0
x-zumo-version: Zumo.master.0.1.6.3803.Runtime
X-Powered-By: ASP.NET
Set-Cookie: ARRAffinity=da4a9f7437a690e3c1a799d3a6c3ddf3ee0c
bb9f5a67008d3c919f0149f34ee3;Path=/;Domain= nomduservice.azure
-mobile.net
Date: Sun, 31 Aug 2014 15:40:12 GMT
Connection: close

{"sensorID":22,"channel":5,"instSpeed":12,"averSpeed":5,"direction"
:2,"batterylife":90,"id":"931CFDDE-AB7F-4480-BA28-F1D5C611398B"}
```

Après l'entête assez longue mais contenant toutes les informations nécessaires sur la transaction, les données sont renvoyées en JSON avec l'ID de l'enregistrement. Dans notre cas, nous avons juste besoin de savoir si l'enregistrement a bien été effectué et n'avons pas besoin de stocker l'ID. A noter qu'il est nécessaire de lire l'ensemble des informations retournées pour que la transaction côté Mobile Services soit validée.
Il est possible d'optimiser ce que l'on envoie en modifiant le comportement du Mobile Service Azure. En effet, comme http est au format texte, il est assez bavard et utilise de la bande passante. En même temps, les ressources des processeurs embarqués sont limitées et il est très coûteux d'activer de la compression de données. Il faut donc essayer d'envoyer un minimum de données et d'en recevoir également un minimum.
Une façon simple dans un premier temps est de modifier la fonction insert du Mobile Service. Au lieu de renvoyer l'ensemble des données, je vais me contenter de renvoyer uniquement un OK (code 200) ou mauvaise requêtes (code 400). Voici le code en javascript :

```
function insert(item, user, request) {
request.execute({
success: function(results) {
request.respond(statusCodes.OK);  },
error: function(results) {
request.respond(statusCodes.BAD_REQUEST); }  }); }
```

A noter qu'il est également possible d'écrire ses propres API REST soit en Javascript soit en .NET. Cela permettrait par exemple d'envoyer les données avec des noms plus courts de colonnes, d'effectuer le mapping sur des noms plus long et lisibles dans la base, d'envoyer plusieurs enregistrement en même temps, effectuer des validations. La fonction insert de base ne supporte l'ajout que d'un seul enregistrement. Nous le verrons plus loin pour la boucle de retour des données
Attention, dans notre cas, la sécurité est ici vraiment minimale. Le protocole utilisé est HTTP, tout passe en clair, pas d'authentification même simple. La seule chose qui est ici utilisée est la clé de l'application Azure. Dans un projet plus complexe, il est impératif de pouvoir y ajouter un peu plus de sécurité par l'utilisation de HTTPS si le board le permet, de l'authentification plus forte ou de crypter un minimum les données envoyées si elles sont sensibles.
Nous voici à mi-chemin dans notre projet d'IoT. Nous avons donc vu comment connecter des objets qui ne le sont pas, ici mes sondes Oregon Scientific, créer des objets de toute pièce et les connecter au Cloud. Comme vous avez pu le constater, à partir du moment où vous avez accès à des sockets, il est possible de poster n'importe quel type d'information dans Azure. La partie la plus compliquée est de pouvoir connecter des objets. Cela devient encore plus compliqué quand il s'agit de connecter des objets qui communiquent déjà entre eux avec leur propre protocole

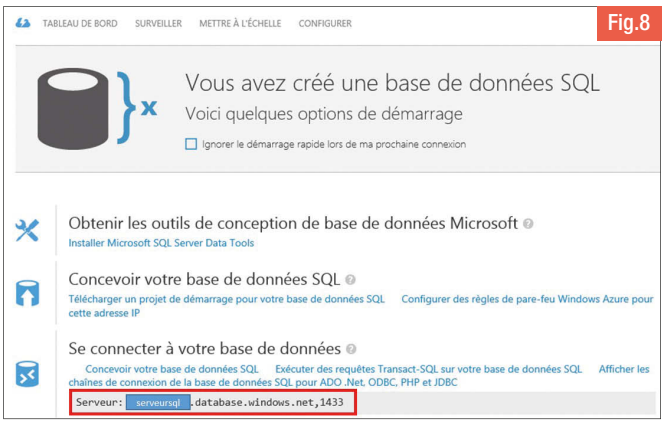


Fig.8

propriétaire ou qui ne communiquent pas encore. Dans mon exemple, il m'a été assez facile de décrypter le protocole utilisé par les sondes car il n'est pas crypté et assez simple. C'est souvent loin d'être le cas.

Rappels

Dans le précédent numéro, nous avons vu comment créer des objets et les connecter à Microsoft Azure. Le projet que j'utilise permet de récupérer des informations de température et humidité de sondes Oregon Scientific que je possède déjà. J'ai en plus créé à base d'ATMEL ATmega328 mes propres capteurs notamment un anémomètre, un capteur d'humidité du sol, d'ensoleillement. Ce projet d'Internet des objets me permet de couvrir l'ensemble des étapes d'un projet IoT.

J'ai donc déjà réalisé la publication des données de mes objets et leur stockage. Nous allons voir comment les consommer et les analyser. Enfin, nous terminerons par le retour d'information pour impacter les objets et leur environnement. Dans mon exemple, ce sera mon arrosage automatique. Mon arrosage automatique est lui-même un objet connecté. Il est basé sur une architecture de processeur ATMEL AT91SAM7X512 et fait tourner .NET Microframework (NETMF). NETMF est une implémentation de .NET qui tourne directement sur des chipsets sans OS en dessous. Ma carte possède une carte réseau qui lui permet donc d'être reliée au reste du monde. Et c'est elle qui ira chercher les informations d'humidité pour décider si elle doit arroser ou non.

Analyse des données

Mes capteurs vont remplir au fur et à mesure ma base de données et une base SQL Azure. Il s'agit d'une vraie base SQL et il est donc possible de consommer les données comme on le fait habituellement. Une façon très simple pour commencer et vérifier que tout se passe bien est de les consommer avec Excel. Pour cela il faut suivre les étapes suivantes :

- Dans le portail Azure, récupérer le nom de la base de données, le port d'accès, le nom d'utilisateur de la base de données et le mot de passe (accessibles dans les paramètres de la base de donnée) Fig.8.
- Penser à ajouter l'IP du PC sur lequel vous vous connectez dans la liste des IP acceptées Fig.9.
- Dans Excel, ajouter la connexion à une base de données SQL Server Fig.10.

Attention, de bien sélectionner dans l'assistant « utiliser le nom d'utilisateur et le mot de passe suivants » et de rentrer ceux de la base de données. Après avoir enlevé les colonnes qui ne sont pas utiles, créer des colonnes calculées pour faire des filtres facilement, créer des graphiques, on peut obtenir quelque chose qui ressemble à ça : Fig.11.

L'intérêt d'Excel est de pouvoir rapidement et simplement visualiser des données et prototyper des graphiques, ajouter des validations de données pour tester des modèles. L'exemple est celui de l'anémomètre et de la direction du vent. Nous avons donc vu dans cette étape comment poster des informations dans Azure, à partir du moment où notre objet a une pile IP et des sokets. L'étape suivante est de pouvoir exposer les données pour pouvoir les analyser. Dans mon cas, ce sera à travers un site Web également hébergé dans Azure à travers les Azure Web Sites. Le choix offert pour le site que ce soit en PHP, Java ou ASP.NET est très large. Je vais donc plutôt opter pour un site en ASP.NET. Je vais réaliser mon site en utilisant Modele View Controller (MVC), en utilisant Entity Framework pour l'accès aux données. L'intérêt est de pouvoir gagner beaucoup de temps avec une génération dynamique de code pour l'accès aux données directement depuis le modèle de la base de données. Il est quand même nécessaire d'avoir une base dont le design ne va pas trop changer avant de se lancer dans les développements. Mon site ne contient pas de réelle intelligence ni d'analyse de données. Il se contente de les exposer et de laisser à l'humain le soin de les analyser Fig.12.

L'objet de cet article n'est pas de rentrer dans le détail de cette partie. Je ne la détaillerai donc pas. La réalisation des graphiques est faite à l'aide du composant « Chart » présent dans le framework MVC. Je n'avais pas fait de développement Web depuis de nombreuses années et j'ai été très agréablement surpris par la facilité d'utilisation du trio APS.NET + MVC + Entity Framework. Mention spéciale aux lambda expressions pour les requêtes dans les données, très puissant et à la fois très simple à manipuler Fig.13 et 14.

Intelligence et retour d'information

Pour compléter l'exemple, maintenant que j'ai réalisé une chaîne complète du capteur jusqu'à l'intelligence et l'analyse des données, je vais donc implémenter la chaîne retour qui est donc la modification du comportement de certains objets en fonction des données remontées. Mon arrosa-

Fig.9

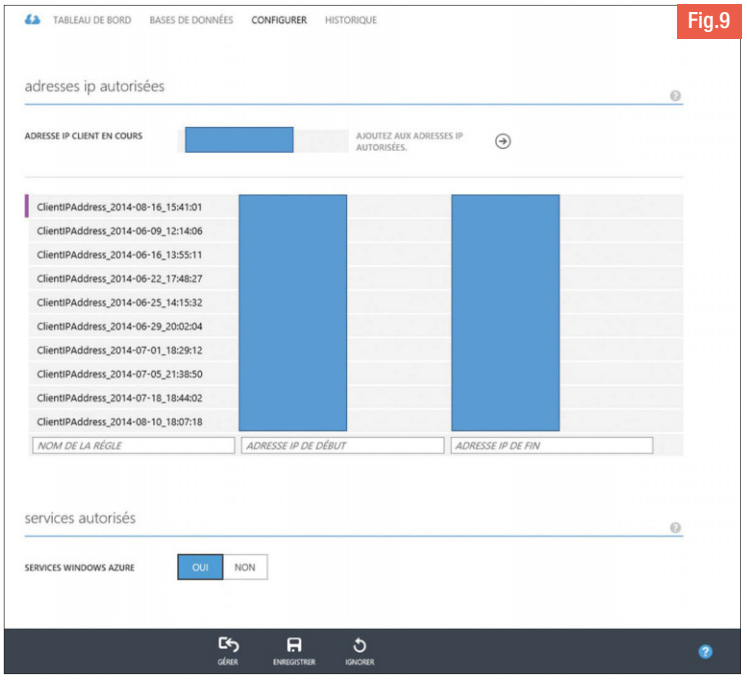


Fig.10

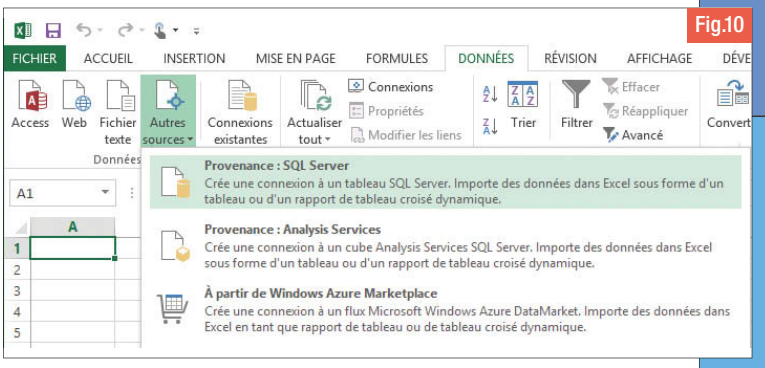


Fig.11

ge automatique viendra régulièrement vérifier s'il doit arroser ou pas. Pour cela, j'ai créé une API REST personnalisée dans Mobile Services. De façon très simple, je vérifie quelle est l'humidité du sol, et en fonction d'un niveau que j'ai déterminé, je décide qu'il faut arroser ou non. Ici le modèle est très simple mais peut très facilement utiliser le modèle de prédiction des Azure Machine Learning.

Voici le code de l'API personnalisée en javascript :

```
exports.get = function(request, response) {
    var mssql = request.service.mssql;
    //select the last humidity value created
    var sql = "SELECT TOP 1 soilhumid FROM arrosage ORDER BY __createdAt DESC";
    mssql.query(sql, {
            success: function(results) {
            //return the restults.
            response.send(statusCodes.OK, results);
            },
            error: function(err) {
            //return the erro
            response.send(statusCodes.FORBIDDEN, err);
            }
    });
};
```

Je fais donc une simple requête SQL dans la table qui stocke les informations contenant l'humidité du sol. Je récupère la dernière données (__createdAt est un timestamp automatiquement créé lors de la création de la ligne dans la base). Je renvoie donc OK (code 200) si tout est bon ainsi que la donnée soilhumid et la valeur.

Sa consommation se fait exactement comme lors de l'envoi des données dans Mobile Services. Il est donc nécessaire de faire une requête en HTTP en passant ou non des arguments.

Mon board contenant l'arrosage automatique étant en .NET Microframework, voici le code nécessaire pour récupérer les données, ici en C# :

```
// http_Client est une classe de client http très simple
// IntegratedSocket est une classe facilitant l'usage des sockets
// Même sans ces classes,l'usage reste simple
HTTP_Client myClient = new HTTP_Client(new IntegratedSocket
("nomduservice.azure-mobile.net", 80));
String myRequest = "GET http://nomduservice.azure-mobile.net/
api/nomAPI";
// nomduservice et nomAPI sont les noms du service et de l'API
myRequest += " HTTP/1.1\r\n";
myRequest += "Accept: application/json\r\n";
// la clé à utiliser est celle de l'application
myRequest += "X-ZUMO-APPLICATION: 123456789abcdefghij1234567
89abcd\r\n";
myRequest += "Host: nomduservice.azure-mobile.net\r\n";
myRequest += "Connection: Close\r\n";
myRequest += "\r\n";
HTTP_Client.HTTP_Response response = myClient._DoRequest(myRequest);
if (response.ResponseCode == 200) {
// Recherche la valeur d'humidité et décide quoi faire
}
```

Comme pour l'Arduino, le code est très similaire, l'entête http avec la clé de l'application, le nom de l'API est envoyé, la réponse ressemble à cela (ici brute) :

"HTTP/1.1 200 OK\r\nCache-Control: no-cache\r\nContent-Length: 18\r\nContent-Type: application/json; charset=utf-8\r\nServer: Microsoft-IIS/8.0\r\nx-zumo-version: Zumo.master.0.1.6.3711.Runtime\r\nX-Powered-By: ASP.NET\r\nSet-Cookie: ARRAffinity=09482d5b42db702a74aa7bd65eb9fb93fe0f43210a649f6bd3c17386ab00447d;Path=/;Domain=arrosage.azure-mobile.net\r\nDate: Sat, 16 Aug 2014 17:09:52 GMT\r\nConnection: close\r\n\r\n[{\"soilhumid\":38}]"

Une fois l'entête analysée, il reste les données « [{\"soilhumid\":38}] ». A moi ensuite dans le code de mon arrosage automatique de décider de la valeur seuil minimum pour arroser Fig.15.

Fig.13

Page d'affichage des graphiques des sondes d'humidité et de température sur écran large

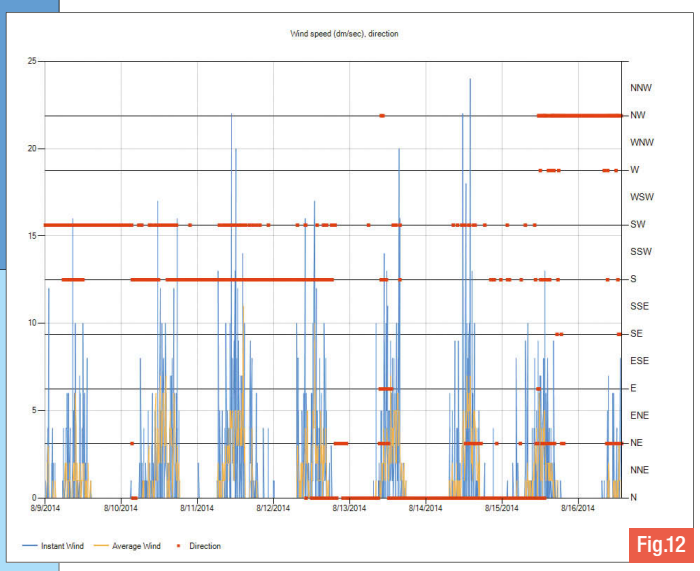


Fig.12

Création dynamique de graphiques représentant la force du vent et la direction principale

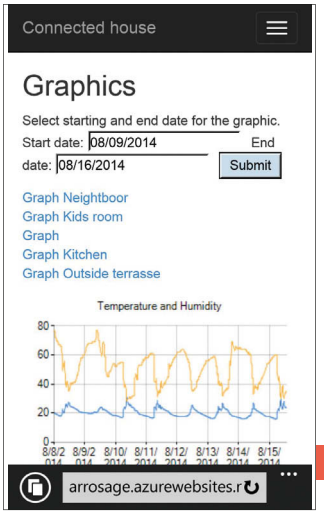


Fig.14

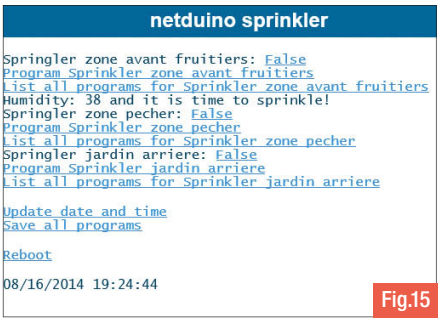


Fig.15

La même vue mais sur l'écran d'un Windows Phone. ASP.NET + MVC permet facilement d'adapter l'affichage au type de périphérique.

Interface web du Netduino en .NET Microframework gérant l'arrosage automatique. L'humidité est directement récupérée depuis l'API REST.
Je n'appelle cette API qu'une seule fois par jour en mode automatique. J'ai laissé la possibilité d'en plus pouvoir continuer à effectuer mes propres arrosages et affiche donc cette information quand la page est affichée.

Résumé

Voilà donc un exemple où j'ai connecté des objets existants (les sondes Oregon Scientific), créé de nouveaux objets (à l'aide d'ATmega328, langage C++), réalisé une architecture de collecte sur protocole propriétaire (433MHz, protocole Oregon Scientific) à l'aide de Spark.IO (architecture Arduino avec connection wifi, langage C++), publié les données de ces capteurs dans le Cloud à l'aide de Azure et les mobile services (à travers HTTP en API REST personnalisé en javascript), analysé ces données (hébergées dans Azure Web Sites en ASP.NET + MVC + Entity Framework, langage C#), les ai exposés avec ma propre API REST (créée en javascript), ai consommé ces données dans d'autres objets intelligents (à travers .NET Microframework en C#) pour influencer le comportement de mon arrosage automatique. J'espère avoir pu montrer avec cet exemple concret et que j'utilise en production depuis cet été l'ensemble des problématiques de l'Internet des objets. Tout projet d'IoT est un projet qui est donc rapidement assez complexe et nécessite de mettre en œuvre de nombreuses technologies, de créer une architecture de communication entre les objets, de faire attention à la sécurité notamment.
Même si un projet d'IoT est complexe, il reste à la portée de tout passionné. Il faut juste avoir de bonnes idées et un peu de temps devant soi. Au-delà de mettre en œuvre de nombreuses technologies pour illustrer un projet réel, mon projet m'est vraiment utile et je l'utilise réellement. J'avais déjà réalisé le pilotage de mon arrosage automatique il y a quelques années mais je n'avais pas intégré un mode automatique. Je pourrais bien sûr ajouter une simple sonde d'humidité plutôt que de me connecter au Cloud pour récupérer cette information d'un autre capteur.
Ou même plus simplement, la récupérer de ce capteur en local. Mais la publication de ces données météorologiques m'aide également à mieux gérer l'énergie chez moi, à prévoir quand il faut allumer le chauffage, descendre les stores, garder les volets fermés, arroser mes plantes d'intérieur. En déplacement ou en voyage, cela me permet également de vérifier si tout se passe bien chez moi.

Volley : la bibliothèque qui facilite les appels réseaux sous Android

Au sein d'un monde tendant à l'hyper connexion avec des réseaux mobiles toujours plus puissants, les applications sur smartphones utilisent massivement le réseau pour échanger des données avec des serveurs. S'appuyant sur la bibliothèque Apache HTTP Client depuis ses débuts, la plateforme Android ne propose pas de solutions haut niveau facilitant la gestion des appels réseaux. Présentée lors du Google I/O édition 2013, la bibliothèque Volley se veut être la solution répondant aux problèmes rencontrés par les développeurs lors de la réalisation de ces appels. Tour d'horizon d'une solution prometteuse.

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

Problématique centrale de la majorité des applications mobiles, les appels réseaux ne bénéficient pas sous Android de solution haut niveau simplifiant les échanges avec le serveur. En standard, le framework propose 2 voies pour réaliser ces échanges :

- La classe HttpURLConnection, solution légère mais souffrant de plusieurs bugs sur d'anciennes versions de l'OS,
- L'API HTTP Client d'Apache intégrée dans le SDK qui se révèle puissante mais porteuse elle aussi d'un certain nombre de bugs difficilement corrigibles sans toucher à la rétro compatibilité de l'OS

En dépit de bugs, ces solutions restent fonctionnelles et sont utilisées en production. Le vrai problème vient plutôt du fait qu'elles soient bas niveau, laissant à la charge du développeur la gestion des threads au sein desquels elles seront employées. L'utilisation d'un thread différent de l'UI Thread ayant vocation à éviter le blocage de l'interface utilisateur durant les échanges réseaux. Dans un souci de faciliter cette gestion des threads, la classe AsyncTask a rapidement fait son apparition. Elle permet d'exécuter du code dans un thread différent de l'UI Thread tout en gérant les échanges avec celui-ci à des fins de mise à jour IHM. Ainsi, le chargement d'une image depuis le réseau via une AsyncTask se fait comme suit :

```
private class LoadImageTask extends AsyncTask<URL, Void, Bitmap> {

 private ImageView imageView;

 public LoadImageTask(ImageView imageView) {
  this.imageView = imageView;
 }

 protected Bitmap doInBackground(URL... params) {
  HttpURLConnection conn = (HttpURLConnection) params[0].open
Connection();
  InputStream input = conn.getInputStream();
  return BitmapFactory.decodeStream(input);
 }

 protected void onPostExecute(Bitmap result) {
  imageView.setImageBitmap(result);
 }
}
```

Las, les AsyncTasks présentent un certain nombre de particularités induisant une mauvaise utilisation au sein des applications. Ainsi, contrairement à la croyance populaire, les AsyncTasks s'exécutent par défaut en série et il est nécessaire de réaliser un paramétrage si l'on souhaite une exécution en parallèle. En sus, la rotation de l'écran entraîne un rechargement complet depuis le réseau avec création d'une nouvelle AsyncTask alors que bien souvent le développeur oublie que la destruction des AsyncTasks en cours d'exécution lui incombe lors de cette rotation. Si cela n'est pas fait, elles continuent à tourner provoquant bien souvent un crash applicatif puisque, par exemple, tentant de mettre à jour des vues déjà recyclées. L'obligation de manipuler un cache à la main étant également un manque important pour obtenir une API de plus haut niveau. Enfin, elles connaissent de sérieux problèmes sur d'anciennes versions d'Android (Froyo par exemple).

Volley à la rescousse

Afin d'adresser ces problématiques et de fournir à la communauté Android une solution robuste de plus haut niveau pour la gestion des échanges réseaux, les équipes de Google ont mis au point la bibliothèque Volley. Légère et facile d'accès, elle présente un certain nombre d'avantages :

- Gestion de l'ensemble des demandes réseaux,
- Basée sur un mécanisme de thread pool configurable,
- Gestion automatisée du cache des requêtes,
- Annulation aisée des requêtes en cours, unitairement ou par scope,
- Priorisation des requêtes au sein d'une file d'attente,
- Support natif JSON, images et texte avec possibilités d'extension pour d'autres formats,
- Implémentation aisée de mécanismes de retry,
- Outils de débug et de tracing intégrés.

Volley étant open source, il est facile pour quiconque de modifier à souhait la bibliothèque pour la personnaliser encore plus, bien que son extensibilité permette d'éviter le recours à ces modifications. Avec l'API haut niveau de Volley, le développeur va enfin dire adieu aux AsyncTasks en voyant son travail simplifié avec une gestion du cache automatisée et un support natif du format d'échange JSON. En outre, la bibliothèque est compatible Android 2.2 ce qui couvre quasiment la totalité des smartphones Android du marché.

Au coeur de l'architecture de Volley (figure 1) se trouve la classe RequestQueue modélisant le pool de thread associé au framework. Ce dernier doit être initialisé une seule fois dans une application, de préférence à un endroit où il sera facilement accessible par la suite. Pour réaliser des appels réseaux, l'API propose la classe abstraite Request supportant des implémentations pour le chargement des données JSON avec JsonRequest et ses sous-classes, pour les données textuelles avec StringRequest ou encore les images avec ImageRequest. Bien entendu, cette classe est extensible pour s'adapter aux besoins spécifiques des applications. Les réponses réseaux sont gérées au sein des interfaces Listener et ErrorListener définies dans l'objet Response. D'autre part, l'API propose une interface Cache définissant les fonctionnalités d'un cache sous Volley. Quelques implémentations sont fournies avec

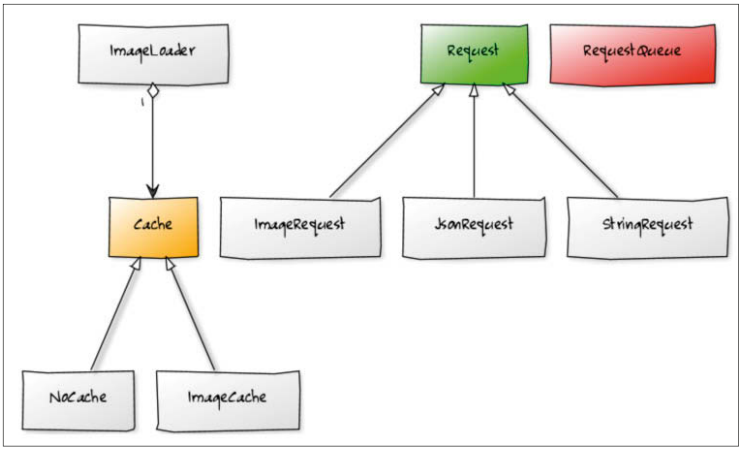


Figure 1 : Architecture simplifiée de Volley

DiskBasedCache par exemple. Enfin, un certain nombre de classes supplémentaires sont présentes facilitant la gestion des erreurs, du retry ou encore du débuggage.

Installation

Pour récupérer Volley, il faut passer directement par les sources qui sont mises à disposition sur un dépôt Git de Google. Une fois celles-ci récupérées, il reste à exécuter 2 commandes pour construire la bibliothèque et obtenir l'archive volley.jar qui sera à ajouter aux dépendances de l'application souhaitant utiliser Volley. L'installation se résume aux commandes suivantes :

```
git clone https://android.googlesource.com/platform/frameworks/volley
cd volley
android update project -p .
ant jar
```

La bonne exécution de ces commandes présuppose que l'utilisateur possède sur son poste de travail Git, le SDK Android et également l'outil de build Ant. Le JAR généré est récupérable au sein du dossier bin/ du répertoire où les sources de Volley ont été téléchargées.

Création du projet

Dans le cadre de la mise en oeuvre de Volley, nous allons réaliser une application composée d'une activité affichant une liste d'images avec un titre associé à chaque image. L'intérêt de l'application résidant dans la méthode de peuplement de cette liste puisque les images seront requêtées depuis l'API REST du service d'images Picasa de Google. L'application ainsi développée permettra de valider que les différents appels réseaux sont bien effectués en parallèle via le thread pool de Volley et sa RequestQueue. Enfin, le chargement des images se fera par pages de taille définie afin de compléter la liste au fur et à mesure du scroll sur celle-ci.

Dans Eclipse, il faut créer un projet de type Application Android en choisissant la dernière version du SDK comme cible. Le projet ainsi créé, il faut configurer le manifest de l'application en rajoutant l'activité principale MainActivity, la permission android.permission.INTERNET pour pouvoir effectuer les appels réseaux ainsi que l'objet SampleApplication au sein duquel la RequestQueue sera initialisée au lancement de l'application.

Création de la RequestQueue

La première étape pour utiliser Volley consiste à créer son thread pool via l'objet RequestQueue. Ce dernier ayant vocation à n'être créé qu'une fois et être accessible par l'ensemble des classes de l'application, il est pertinent de réaliser son instanciation au sein de la classe SampleApplication appelée au démarrage de l'application. Héritant d'Application, elle créé également l'objet ImageLoader utilisé par la suite pour le chargement des images :

```
public class SampleApplication extends Application {

 private RequestQueue requestQueue;
 private ImageLoader imageLoader;

 @Override
 public void onCreate() {
  super.onCreate();
  init();
 }

 @Override
 public void onTerminate() {
  requestQueue.stop();
  super.onTerminate();
 }

 private void init() {
  requestQueue = Volley.newRequestQueue(this);
  int memClass = ((ActivityManager) getSystemService(Context.
ACTIVITY_SERVICE)).getMemoryClass();
  int cacheSize = 1024 * 1024 * memClass / 8;
  imageLoader = new ImageLoader(requestQueue, new BitmapLru
Cache(cacheSize));
 }

 public RequestQueue getRequestQueue() {
  return requestQueue;
 }

 public ImageLoader getImageLoader() {
  return imageLoader;
 }
}
```

Ici, on note l'importance de stopper le thread pool à la fermeture de l'application via l'appel à la méthode stop du RequestQueue. Avantage par rapport aux AsyncTasks, l'appel à cette méthode garantit l'arrêt de l'ensemble des appels réseaux en file d'attente.

D'autre part, on remarque l'instanciation de la classe BitmapLruCache et son association avec l'ImageLoader. Ici, nous implémentons l'interface ImageCache, dédiée au cache d'images, avec un cache LRU pour bitmaps :

```
public class BitmapLruCache extends LruCache<String, Bitmap>
implements ImageCache {

 public BitmapLruCache(int maxSize) {
  super(maxSize);
 }

 @Override
 protected int sizeOf(String key, Bitmap value) {
  return value.getRowBytes() * value.getHeight();
 }

 @Override
 public Bitmap getBitmap(String url) {
  return get(url);
 }
```

```
 @Override
 public void putBitmap(String url, Bitmap bitmap) {
  put(url, bitmap);
 }
}
```

Cette implémentation de cache est minimaliste et il est à tout à fait possible de réaliser des implémentations plus complexes pour des besoins particuliers.

Préparation du modèle

L'application devant manipuler une liste de données issues de l'API REST de Picasa, il est intéressant de modéliser les éléments retournés par la service au sein d'un POJO PicasaEntry. Concrètement, il s'agit d'une classe stockant le titre et l'URL de l'image. En sus, il est nécessaire de définir l'Adapter qui sera utilisé pour la gestion des données de la liste. Devant manipuler une collection d'objets de type PicasaEntry, notre Adapter étend ArrayAdapter et prendre l'ImageLoader de l'application en entrée.

Au sein de la méthode getView de l'adapter, la seule spécificité réside dans la manipulation de l'image associée aux éléments de la liste. Pour faciliter le chargement de ces images, l'API Volley propose la vue NetworkImageView. Ainsi, si une URL est présente pour un objet PicasaEntry, il suffit d'appeler la méthode setImageUrl avec en entrée l'URL de l'image et l'ImageLoader. Dans le cas où aucune URL n'est définie, on appelle simplement la méthode setImageResource pour positionner une image donnée. Le code de l'Adapter est donc le suivant :

```
public class PicasaArrayAdapter extends ArrayAdapter<PicasaEntry> {

 private ImageLoader imageLoader;

 public PicasaArrayAdapter(Context context, int textViewResourceId,
 List<PicasaEntry> objects, ImageLoader imageLoader) {
  super(context, textViewResourceId, objects);
  this.imageLoader = imageLoader;
 }

 @Override
 public View getView(int position, View convertView, View
Group parent) {
  View v = convertView;

  if (v == null) {
   LayoutInflater vi = (LayoutInflater) this.getContext().get
SystemService(Context.LAYOUT_INFLATER_SERVICE);
    v = vi.inflate(R.layout.row, null);
  }

  NetworkImageView image = (NetworkImageView) v.findViewBy
Id(R.id.thumb);
  TextView title = (TextView) v.findViewById(R.id.title);
  PicasaEntry entry = getItem(position);

  if (entry.getThumbnailUrl() != null) {
   image.setImageUrl(entry.getThumbnailUrl(), imageLoader);
  } else {
   image.setImageResource(R.drawable.no_image);
  }

  title.setText(entry.getTitle());

  return v;
 }
}
```

Le layout row.xml utilisé pour afficher chaque élément de la liste est simplement composé d'un NetworkImageView et d'un TextView.

Appel réseau

L'activité principale de l'application affiche un ListView qu'elle va remplir suite au retour de l'appel réseau vers Picasa. Pour remplir la liste, nous réalisons un chargement par pages avec une taille de page définie. Au lancement de l'activité, la première page est donc chargée alors que le chargement des pages suivantes est réalisé au fur et à mesure du scroll vers le bas de la liste. L'idéal étant d'anticiper la demande de chargement de nouvelles images un peu avant d'atteindre la fin des éléments de la liste. Le chargement d'une page s'appuie sur JsonObjectRequest puisqu'un objet JSON est requêté depuis le service REST de Picasa :

```
private void loadPage() {
 RequestQueue queue = application.getRequestQueue();
 int startIndex = 1 + picEntries.size();
 JsonObjectRequest myReq = new JsonObjectRequest(Method.GET,
"https://picasaweb.google.com/data/feed/api/all?q=android+robot
+png&max-results="
  + RESULTS_PAGE_SIZE
  + "&thumbsize=160&alt=json"
  + "&start-index="
  + startIndex, null, createReqSuccessListener(), createReq
ErrorListener());
 queue.add(myReq);
}
```

Ici, on cherche les images associées à droid le petit robot de la plateforme Android. La requête est de type GET et est ajoutée à la RequestQueue de Volley pour exécution. L'index de départ est déterminé à partir du nombre de d'éléments déjà chargés dans la liste. Les méthodes createReqSuccessListener et createReqErrorListener servent quant à elle à créer les listeners de réponse pour l'appel réseau. Si l'objet Response.ErrorListener se contente d'afficher une boîte de dialogue affichant un message d'erreur, l'objet Response.Listener est plus intéressant puisque parsant l'objet JSON récupéré et mettant à jour la liste des images :

```
private Response.Listener<JSONObject> createReqSuccessListener() {
 return new Response.Listener<JSONObject>() {
  @Override
  public void onResponse(JSONObject response) {
   try {
    JSONObject feed = response.getJSONObject("feed");
    JSONArray entries = feed.getJSONArray("entry");

    for (int i = 0; i < entries.length(); i++) {
     JSONObject entry = entries.getJSONObject(i);
     String url = null;
     JSONObject media = entry.getJSONObject("media$group");

  if (media != null && media.has("media$thumbnail")) {
```

```
    JSONArray thumbs = media.getJSONArray("media$thumbnail");

    if (thumbs != null && thumbs.length() > 0) {
        url = thumbs.getJSONObject(0).getString("url");
       }
      }

     picEntries.add(new PicasaEntry(entry.getJSONObject("title").
getString("$t"), url));
     }

     mAdapter.notifyDataSetChanged();
    } catch (JSONException e) {
     showErrorDialog();
    }
   }
  };
}
```

L'ArrayAdapter de la liste ayant été créé avec en entrée l'objet picEntries contenant les éléments PicasaEntry récupérés depuis le réseau, il suffit de notifier que la liste doit être mise à jour via l'appel à la méthode notifyDataSetChanged lorsque l'on ajoute de nouveaux éléments à la collection picEntries.

Chargement différé

Le dernier point en suspens concerne le chargement différé de la liste d'images lors du scroll sur la ListView. Pour implémenter ce comportement, il faut placer un OnScrollListener sur la liste qui permet de connaître le nombre d'items affichés et en réserve afin de procéder ou non au chargement de la page suivante de résultats. Le seuil à partir duquel ce chargement est opéré étant arbitraire. L'objet EndlessScrollListener a l'allure suivante :

```
public class EndlessScrollListener implements OnScrollListener {
 private int visibleThreshold = 5;
 private int currentPage = 0;
 private int previousTotal = 0;
 private boolean loading = true;

 public EndlessScrollListener() {
 }

 public EndlessScrollListener(int visibleThreshold) {
  this.visibleThreshold = visibleThreshold;
 }

 @Override
 public void onScroll(AbsListView view, int firstVisibleItem,
  int visibleItemCount, int totalItemCount) {
  if (loading) {
   if (totalItemCount > previousTotal) {
    loading = false;
    previousTotal = totalItemCount;
    currentPage++;
   }
  }

  if (!loading && (totalItemCount - visibleItemCount) <= (first
VisibleItem + visibleThreshold)) {
   loadPage();
   loading = true;
  }
 }

 @Override
 public void onScrollStateChanged(AbsListView view, int scroll
State) {
 }

 public int getCurrentPage() {
  return currentPage;
 }
}
```

Screenshot de l'application **Fig.2**

Test de l'application

Le test de l'application consiste dans un premier temps à vérifier que la liste d'images requêtées s'affiche bien et dans un second temps que les différentes images chargées le sont bien en parallèle. La dernière partie du test servant quant à elle à valider le chargement différé des éléments de la liste lors du scroll vertical vers le bas. La figure 2 présente le résultat du test de l'application développée.

Une rotation de l'écran montre clairement que les images déjà téléchargées ont été mises en cache compte tenu de la vitesse à laquelle elles se réaffichent. D'autre part, pour déboguer et tracer le comportement de Volley de manière poussée, il suffit d'activer la propriété suivante :

```
adb shell setprop log.tag.Volley VERBOSE
```

Bien entendu, l'application nécessiterait plusieurs améliorations afin d'être opérationnelle en production avec par exemple une gestion plus fine de la rotation de l'écran, l'affichage d'un indicateur de progression lors du chargement ou encore la vérification de fin des enregistrements retournés par la service Web.

Conclusion

Au travers de l'exemple du chargement d'une liste d'images depuis un service REST, cet article aura mis en exergue la puissance alliée à la simplicité de la bibliothèque Volley. Conçue pour gérer tous les types d'opérations RPC, elle se révèle être une solution de haut niveau simplifiant grandement les appels réseaux au sein des applications Android. En outre, le support d'un certain nombre de types de données en standard amène un gain de temps indéniable tout comme son extensibilité puisque Volley pourra, par exemple, facilement gérer le XML. Déjà utilisé, dans bon nombre d'applications Android de Google, Volley s'annonce comme la solution du futur répondant à la plupart des problématiques réseaux sous Android. Attention toutefois à l'utiliser à bon escient et ne pas l'employer dans des contextes inappropriés puisqu'il est notamment déconseillé d'y recourir pour le chargement de gros volumes de données.

ASP.NET vNext : quoi de neuf ?

ASP.NET vNext correspond à la prochaine version des technologies Web de Microsoft à savoir ASP.NET MVC, Web API, Web Pages and SignalR. Cette nouvelle mouture amène des changements profonds dans la façon dont Microsoft exploite ces technologies. Nous allons faire ici un tour d'horizon de ces évolutions.


Eric Galiano
Consultant chez Cellenza
Cellenza - Software Development Done Right
Blog : http://blog.cellenza.com/


Pierre-Henri Gache
Consultant Senior chez Cellenza
Cellenza - Software Development Done Right
Blog : http://www.pierrehenrigache.com/

Cellenza

VISUAL STUDIO 2014

Les équipes qui ont travaillé sur cette nouvelle version ont cherché à rendre ASP.NET plus ouvert et plus flexible. C'est pourquoi ASP.NET vNext n'est plus lié à Visual Studio : il est possible de coder avec votre éditeur favori. En effet, la compilation s'effectuant à la volée et les références projets s'ajoutant directement dans un fichier JSON, il est tout à fait envisageable de le faire sans Visual Studio. Cela permet également, dans un environnement restreint comme un serveur Web, de pouvoir modifier son application avec un simple éditeur de texte. Cela peut être très utile notamment lors de la mise au point de son application. Malgré tout, Visual Studio reste l'outil le plus adapté pour développer son application Web au vu de l'étendue des fonctionnalités qu'il propose.

Roslyn Compiler

Voilà une innovation qui en ravira beaucoup : il est maintenant possible d'éditer son code C# et de visualiser le changement dans le navigateur sans avoir à recompiler le projet.

Cette nouveauté est possible grâce à l'utilisation du compilateur .NET Roslyn (nous n'allons pas voir ici tout ce dont est capable ce compilateur, mais nous vous invitons à regarder de plus près cette nouvelle mouture du compilateur de Microsoft). Concrètement, à chaque modification du code source, le serveur Web de développement est arrêté et lors de l'appel à une page, Roslyn va recompiler l'ensemble du code source et fournir ainsi la dernière version du site. Un point intéressant à noter est que si vous consultez le contenu du répertoire « bin » qui se trouve dans votre projet, il sera vide. La compilation étant effectuée en mémoire, aucune dll ni pdb n'est déposée sur le disque.

Ce processus peut être un peu déroutant au début, c'est pourquoi il faut bien avoir conscience de son fonctionnement.

Framework optimisé pour le Cloud

Jusqu'à présent, lors du développement d'un site Web ou bien d'une API REST, n'importe quelle application utilisait le Framework .Net. Ce dernier fournit l'ensemble des éléments pour ces types de projets, mais également tout ce qui est lié aux autres types d'applications (Winforms, WPF par exemple). Afin d'optimiser le déploiement de sites dans le Cloud, il est désormais possible d'utiliser un sous ensemble du Framework .Net : le K Runtime. De plus, l'abandon de System.Web.dll permet de construire des applications plus modulables, car on peut choisir les librairies que l'on souhaite intégrer à notre projet.

Open Source et Cross Platform

Un autre point qui mérite d'être souligné est la disponibilité des sources sur GitHub. Il s'agit d'une réelle évolution dans la manière dont Microsoft souhaite vivre sa solution. De plus grâce à Mono, il est possible de faire tourner son application sur d'autres plateformes que Windows.

Créer votre premier projet

Avec la nouvelle version de Visual Studio, vous pouvez désormais créer, en plus des traditionnels sites ASP.Net MVC, des sites ASP vNext. Lorsque que vous aurez créé un projet à partir de ces nouveaux Templates, deux choses peuvent être remarquées. La première est la disparition des fichiers Global.asax et Web.config. La seconde est liée aux références du projet qui font toutes appel à des packages NuGet. De plus, si vous faites un clic droit sur les références du projet, vous verrez qu'il n'y a aucun moyen d'en ajouter de nouvelles avec Visual Studio. Voyons maintenant tout cela d'un peu plus près.

Pour bien comprendre ces différences, créons un nouveau projet basé sur le template « ASP.NET vNext Empty Web Application ». En consultant l'explorateur de solution, vous pourrez constater que seulement deux fichiers ont été créés : « Startup.cs » et « project.json ». Le premier fichier contient le code permettant d'initialiser l'application. Nous allons maintenant voir à quoi sert le deuxième fichier et en quoi son apparition change radicalement le paramétrage du projet par rapport à ce que l'on avait auparavant.

Fichier Project.json

Ce fichier est l'un des plus importants car il permet de centraliser la configuration du projet et la gestion des dépendances. Pour cela, on retrouve plusieurs sections dans ce fichier comme le montre l'exemple ci-dessous. Intéressons-nous maintenant à chacune d'entre elles.

```
{
  "dependencies": {
    "EntityFramework.SqlServer": "7.0.0-alpha3",
    "Microsoft.AspNet.Mvc": "6.0.0-alpha3",
    "Microsoft.AspNet.Identity.EntityFramework": "3.0.0-alpha3",
    "Microsoft.AspNet.Identity.Authentication": "3.0.0-alpha3",
    "Microsoft.AspNet.Security.Cookies": "1.0.0-alpha3",
    "Microsoft.AspNet.Server.IIS": "1.0.0-alpha3",
    "Microsoft.AspNet.Server.WebListener": "1.0.0-alpha3",
    "Microsoft.AspNet.StaticFiles": "1.0.0-alpha3",
    "Microsoft.Framework.ConfigurationModel.Json": "1.0.0-alpha3",
    "Microsoft.VisualStudio.Web.BrowserLink.Loader": "14.0-alpha2"
  },
  "commands": {
    /* Change the port number when you are self hosting this
application */
    "web": "Microsoft.AspNet.Hosting --server Microsoft.AspNet.
Server.WebListener --server.urls http://localhost:5000"
  },
  "frameworks": {
    "net451" : { },
    "k10" : { }
```

```
  }
}
```

References

La première section nommée « dependencies » regroupe la liste des packages NuGet qui seront utilisés dans le projet. Comme évoqué précédemment, inutile de chercher un autre moyen pour éditer cette partie, il n'y en a pas. Toutefois, l'IntelliSense de Visual Studio est présente pour vous aider notamment dans le choix des numéros de version.
Ces derniers sont d'ailleurs optionnels. Si rien n'est spécifié, c'est la dernière version qui sera utilisée.
Dès lors que l'on a compris comment fonctionne la gestion des dépendances, on peut se demander pourquoi elles ne sont plus dans le fichier « csproj ».
La raison est simple et a déjà été évoquée : pouvoir éditer son projet en se passant de Visual Studio. C'est pourquoi désormais le fichier « csproj » sert uniquement à référencer les différents fichiers.

Commandes

Cette section permet de définir des commandes pour démarrer son site. Dans notre exemple, elle permet de lancer le serveur sur le port 5000. Pour cela, il suffit d'initier une invite de commande puis de se placer dans le répertoire du projet et de saisir l'instruction suivante : « k web ». Un point intéressant à remarquer est la commande qui commence par le namespace « Microsoft.AspNet.Hosting ». Ce dernier est contenu dans la seule dll du répertoire bin du projet. Elle sert à amorcer le lancement du site. Inutile de dire que cette dll ne devra pas être supprimée, auquel cas la commande précédente échouera.

Frameworks

Cette dernière section permet de définir la liste des Frameworks disponibles : le Framework 4.5.1 traditionnel et celui qui est optimisé pour le Cloud : le "K runtime".

Side by side

Hier, si vous exécutiez plusieurs sites sur votre serveur IIS, ils auraient tous utilisé la même version du Framework, à savoir celle qui est installée. Avec ASP.Net vNext et l'introduction du side by side, c'est différent. Il est possible d'exécuter son application de manière autonome et de lui associer une version spécifique du Framework K.
Pour cela un nouvel outil est disponible : K Version Manager (KVM). Il permet de lister et de choisir la version du Framework à associer à notre application. Nous allons maintenant voir comment faire tourner deux applications Web utilisant deux versions différentes. La première étape consiste donc à l'installer :

```
@powershell -NoProfile -ExecutionPolicy unrestricted -Command
"iex ((new-object net.webclient).DownloadString('https://raw.
githubusercontent.com/aspnet/Home/master/kvminstall.ps1'))"
```

En utilisant la commande "kvm list", il est possible de visualiser les différentes versions disponibles.

```
PS C:\Users\VS2014CPT3> kvm list

Active Version      Runtime Architecture Location                       Alias
------ -------      ------- ------------ --------                       -----
       1.0.0-alpha3 svr50   x86          C:\Users\VS2014CPT3\.kre\packages default
       1.0.0-alpha3 svrc50  x86          C:\Users\VS2014CPT3\.kre\packages
```

Pour sélectionner une version, la commande à utiliser est "kvm use [nom de la version]". On remarque ensuite un astérisque dans la colonne "Active" indiquant la version sélectionnée.

```
PS C:\Users\VS2014CPT3> kvm use 1.0.0-alpha3
Adding C:\Users\VS2014CPT3\.kre\packages\KRE-svr50-x86.1.0.0-alpha3\bin to process PATH
PS C:\Users\VS2014CPT3> kvm list

Active Version      Runtime Architecture Location                       Alias
------ -------      ------- ------------ --------                       -----
  *    1.0.0-alpha3 svr50   x86          C:\Users\VS2014CPT3\.kre\packages default
       1.0.0-alpha3 svrc50  x86          C:\Users\VS2014CPT3\.kre\packages
```

Pour lancer l'application, placez-vous sur le répertoire contenant le fichier project.json et lancez la commande "k web". L'application est maintenant exécutée avec la version spécifiée. Ici "web" correspond à la clé de la partie "commands" du fichier "project.json" que nous avons vu avant.
Pour lancer l'application avec une autre version du Framework, il suffit d'ouvrir une nouvelle fenêtre de commande et d'exécuter la commande "kvm use" en spécifiant une autre version, puis de lancer l'application.
Cette fonctionnalité à un réel sens dans au moins deux cas de figure : la montée de version du Framework et la migration de l'application. Dans le premier cas, en admettant que vous souhaitiez upgrader le Framework .Net utilisé par votre application, vous pourrez installer deux versions de votre site. L'un tournera avec la version actuelle et l'autre avec la nouvelle version. Lorsque l'ensemble des tests de non régression aura été effectué, vous pourrez diriger vos utilisateurs vers la nouvelle instance. En cas d'imprévu, le basculement vers l'ancienne version sera des plus faciles car les deux seront disponibles sur votre serveur. Dans le second cas, celui de la migration, vous pourrez également tirer avantage de cette fonctionnalité en migrant par morceau votre application. Les nouveaux modules utiliseront le Framework K tandis que les anciens continueront d'utiliser la version classique. Cette cohabitation vous permettra de migrer facilement.

Injection de dépendance

Une autre fonctionnalité intéressante de cette version d'ASP.NET est la mise à disposition d'un outil d'injection de dépendances. La configuration du conteneur s'effectue au niveau de la méthode « Configure » du fichier « Startup.cs ». Comme le montre l'exemple ci-dessous, il est possible d'enregistrer, par exemple, un service :

```
public void Configure(IBuilder app)
{
  // Ajout de services
  app.UseServices(services =>
  {
    services.AddScoped(typeof(IMonService), typeof(MonService));
  });
}
```

Ensuite si vous désirez l'utiliser, dans un contrôleur par exemple, il suffira d'ajouter au constructeur de ce dernier un paramètre du type de votre service. Lors du Runtime, une instance de votre service sera injectée et pourra ensuite être utilisée. Bien que cette fonctionnalité soit fournie de façon standard, il est tout de même possible d'utiliser n'importe quel autre Framework d'injection de dépendance.

Publication

La fonctionnalité de publication a elle aussi évolué. En effet, lors de la publication de votre site, deux choix s'offrent à vous : soit publier vers Azure, soit publier vers le disque. C'est la deuxième option qui mérite un peu d'attention. Avec l'utilisation des packages NuGet en guise de dépendances et avec la mise en place du Framework optimisé pour le Cloud, il est possible d'avoir une fonction de publication qui package l'ensemble de ces éléments pour avoir un tout cohérent.
Ainsi, si on lance une publication via l'option « Publish », puis que l'on examine le contenu du répertoire de sortie, on peut constater plusieurs points. Premièrement, chaque « commands » du fichier « project.json » va

donner lieu à un fichier cmd portant le même nom. Ensuite, vous trouverez deux répertoires : « approot » et « wwwroot ». Le premier contient l'ensemble des packages NuGet nécessaires ainsi que les sources du projet. Le second, quant à lui, contient la structure du projet avec toutes les vues. Ainsi, si vous lancez l'un des fichiers « cmd » générés, vous pourrez accéder à votre site. Vous pouvez également effectuer la même opération sur un poste dépourvu d'environnement de développement ou encore grâce à Mono sur un Mac ou Linux.

Déploiement sur Azure

L'intégration d'Azure a été améliorée: il est désormais plus facile de déployer son site Web sur le Cloud de Microsoft. En cliquant droit sur le projet et en cliquant sur "Publish", l'option "Microsoft Azure Websites" apparaît. Une fenêtre s'ouvre nous permettant de choisir, soit de déployer sur un site existant, soit de créer un nouveau site. Ici, nous allons créer un nouveau site en cliquant sur "New...". Il nous est maintenant demandé de rentrer le nom du site Web, une zone géographique dans laquelle sera hébergé le site. Il nous est aussi possible de créer et lier notre site à une base de données afin d'utiliser, entre autres, les fonctionnalités d'ASP.Net Identity. Nous pouvons maintenant valider la création du site ainsi que la publication. Visual Studio va automatiquement déployer le site sur Azure et ouvrir le navigateur par défaut pour visualiser le site fraîchement déployé.

CONCLUSION

Avec cette nouvelle version d'ASP.NET, Microsoft nous apporte de véritables changements sur la manière de construire et d'exploiter un site. Il ne s'agit pas simplement d'une modification de quelques APIs, mais bien d'une refonte en profondeur. En utilisant les dernières technologies telles que le compilateur Roslyn, vous pourrez désormais construire des applications modulables et capables de s'adapter à n'importe quel environnement qu'il soit on- premise ou dans le Cloud. De plus, l'ouverture vers les systèmes Linux et Mac via Mono permet d'envisager de nombreux scénarios de déploiement. Enfin, la disponibilité des sources sur GitHub permet à tout moment d'implémenter ses propres fonctionnalités au sein même des librairies ou bien encore de découvrir et de comprendre le fonctionnement de chaque partie du système.

Parse.com kezako ?

Parse est une plateforme de développement hébergée dans le Cloud, qui permet de développer des applications mobiles facilement dans une infrastructure back-end info gérée en SAAS. Cette plateforme rachetée en avril 2013 par Facebook, renforce la position de ce dernier sur le mobile et ses développeurs. Tout d'abord ce service est GRATUIT, une offre payante existe mais est principalement destinée aux applications à fort trafic.

GIACOPINO Cyril & MORICEAU Mathieu
Studio YMAGYN
www.ymagyn.com

Parse propose une gamme de produits indépendants et complémentaires regroupés en 3 catégories. **Parse Core** contient Parse Data, Parse social, Parse Cloud code, et Parse background Jobs. **Parse Push** permet de gérer l'envoi de push sur les populations profilées par canaux et par géolocalisation. Enfin **Parse Analytics** offre le tracking comportemental, et le monitoring de campagnes push.
Concernant les kits de développements fournis par Parse, ils couvrent toutes les technologies mobiles existantes, Objective C, Java, .NET et Javascript. De plus, Parse ne permet pas seulement de la programmation client mobile. Il s'adresse aussi aux développeurs Web côté serveur incluant depuis fin juillet un SDK PHP.

Parse.com c'est smart mais ça m'apporte quoi ?

Rapidité de développement

Comme tout développeur mobile, ce dernier se concentre sur l'UI et l'intégration des données externes fournies par un autre développeur, généralement d'une autre équipe ou même d'une autre société. Parse permet de rompre ces frontières.
Le schéma actuel Fig.1. Le schéma Parse Fig.2.
Le développeur devient maître de l'UI et peut facilement intervenir sur le fonctionnel sans même sortir de son langage de prédilection, la connexion à la source de données se faisant intra app; si l'on veut vulgariser, nous dirons que le client mobile se connecte directement à la source de données sans passer par un serveur distant tiers.

Scalable

Quand vous êtes responsable de l'architecture machine, il est de votre responsabilité que tout fonctionne sans bugs (ce n'est déjà pas facile avouez-le !) et que votre applicatif supporte la montée en charge en cas de nombreuses connexions utilisateurs.
Aujourd'hui provisionner son serveur dédié n'est plus envisageable car avec la mobilité, la montée en charge peut atteindre des pics extrêmement vite. La solution est donc d'héberger son applicatif sur du Cloud permettant d'être scalable très rapidement. Azure ou Heroku sont ceux que j'utilise principalement.
Une fois l'applicatif déployé, vous avez besoin d'utiliser une source de données comme MongoDB, chez MongoLab à 200 € par mois TTC, profitant ainsi d'un serveur maître et d'une escale ainsi que d'un arbitre faisant le load-balancing.
Great ! Nous avons maintenant une infrastructure digne de ce nom pouvant supporter une montée en charge avec un coût mensuel moyen de 300 euros TTC par mois !

Et si maintenant j'utilise Parse, comment cela se passe?

Je me connecte sur parse.com avec mon compte Facebook, je crée mon application et .... et c'est tout ! Et cela pour la modique somme de zéro € TTC.

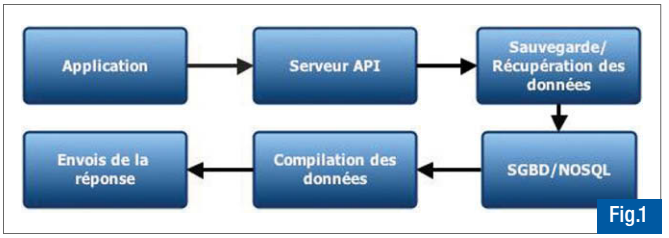


Fig.1

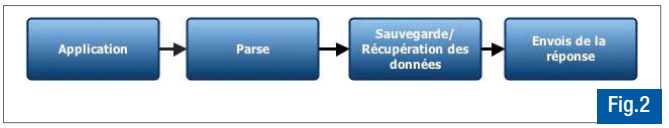


Fig.2

Vous avez donc une offre complète, très simple d'utilisation, scalable et sécurisée.
Par ailleurs, gardez à l'esprit que c'est Facebook et ses 2 milliards d'utilisateurs derrière ce produit. En tant que développeur, c'est l'assurance que mes développements reposent sur quelque chose de solide.
En parlant de Facebook, qu'en est-il des données hébergées et des informations relatives à votre vie privée ? Il faut savoir que sur Parse vos données sont et resterons vos données ! Certes vous êtes hébergé sur le réseau Facebook, mais aucune donnée de qualification ou d'enrichissement de votre profil n'est transmise à la société mère.

Parse.com est gratuit, c'est donc le messie ?

Moi-même je me suis posé la question. Parse offre un service gratuit, avec des ressources serveurs, un support, des SDK qu'une équipe dédiée corrige et maintien, et aucune donnée n'est collectées par Facebook, il y a forcément un loup !
En fait non. Lorsque l'on prend de la hauteur et que l'on analyse la chose, on perçoit la stratégie de Facebook dans l'acquisition de Parse.com. Facebook arrivé sur le tard, a d'ores et déjà perdu la bataille du Web sur la publicité (car Facebook vit grâce à la pub) face au géant Google. Facebook a donc massivement investi dans le mobile afin d'en devenir le leader. C'est aujourd'hui le premier et la société compte bien assurer cette position.
En définitive, pour Facebook, plus d'applications mobiles sont développées, plus la compagnie renforcera sa place dominante sur le marché. Pour accentuer cela Facebook souhaite démocratiser la création d'applications mobiles et Parse en est l'outil. Pas folle la guêpe !
Trèves de bavardages, entrons maintenant dans l'outil lui-même, c'est finalement tout ce qui nous intéresse.
1 - 2 - 3 Parsez !

Parse Core

Parse Data

De mon temps (oui j'ai 32 ans dont 16 ans de développement, ce n'est pas vieux finalement), le traitement des données passait par des SDGB, je travaillais principalement sur MySQL. Sur MySQL nous devions créer notre modèle de données donc nos tables et nos colonnes pour pouvoir stocker nos données.
La programmation objet nous a incité à travailler avec une couche

d'abstraction supplémentaire permettant de manipuler des objets, des instances de classes, et de les stocker dans une base de données, c'est ce que l'on appelle un O.R.M. (Mapping Objet Relationnel).
Puis vinrent les bases de données sur du NoSQL; il devenait possible de stocker des données et de les requêter sans passer par un modèle de données fixe. Cela devenait très rapide de stocker des données car plus de schéma à définir, je venais directement stocker mes objets.
Toutefois, le fait de ne pas avoir de modèle posait d'autres problèmes en développement. Par exemple, une même propriété ("colonne" en MYSQL) nommée "Amount" pouvait désormais contenir un "number", mais aussi éventuellement une "string" ou même des objets complexes type "Array" ou "Date". Enfin, d'un document à l'autre, la propriété pouvait ne même pas exister... Eh bien Parse Data dans tout cela c'est quoi? C'est simplement le meilleur des deux mondes !
Vous créez une instance de classe "Product" avec une propriété "Amount" de type "int" que vous stockez sur Parse directement. Le système va automatiquement créer le modèle à partir de ce qui est stocké la première fois; la propriété sur "Amount" sera typée "Number" et cela sera immuable. Si d'aventure vous y stockez une valeur pour "Amount" de type "string" ou "array", Parse vous criera dessus et renverra une erreur. Voyons un exemple sur Parse pour l'insertion d'un objet dans une classe : EXEMPLE CODE PARSE EN OBJECTIVE C : Fig.3.
Voyons le code équivalent qu'il aurait fallu pour réaliser la même chose sur une architecture maison hors de Parse. Voyons un exemple sur Parse pour l'insertion d'un objet dans une classe : EXEMPLE CODE EN OBJECTIVE C APPEL REST : Fig.4. EXEMPLE CODE COTE SERVEUR POUR STOCKER SUR MONGO : Fig.5.
Vous allez aussi me dire que les SGDB permettent, comme leur nom l'indique, de gérer des relations entre les données stockées, et que le NoSQL ne le permet pas. Que Neni, Parse vous permet de gérer des Pointer (référence à un objet d'une autre classe, One To One) et des Relations (les relations One To Many ou Many To Many).
Voyons un exemple sur Parse pour l'insertion d'un objet dans une classe en utilisant le vaisseau précédemment créé : EXEMPLE CODE PARSE EN OBJECTIVE C : Fig.6.
Attention tout de même à ne pas abuser des requêtes incluant des relations côté client car le traitement de ces requêtes peuvent être parfois coûteux en terme de temps. Mais des solutions toutes simples existent pour palier ce problème, c'est soit d'utiliser des **Cloud Functions,** soit des **Background Jobs** qui seront présentés un peu plus loin dans cet article. Soyez patient (j'aime entretenir le suspens).

```
PFObject *enterprise = [PFObject objectWithClassName:@"Ship"];
enterprise[@"length"] = @289;
enterprise[@"width"] = @132;
enterprise[@"name"] = @"USS Enterprise";
[enterprise saveInBackground];
```
Fig.3

```
NSString *url = @"http://monserver.fr/ship";
NSString *post = @"length=289&width=132&name=Enterprise";
NSData *postData = [post dataUsingEncoding:NSASCIIStringEncoding allowLossyConversion:YES];
NSString *postLength = [NSString stringWithFormat:@"%lu",(unsigned long)[postData length]];

NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:[NSURL URLWithString: url]];
[request setHTTPMethod:@"POST"];
[request setValue:postLength forHTTPHeaderField:@"Content-Length"];
[request setValue:@"application/x-www-form-urlencoded" forHTTPHeaderField:@"Content-Type"];
[request setHTTPBody:postData];
```
Fig.4

```
$m = new MongoClient();
$collection = $m->selectCollection('database', 'ship');

$ship = array('length' => $_POST['length'], 'width' => $_POST['width'], 'name' => $_POST['name']);
$collection->insert($ship);

header('Content-Type: application/javascript');
echo json_encode(array('error' => false, 'message' => "Vaisseau créé avec succès."))
```
Fig.5

```
PFObject *constellation = [PFObject objectWithClassName:@"Ship"];
constellation[@"length"] = @364;
constellation[@"width"] = @168;
constellation[@"name"] = @"USS Constellation";
constellation[@"ally"] = enterprise;
[constellation saveInBackground];
```
Fig.6

Parse Social

Aujourd'hui, l'intégration des réseaux sociaux est devenue un must-have dans les applications mobiles. Exit l'identification avec un e-mail et un mot de passe, exit la page d'inscription, désormais c'est votre compte Facebook et Twitter qui vous identifient dans une application mobile !
Et si je vous disais que Parse vous offre un fonctionnement natif au système de connexion de ces deux réseaux sociaux ?
Inscription traditionnelle de l'utilisateur : Fig.7.
Inscription de l'utilisateur avec **Parse** : Fig.8.
Comme vous pouvez le voir, les différentes actions effectuées, mais aussi les différents services appelés sont bien moindre avec Parse.
Et si nous connections un utilisateur à Facebook ? Facile !
CONNEXION A FACEBOOK : Fig.9.

Parse Cloud

Pour Parse, un développeur doit pouvoir développer une application mobile sans aucune architecture serveur. Certaines applications requierent un traitement spécifique sur votre serveur, et c'est là qu'intervient le **Parse Cloud**. Vous n'êtes pas familier avec le développement d'application mobile ? Ce n'est pas grave ! Le Cloud Code fonctionne grâce au SDK Javascript de Parse !
Parse Cloud se divise en trois parties, les **Cloud Functions**, les **Webhooks** et les **Background Job**.
Les **Webhooks** vous permettent d'initialiser un serveur **Express** qui vous permet de recevoir vos datas sous différents formats à l'instar des Cloud functions qui, elles, ne peuvent recevoir que du JSON.
EXEMPLE CLOUD CODE WEBHOOK] : Fig.10.

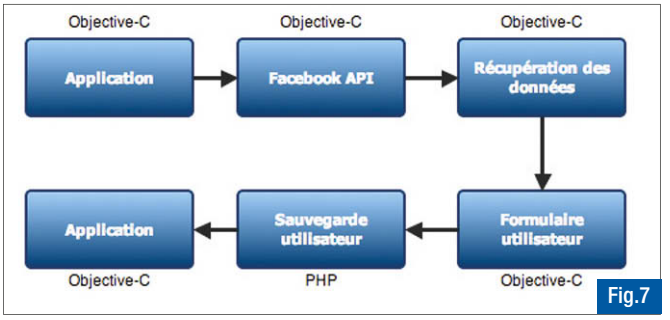


Fig.7

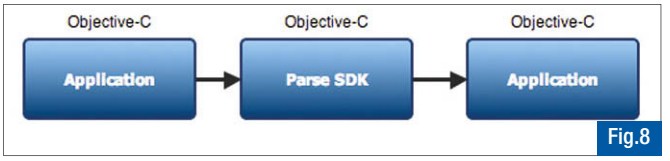


Fig.8

```
NSArray *permissionsArray = @[ @"user_about_me", @"user_relationships", @"user_birthday", @"user_location"];
[PFFacebookUtils logInWithPermissions:permissionsArray block:^(PFUser *user, NSError *error) {
    if (!user) {
        if (!error) {
            NSLog(@"Uh oh. The user cancelled the Facebook login.");
        } else {
            NSLog(@"Uh oh. An error occurred: %@", error);
        }
    } else if (user.isNew) {
        NSLog(@"User with facebook signed up and logged in!");
    } else {
        NSLog(@"User with facebook logged in!");
    }
}];
```
Fig.9

```
var express = require('express');
var app = express();

app.use(express.bodyParser());
app.post('/notify_message', function(req, res) {
        var Message = Parse.Object.extend("Message");
        var message = new Message();

        message.save({ text: req.body.text }).then(function(message) {
                res.send('Success');
        }, function(error) {
                res.status(500);
                res.send('Error');
        });
});
app.listen();
```
Fig.10

Ici nous créons simplement un objet Message avec une variable `text` que nous récupérons du body de la requête.
Les **Cloud Functions** vous permettent d'exécuter une logique serveur sans avoir à l'appliquer sur votre mobile et sans obligation de posséder une architecture machine derrière. Ou encore de diviser le code à écrire. Par exemple, si vous avez deux traitements identiques à réaliser sur vos applications, pourquoi ne pas le faire directement sur le Cloud ? Vous devez faire une moyenne mais le traitement in-app serait lourd et occuperait beaucoup de bande passante, alors utilisons le Cloud !
AVERAGESTARS PARSE CLOUD : Fig.11.
Ici nous récupérons la moyenne des notes reçues par le film dont le titre est envoyé en paramètre à la Cloud Function.
Mais ce n'est pas tout, les Cloud Functions vous offrent un éventail de fonctions disponibles pour vos objets telles que **beforeSave**, **afterSave**, **beforeDelete** et **afterDelete** qui, comme leurs noms l'indiquent, sont des fonctions exécutées avant ou après la suppression ou l'ajout d'un objet.
Attention toutefois, votre fonction doit respecter quelques conditions:

- elle doit s'exécuter en un maximum de 15 secondes.
- Les hooks beforeSave, afterSave, beforeDelete et afterDelete doivent s'exécuter en 3 secondes.
- Si une autre Cloud function est appelée dans une autre Cloud function, elle sera limitée sur le temps par rapport au temps restant de la fonction qui appelle cette dernière.

Last but not least, les **Background Jobs**. Ces derniers fonctionnent comme les Cloud functions sauf que vous en programmez l'exécution. Disons que si vous vouliez exécuter un script de migration qui vous permette d'ajouter un champ **plan** sur chaque **Parse.User**, votre code serait semblable à cela : BACKGROUND JOB USER MIGRATION : Fig.12.

Parse Push

La mise en place d'un service de push peut vite devenir une usine à gaz et requiert de passer par plusieurs plateformes:

- **iOS**: Apple Push Notification Service (APNS)
- **Android**: Google Cloud Messaging (GCM)
- **Windows Phone**: Microsoft Push Notification Service (MPNS)

```
Parse.Cloud.define("averageStars", function(request, response) {
  var query = new Parse.Query("Review");
  query.equalTo("movie", request.params.movie);
  query.find({
    success: function(results) {
      var sum = 0;
      for (var i = 0; i < results.length; ++i) {
        sum += results[i].get("stars");
      }
      response.success(sum / results.length);
    },
    error: function() {
      response.error("movie lookup failed");
    }
  });
});
```

Fig.11

```
Parse.Cloud.job("userMigration", function(request, status) {
  // Set up to modify user data
  Parse.Cloud.useMasterKey();
  var counter = 0;
  // Query for all users
  var query = new Parse.Query(Parse.User);
  query.each(function(user) {
      // Update to plan value passed in
      user.set("plan", request.params.plan);
      if (counter % 100 === 0) {
        // Set the  job's progress status
        status.message(counter + " users processed.");
      }
      counter += 1;
      return user.save();
  }).then(function() {
    // Set the job's success status
    status.success("Migration completed successfully.");
  }, function(error) {
    // Set the job's error status
    status.error("Uh oh, something went wrong.");
  });
});
```

Fig.12

Il vous faut donc prévoir:

- Un système de communication avec ces différents services
- Un dashboard pour administrer l'envoi de push et toutes les options liées à ce dernier (channels, push géolocalisé, push ciblé)
- Un serveur de données pour stocker toutes les informations liées aux pushs.
- Une architecture machine pour gérer l'envoi de ces derniers.

Chez Parse, encore une fois, on vous fait gagner du temps et de l'argent pour vos développements. INSCRIPTION DEVICE UTILISATEUR : Fig.13.
Et voilà, avec ces quelques lignes votre utilisateur est désormais inscrit dans votre base et vous pouvez d'ores et déjà lui envoyer des notifications grâce au Dashboard de Parse ! Mais ne nous arrêtons pas en si bon chemin, disons que notre utilisateur est mordu des actualités de l'**OGC Nice;** il nous suffit de l'inscrire sur le **channel** correspondant, ce qui nous permettra d'envoyer les notifications liées au club Niçois uniquement aux personnes inscrites à ce channel. OBJ-C INSCRIPTION CHANNEL PUSH : Fig.14.

Parse Analytics

Si je vous dis Analytics ? Vous me remontez forcement le mot Google avec... Mais ici nous parlons mobile !
Aujourd'hui Parse vous offre des outils faciles d'utilisation et clé en main pour remonter n'importe quel type de données utilisateur pour vous permettre d'appliquer une stratégie Marketing de qualité !
Vous voulez remonter des données démographiques, le taux de rebonds, le % de nouvelles sessions ainsi que la langue de l'utilisateur, son âge, son sexe et ses centres d'intérêts, les recherches effectuées ? Pas de problèmes, en une ligne remontez facilement tout cela :
OBJ-C CREATE PARSE EVENT : Fig.15.
Maintenant, mettez Parse Analytics et Parse Social ensemble et devenez le roi du monde, ou du moins, celui des retours de données et de la stratégie Marketing.

Parse PHP

Après le mobile, si on utilisait Parse côté serveur ? Comme vu plus haut, exit les bases de données avec Parse. Et même si vous n'êtes pas sur mobile, toutes les fonctionnalités de Parse sont disponibles avec le SDK PHP. Aujourd'hui, déployer un projet basé sur Mysql peut s'avérer être fastidieux pour la mise en place de la sécurisation des champs et les différents CRUD pour vos classes. Comme vous le savez Parse vous permet d'effectuer du relationnel One-To-One, One-To-Many, Many-To-Many : finis les ennuis avec vos foreign_keys, les deletes en cascade, les jointures de tables, avec le service de Facebook on fait des pointeurs !
Et si nous faisions le tour des fonctionnalités disponibles sur le SDK PHP ? SAUVEGARDE OBJET PHP : Fig.16.

```
PFInstallation *currentInstallation = [PFInstallation currentInstallation];
[currentInstallation setDeviceTokenFromData:deviceToken];
[currentInstallation saveInBackground];
```

Fig.13

```
PFInstallation *currentInstallation = [PFInstallation currentInstallation];
[currentInstallation addUniqueObject:@"OGCN" forKey:@"channels"];
[currentInstallation saveInBackground];
```

Fig.14

```
[PFAnalytics trackEvent:@"Enregistrement" dimensions:@{ @"sexe": @"f", @"age": @"22", @"localisation": @"France" }];
```

Fig.15

```
$ship = ParseObject::create("Ship");
$ship->setName('Enterprise');
$ship->setLength(342);
$ship->save();
```

Fig.16

Nous avons aussi la notion de **Subclass**, il s'agit de faire hériter une classe par **ParseObject** pour en faire une classe Parse.
PHP SUBCLASS: Fig.17.
Comme nous somme côté serveur, peut être que vous voulez remonter des informations (events). Encore une fois, facile !
PHP TRACK EVENTS : Fig.18.
Et voilà, vous avez désormais un évènement généré pour chaque appel à cette fonction.
Parse vous offre également le moyen de stocker des fichiers dans sa base de données tout en vous donnant un lien pour télécharger ce dernier. PHP CREATE FILE PARSE : Fig.19.
Vous pouvez également charger un fichier local pour l'envoyer sur Parse avec cette fonction : PHP LOAD LOCAL FILE.
Et si nous envoyions des pushs à nos applications mobiles ?
PHP SEND PUSH : Fig.20.
Nous avons donc envoyé l'alerte "Victoire du ..." aux personnes abonnées aux channels RCT et LAR.
Nous avions vu plus haut que nous pouvions exporter des traitements sur le Cloud, sachez qu'il est également possible d'appeler une Cloud Function avec le SDK PHP : PHP CLOUD FUNCTION : Fig.21.
Finalement nous allons voir les ACL (Access Control List). Derrière ce nom barbare se cache un système de sécurisation de vos données qui permet de définir des droits sur vos objets Parse ou Utilisateurs.
Par défaut les utilisateurs ont déjà une sécurité qui empêche toutes modifications par un autre utilisateur (ParseUser). Les objets (ParseObject) ont, eux, besoin qu'on leur définisse les droits globaux ou pour chaque utilisateur.
PHP PARSE ACL PUBLIC ACCESS : Fig.22. Ici l'objet Message sera disponible uniquement en lecture; si vous essayez de le modifier vous recevrez une erreur ParseError.OBJECT_NOT_FOUND.
PHP ACL FROM USER : Fig.23.
Ici notre objet Message sera disponible en lecture et écriture pour l'utilisateur actuellement connecté (renvoyé par ParseUser::getCurrentUser), et en lecture pour tous les autres utilisateurs.
Quand votre application devient de plus en plus grosse, il va devenir difficile de maintenir les droits sur chacun des objets pour chaque utilisateur; Parse l'a bien compris et vous permet de définir des **Rôles** qui vont englober plusieurs utilisateurs, ces même utilisateurs vont eux hériter des droits qui seront définis sur le rôle auquel ils appartiennent.
Un rôle peut contenir des utilisateurs mais aussi d'autre rôles, sur le même principe que les utilisateurs, les rôles contenus dans le rôle parent vont hériter des droits du parent.
PHP PARSE CREATE ROLE : Fig.24.
Ici nous avons créé un rôle avec des droits uniquement en lecture sur ce dernier à l'exception pour l'utilisateur courant qui lui a aussi les droits en écriture. Sachez que si vous ne mettez pas le rôle avec des droits d'écriture limités, n'importe quel utilisateur pourra modifier ce dernier.
Et si nous donnions à ce rôle un droit d'écriture sur l'un de nos objets ?
PHP PARSE OBJECT ROLE ACL : Fig.25. Et voilà, nous disposons maintenant d'un objet Message auquel seuls les membres du rôle Administrateur ont les droits d'écriture tandis que les autres n'ont, eux, que les droits de lecture.

Pour en terminer

Sesame Street, Emporio Armani, Cisco, Swisscom, Zynga, Yo, ces noms vous disent peut être quelque chose ? Un point commun entre tous: ils utilisent Parse ! Le service de Facebook est en phase de devenir le plus performant fournisseur de back-end en plus d'avoir la meilleure documentation disponible actuellement. Rapide, simple d'utilisation et gratuit, il vous fera gagner un temps de développement conséquent pour vous permettre de vous focaliser sur le plus important: **votre application**. Chez Studio Ymagyn, nous avons déjà emboîté le pas et nous n'utilisons plus que Parse pour nos applications.
Vive le mobile, vive les apps, vive Parse !

```php
class Ship extends ParseObject
{
        public static $parseClassName = "Ship";

        public static function build($name, $length)
        {
                $ship = new Ship();
                $ship->set("name", $name);
                $ship->set("length", $length);

                return $ship;
        }

        public function isBig()
        {
                return $this->get('length') > 300;
        }
}
// A appeler avant ParseClient::initialize( $app_id, $rest_key, $master_key );
Ship::registerSubclass();
// [...]
$ship = Ship::build('Enterprise', 342);
$ship->save();
$ship->isBig(); // Return true
```
Fig.17

```php
ParseAnalytics::track("Visite", array("livre" => "The Hitchhiker's guide to the Galaxy"));
```
Fig.18

```php
$contents = "Hello World.";
$file = ParseFile::createFromData($contents, "myfile.txt");
$file->save();
$url = $file->getURL();
```
Fig.19

```php
ParsePush::send(array(
    "channels" => ["RCT", "LAR"],
    "data" => array("alert" => "Victoire du RCToulon contre La Rochelle 60 à 19 !")
));
```
Fig.20

```php
$results = ParseCloud::run("aCloudFunction", array("from" => "php"));
```
Fig.21

```php
$message = new ParseObject('Message');
$acl = new ParseACL();
$acl->setPublicReadAccess(true);
$message->setACL($acl);
$message->save();
```
Fig.22

```php
$message = new ParseObject('Message');
$acl = ParseACL::createACLWithUser(ParseUser::getCurrentUser());
$acl->setPublicReadAccess(true);
$message->setACL($acl);
$message->save();
```
Fig.23

```php
$roleACL = new ParseACL();
$roleACL->setPublicReadAccess(true);
$roleACL->setWriteAccess(ParseUser::getCurrentUser(), true);
$role = ParseRole::createRole("Administrateur", $roleACL);
$role->save();
```
Fig.24

```php
$role = ParseRole::query()->equalTo('name', "Administrateur")->first();
$message = new ParseObject("Message");
$acl = new ParseACL();
$acl->setPublicReadAccess(true);
$acl->setRoleWriteAccess($role, true);
$message->setACL($acl);
$message->save();
```
Fig.25

Partager du contenu entre applications Drupal

De nos jours, les entreprises ne voient plus leur site Internet comme une application isolée mais comme une pièce centrale du puzzle qu'est leur système d'informations. De nombreux modules Drupal sont disponibles sur Drupal.org afin de rendre Drupal interopérable avec tous les outils majeurs (analytics, marketing, CRM, ...), qu'ils soient open source ou propriétaires.

Raphaël Khaïat
Solution Architect chez Acquia.

Un exemple moins courant est d'intégrer un site Drupal à un autre. Il existe de nombreux cas d'utilisation pour une telle solution (environnement de staging(1), partage de contenu, headless Drupal(2), etc). Le but de cet article est de vous montrer étape par étape comment partager du contenu entre sites Drupal via de simples appels Web services.

Installation rapide

Tout d'abord, vous avez besoin de 2 sites Drupal installés et opérationnels. Un site "local" sur lequel vous créerez du contenu et un site "distant" qui "recevra" ce contenu. Le plus rapide est d'utiliser **drush(3)** :

```
drush dl Drupal-7.x
drush site-install standard --account-name=admin --account-pass
=password --db-url=mysql://mysqlUser:mysqlPass@localhost/MysqlDb
```

CRÉEZ VOTRE PREMIER WEB SERVICE REST

Module "Services", premier round

Installer via drush le module services sur le site distant.

```
drush dl services uuid
drush en -y services uuid rest_server uuid_services
```

Sur la page du module Services(4), il est inscrit: «[Services is] a standardized solution of integrating external applications with Drupal [...] ». En d'autre terme, il va vous permettre de créer votre propre API.

Configuration

La documentation est peu fournie sur Drupal.org(5) mais la documentation de l'API est consultable sur drupanium(6). Pour configurer le module, allez sur la page `admin/structure/services` et cliquez sur "add" pour créer votre Web service. Donnez-lui un nom et choisissez "REST" pour le «Server». Un "endpoint" est le "chemin" vers votre Web service. Pour **www.domain.tld/api/v1/<actions>**, mettez «**api/v1/**». Pensez à activer le mode debug, il peut toujours vous être utile :). Votre configuration doit ressembler à la figure 1. Sur la page d'accueil du module, utilisez la liste déroulante de votre service et cliquez sur «edit server». Sélectionnez «json» comme «Response formatters» et «application/json» comme «Request Parsing» (voir figure 2). Continuer en cliquant sur l'onglet «resources» afin d'activer certains Web services inclus par défaut. Cliquez sur la flèche à côté de "node" et cochez «retrieve». Ce service va vous permettre de récupérer vos nodes distants via leur UUID (figure 3). Et voilà, vous avez maintenant un Web service activé et configuré !

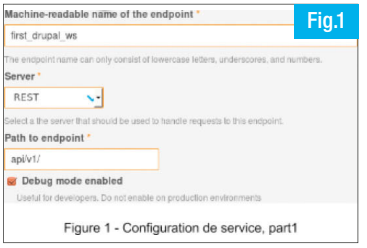


Figure 1 - Configuration de service, part1

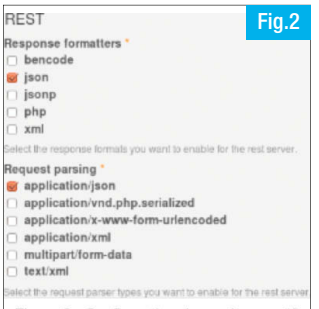


Figure 2 - Configuration de service - part2 le server

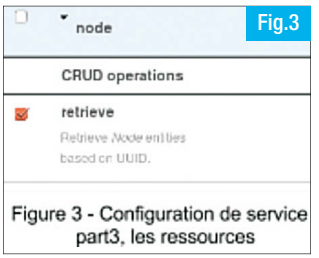


Figure 3 - Configuration de service part3, les ressources

Récupérer votre premier node

Vérifiez d'abord que la configuration est correcte. Pour cela, 2 méthodes : Il existe de nombreux clients HTTP php permettant de consommer des services REST, comme par exemple la fonction (très limitée) de Drupal, `Drupal_http_request`. J'ai choisi d'utiliser GuzzleHttp(7) pour sa flexibilité (il s'agit presque d'un "Framework" pour vos appels http) mais aussi parce qu'il est inclus par défaut dans Drupal 8. Dans cet article, nous n'utiliserons que cette librairie. Tout d'abord, installez la librairie dans un répertoire de test :

```
php composer require guzzlehttp/guzzle:~4
```

Ensuite, créez un fichier index.php avec le code suivant :

```
<?php
require 'vendor/autoload.php';
use GuzzleHttp\Client;

$url = "http://www.domain.tld/api/v1";

$client = new Client();
$response = $client->get($url);
if (200 == $response->getStatusCode()) {
  echo $response->getBody();
}
```

Changez la variable `$url` puis exécutez le script via la commande «`php index.php`». Vous devez obtenir :

```
Services Endpoint "first_Drupal_ws" has been setup successfully.
```

En un mot, ce script va effectuer une requête de type GET sur l'url de base de votre API. Le module services lui répond que le Web service (appelé ici "first_Drupal_ws") est correctement configuré. Pour plus d'informations, consultez la documentation de Guzzle(8). Bon, notre Web service fonctionne, essayons maintenant de récupérer du contenu. Créez tout d'abord un node "simple" sur votre site "distant" (celui où est configuré service). **Publiez-le afin qu'il puisse être accessible anonymement!** Récupérez

(1) https://groups.Drupal.org/large-scale-Drupal-lsd-projects-and-plans/content-staging
(2) https://groups.Drupal.org/headless-Drupal
(3) http://drush.ws
(4) https://www.Drupal.org/project/services
(5) https://www.Drupal.org/documentation/modules/services
(6) http://drupanium.org/api
(7) http://guzzlephp.org/
(8) http://guzzle.readthedocs.org/en/latest/

l'UUID du node (via devel en accédant à l'url `node/<nid>/devel` ou via la bdd dans la table node). Editez le fichier `index.php` et changez l'url en lui ajoutant `/node/<uuid>`. Par exemple : http://www.domain.tld/api/v1/node/4f8862b1-4bc7-4948-94b2-c3aeae9dabab

Exécutez à nouveau ce code. Vous devriez recevoir du json (comme configuré précédemment) représentant le node. Changez à nouveau le fichier index.php en changeant `echo $response->getBody();` par `print_r($response->json());` et relancez le script. Vous avez maintenant votre node représenté sous forme de tableau.

Bravo! Vous venez de créer et configurer un Web service avec Drupal (et le module services) afin de récupérer des nodes publiés. La prochaine étape est de pouvoir récupérer des nodes non publiés :-).

Requête authentifiée

Pour récupérer des contenus (entités, files, taxonomies, ...) non publiés ou pour modifier des contenus, il vous faudra soit autoriser ces actions pour les utilisateurs anonymes (par pitié, non!) ou authentifier vos requêtes pour sécuriser votre application. Plusieurs méthodes existent telles que l'authentification "classique" par login / password, l'authentification via OAuth (2-legged-oauth ou 3-legged-oauth), etc. Dans cet article, nous allons utiliser OAuth. Je déconseille l'utilisation simple de login/mot de passe vu qu'il vous faudra stocker ces informations en clair. Si vous souhaitez tout de même essayer, il vous faudra regarder du côté de l'utilisation de cookie via Guzzle(9). Un exemple est disponible sur le repo git de cet article.

Le protocole OAuth(10)

Pourquoi ?

Pour paraphraser wikipedia(11), «OAuth est un protocole libre [... permettant] d'autoriser un site Web à utiliser l'API sécurisée d'un autre site Web pour le compte d'un utilisateur.». Deux versions de ce protocole existent. GuzzleHttp ne supporte pas encore la version 2 mais une "pull request" est en attente sur github au moment où j'écris cet article(12). Dans notre cas, il s'agit simplement de transmettre des données d'un client à un serveur, c'est pourquoi nous utiliserons la méthode dite "2 legged OAuth" (pas d'action nécessaire par utilisateur). Je vous conseille de lire le détail de ces différentes solutions :).

Installation

Télécharger et installer les modules nécessaires :

```
drush dl oauth
drush en -y oauth_common services_oauth oauth_common_providerui
```

Module services, 2ème round

Retournez sur la page de configuration du service (`/admin/structure/services/list/<webservice_name>`). Une nouvelle option est disponible pour l'authentification : «OAuth authentication». Cocher cette case et sauvegarder. Il vous faut maintenant configurer le fournisseur OAuth vous rendant sur `admin/config/services/oauth`. Sélectionner l'option «Enable the oauth provider» et laissez les autres valeurs qui ne concernent que la méthode "3 legged OAuth". Ensuite, vous devez créer un "contexte" OAuth avec les autorisations nécessaires. Chaque méthode de notre Web service peut être associée à un niveau d'autorisation différent. Choisissez un nom machine, un titre et sélectionnez **uniquement** HMAC-SHA1(13) pour la signature. Ajoutez un niveau d'autorisation et sélectionnez l'option «Selected by default». Les champs dans la partie «Authorization options» sont facultatifs dans notre cas.

Sur l'onglet «authentication» (`admin/structure/services/list/<webservice_name>/authentication`), sélectionnez le contexte que vous venez de créer ainsi que «**Consumer key, also known as 2-legged OAuth**» pour le «Default required authentication». Sauvegardez, puis choisissez le niveau d'autorisation que vous avez créé pour ce contexte (voir figure 5). Pour finir, il vous faut configurer la ressource de votre Web services afin de la sécuriser par OAuth. Allez sur l'onglet «resources» et sélectionnez pour l'action «node -> retrieve» «Consumer key» pour «required authentication» et laissez «default» pour le niveau d'autorisation (voir figure 6).

Rôles et permissions

Créez un utilisateur dédié à vos appels Web service et attribuez lui un rôle lui permettant de créer et modifier des nodes. Les prochaines requêtes seront faites via OAuth comme étant faites par cet utilisateur. Si votre utilisateur n'a pas les bonnes permissions, vous recevrez un code retour *503*. Une fois votre utilisateur créé avec les bons rôles/permissions, cliquez sur l'onglet «Oauth consumers» de sa page de profil (ou à l'url `user/<uid>/oauth/consumer`). Cliquez sur "Add consumer", donnez un nom et sélectionnez votre context. Laissez le champ «callback url» vide (utile que pour la méthode "3 legged OAuth"). Editez ce "comsumer" pour récupérer la clé (**key**) et le jeton secret (**token**) que nous utiliserons dans notre prochain code.

Requête sécurisée via OAuth

Maintenant que la configuration du module services est complète, le précédent script devrait vous afficher l'erreur *401* suivante :

```
[status code] 401 [reason phrase] Unauthorized : The request must be signed'
```

Avant de modifier votre code, il vous faut installer GuzzleHttp OAuth subscriber(14) via composer :

```
composer require gul guzzlehttp/oauth-subscriber:0.1.*
```

Modifiez votre fichier index.php comme suit :

```
<?php
require 'vendor/autoload.php';
use GuzzleHttp\Client;
use GuzzleHttp\Subscriber\Oauth\Oauth1;
```

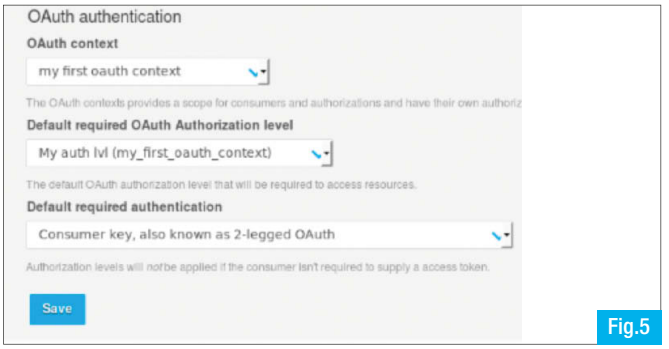


Fig.5

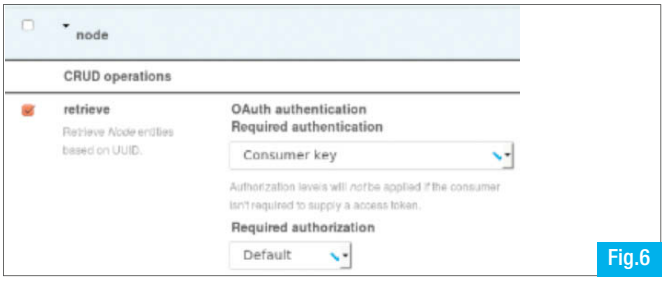


Fig.6

(9) http://guzzle.readthedocs.org/en/latest/clients.html?highlight=cookie#cookies
(10) http://oauth.net/
(11) https://fr.wikipedia.org/wiki/OAuth
(12) https://github.com/guzzle/oauth-subscriber/pull/20
(13) http://oauth.net/core/1.0/#anchor16
(14) https://github.com/guzzle/oauth-subscriber

```
$uuid = '';
$oauth_creds = array('consumer_key' => '', 'consumer_secret' => '');

$url = 'node/' . $uuid;
$client = new Client(array(
  'base_url' => 'http://www.domain.tld/api/v1/',
  'defaults' => array(
 'auth' => 'oauth',
 'debug' => true,
  )
));
$oauth = new Oauth1(array(
  'consumer_key' => $oauth_creds['consumer_key'],
  'consumer_secret' => $oauth_creds['consumer_secret'],
));
$client->getEmitter()->attach($oauth);
$response = $client->get($url);
if (200 == $response->getStatusCode()) {
  print_r($response->json());
}
```

N'oubliez pas de changer les variables `$uuid` et `$oauth_creds`. "Consumer_key" et "consumer_secret" sont respectivement la clé et le jeton récupérés précédemment. Examinons ce nouveau code. L'instanciation de notre client n'est plus la même. Nous lui passons un tableau de configuration avec une "base_url" (afin de ne pas la répéter en cas de requête multiple) ainsi qu'une configuration par défaut pour toutes nos requêtes : mode debug activé et authentification par OAuth.
Ensuite nous créons avec la clé et le token une instance de `GuzzleHttp\Subscriber\Oauth\Oauth1` que nous attachons à notre client afin qu'il puisse modifier les requêtes Guzzle pour intégrer les headers OAuth. Et c'est tout :). Lancer de nouveau le script et vous devriez recevoir le node complet à nouveau. Si vos logs sont activés (`admin/reports/dblog` ou via syslog), vous pouvez voir que services a identifié la requête comme provenant de votre utilisateur grâce à OAuth.

PARTAGEZ DU CONTENU ENTRE DRUPAL

Vous savez maintenant lancer des requêtes authentifiées via OAuth avec Guzzle. Vous saurez bientôt créer votre premier node "à distance". Mais d'abord, il faut configurer une dernière fois le module services.

Module services, dernier round

Dernière modification de configuration du Drupal "distant". Éditez les ressources de votre Web service et ajoutez les actions suivantes: "node CRUD operations -> update". Si vous souhaitez attacher des fichiers à votre node, il vous faudra aussi sélectionner "attach_file" dans "node -> targeted actions" (figure 7). Sur l'onglet "server", cochez la case «`application/x-www-form-urlencoded`» dans la partie «request_parsing» afin que votre Web service accepte les headers pour la soumission de formulaire(15). Votre site Drupal "distant" est prêt :)!

Configuration du 2ème Drupal

C'est l'heure de configurer notre 2ème site Drupal qui prenait la poussière :). Cette configuration est beaucoup plus simple que la précédente. Ensuite, il vous faudra importer le code dans Drupal.

Installation des modules

Pour utiliser GuzzleHttp dans Drupal, le plus simple est d'utiliser le module Drupal composer_manager(16) :

```
drush dl uuid composer_manager
```

Maintenant, vous allez créer un module contenant votre code ainsi qu'un fichier composer.json. Composer_manager cherche à la racine de chaque module l'existence d'un fichier composer.json et se charge d'installer toutes les librairies nécessaires en gérant les numéros de versions et les dépendances dans `sites/all/vendor`. Commençons par créer les fichiers

```
mkdir -p sites/all/modules/custom/remote_publish
touch sites/all/modules/custom/remote_publish/remote_publish.
{info,module}
touch sites/all/modules/custom/remote_publish/composer.json
```

Éditer le fichier `composer.json` avec le contenu suivant :

```
{
  "require": {
    "guzzlehttp/guzzle": "*",
    "guzzlehttp/oauth-subscriber": "*"
  }
}
```

Éditez le fichier `remote_publish.info` et indiquer composer_manager, entity, uuid comme dépendances.
Ensuite, utilisez drush pour installer votre module et ses dépendances :

```
drush en -y remote_publish
```

Il est important d'utiliser drush pour que composer_manager s'exécute correctement. Vérifiez que vos librairies sont présentes dans `sites/all/vendor`.

Utilisez Guzzle dans Drupal

Ok, tout est installé, "yapluka" comme on dit! Ouvrez le fichier `remote_publish.module` et placez la fonction ci-dessous :

```
function _remote_publish($uuid) {
  $oauth_creds = array('consumer_key' => '', 'consumer
_secret' => '');
  $url = 'node/' . $uuid;
  $client = new \GuzzleHttp\Client(array(
    'base_url' => 'http://www.domain.tld/api/v1/',
    'defaults' => array(
      'auth' => 'oauth',
      'debug' => true,
```

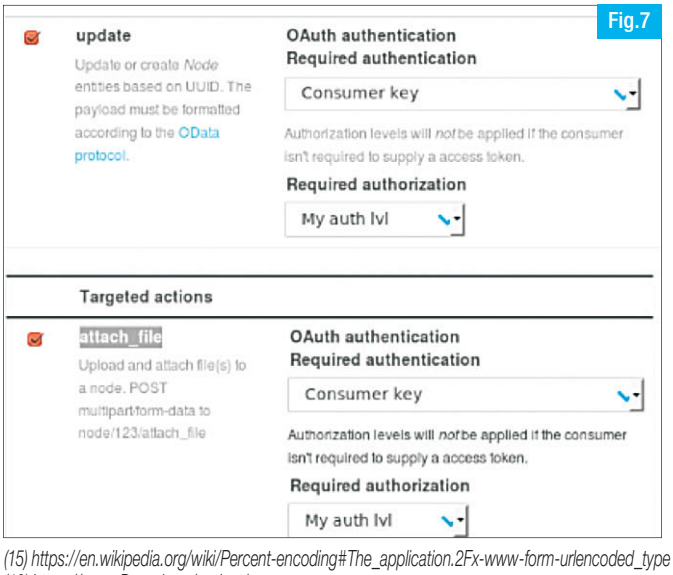


Fig.7

(15) https://en.wikipedia.org/wiki/Percent-encoding#The_application.2Fx-www-form-urlencoded_type
(16) https://www.Drupal.org/project/composer_manager

```
    )
  ));
  $oauth = new \GuzzleHttp\Subscriber\Oauth\Oauth1(array(
 'consumer_key' => $oauth_creds['consumer_key'],
 'consumer_secret' => $oauth_creds['consumer_secret'],
  ));
  $client->getEmitter()->attach($oauth);
  $response = $client->get($url);
  if (200 == $response->getStatusCode()) {
 return $response->json();
  }
}
```

Le même code que précédemment mais placé dans une fonction dans un module Drupal. La seule différence est que j'instancie l'objet avec son nom de classe complet. Vous pouvez tester votre fonction via drush: `drush php-eval "print_r(_remote_publish(<UUID>));"` Remplacer <UUID> par un uuid existant sur votre Drupal "distant" et validez! Si tout se passe bien, vous recevrez un tableau représentant le node sur le Drupal distant :). Tout est enfin en place, ne vous inquiétez pas, vous y êtes presque!

Partager du contenu, enfin!

Créez un node via un appel Web service

Pour le moment, vous avez juste intégré le code du fichier index.php dans un module Drupal. C'est l'heure d'"envoyer" un node d'un Drupal à un autre. Mais avant, soyez sûr d'avoir au moins un node à partager et récupérez son uuid comme vu plus haut. Ensuite, modifiez la fonction `_remote_publish` comme suit :

```
function _remote_publish($uuid = null) {
  if (!$uuid) {
    return FALSE;
  }
  $entities = entity_uuid_load('node', array($uuid));
  // Reset return the first element if it exists.
  $entity = (array)reset($entities);

  $oauth_creds = array('consumer_key' => '', 'consumer_
secret' => '');
  $url = 'node/' . $uuid;
  $client = new \GuzzleHttp\Client(array(
 'base_url' => 'http://www.domain.tld/api/v1',
 'defaults' => array(
     'auth' => 'oauth',
     'debug' => TRUE,
      )
  ));
  $oauth = new \GuzzleHttp\Subscriber\Oauth\Oauth1(array(
 'consumer_key' => $oauth_creds['consumer_key'],
 'consumer_secret' => $oauth_creds['consumer_secret'],
  ));
  $client->getEmitter()->attach($oauth);

  $response = FALSE;
  try {
    $response = $client->put(
      $url,
   array('json' => $entity)
    );
  }
  catch(\Exception $e) {
 drush_log($e->getMessage(), 'error');
  }
  if ($response && 200 == $response->getStatusCode()) {
 return $response->json();
  }

  return FALSE;
}
```

Remarquez l'utilisation de la méthode PUT à la place de POST contrairement à la documentation. Cela à cause du module `uuid_service` qui modifie la ressource de services (via le `hook_services_resources_alter`) afin de permettre la création / modification d'entité en une seule méthode (PUT node/UUID). Pour plus d'informations, ouvrez le fichier `uuid_services.module`. En outre, le 2nd argument permet de préciser que les données seront envoyées au format json (cf la configuration du service). Le node chargé par Drupal (via `entity_uuid_load`) est donc simplement converti au format JSON. Rien de très compliqué donc :).

Exécutez à nouveau la commande `drush php-eval…` en passant l'UUID du node local cette fois ci. Si tout va bien, vous recevrez en retour un tableau le représentant sur le site distant. Simple, n'est-ce pas :) ?

Troubleshooting :

Voici une liste rapide d'erreurs communes que vous pourriez rencontrer durant vos tests :

[status code] 403 [reason phrase] :Access denied for user oauth user: Cette erreur indique que vous n'avez pas donné les bonnes permissions à votre utilisateur OAuth.

[status code] 406 [reason phrase] Not Acceptable: Unsupported request content type application/x-www-form-urlencoded: Cela signifie que vous n'avez pas activé l'option «x-www-form-urlencoded» pour le serveur de votre service. *[status code] 404 [reason phrase] Not found: Could not find the controller*: Services indique ici qu'il n'a pas pu trouver un contrôleur pour votre Web services. La raison la plus problable est que le nombre d'arguments reçu n'est pas bon, ou que vous n'utilisez pas le bon verbe (ex: POST au lieu de PUT).

Remarques sur les références et les assets

L'exemple ci-dessus est très simple et la plupart du temps, vos types de contenus seront plus compliqués avec des références à des termes de taxonomie ou des entités, ou bien des assets (fichiers, images) attachés. Le code tel que montré ne fonctionnera pas avec ces types de champs.

Conclusion

Dans cet article, vous avez appris à configurer le module services afin de créer des Web services avec une authentification via OAuth. Le contenu partagé était assez simple et vous pouvez imaginer que dans des exemples complexes cela peut devenir beaucoup plus compliqué. En outre, nous n'avons pas intégré ce code dans un vrai workflow Drupal (via hook à la sauvegarde du node, via rules, etc.) mais au moins la configuration de services est clair (du moins j'espère :)). Il ne vous reste plus qu'à coder pour faire de puissantes API :). Tous les morceaux de code présentés dans cet article sont disponibles sur ma page github(17).

(17) *https://github.com/bacardi55/remote-publishing-article*

Créer une application Bootstrap avec le mkframework

Vous avez pu découvrir dans PROGRAMMEZ! (167, 170 et 173) un framework différent. Dans ce tutoriel, je vous propose de créer votre première application Bootstrap en toute simplicité avec ce même framework. Bootstrap est un framework css permettant d'avoir une application "responsive". Ce terme signifie que le site va s'adapter à l'écran qui le consulte. Par exemple en mode "smartphone", le menu se minimise et affiche un bouton "burger" pour l'afficher.

Michaël Bertocchi
@dupot_org

Télécharger et installer le framework

Rendez-vous à l'adresse http://mkframework.com
Copiez ce répertoire zip dans votre répertoire Web, et désarchivez-le
Ouvrez votre navigateur sur http://localhost/mkframework_v4_XX_YY_rZZZ
en fonction de la version téléchargée

Créer la base de données

Dans votre base de données MySQL, nous allons créer une table "contact". Executez la requête SQL suivante :

```
CREATE TABLE `note` (
`id` int(11) NOT NULL auto_increment,
`text` varchar(50) NOT NULL,
`user_id` int(11) NOT NULL,
PRIMARY KEY  (`id`)
);
CREATE TABLE `member` (
`id` int(11) NOT NULL auto_increment,
`login` varchar(50) NOT NULL,
`password` varchar(50) NOT NULL,
PRIMARY KEY  (`id`)
);
```

Créons l'application

Ouvrez le navigateur à l'adresse du framework Fig.1. Entrez le champ de formulaire le nom de votre application, sélectionnez "Application compatible "bootstrap" "Le builder vous crée une application dans le répertoire data/genere du framework et vous redirige ensuite sur la page d'administration de celle-ci.

Modifiez les paramètres de connexion

Cliquez sur le bouton "explorer le projet", et ouvrez le répertoire conf/
Cliquez sur le fichier connexion.ini.php, puis sur le bouton "EDITER EN ENTIER" pour modifier vos profils de connexion.

```
Inscrivez votre profil mysql
;<?php die()?>
[db]
bootstrapDb.dsn="mysql:dbname=bootDb;host=localhost"
bootstrapDb.sgbd=pdo_mysql
bootstrapDb.username=root
bootstrapDb.password=root
```

Générer la couche modèle

Le framework utilise une structure MVC, il nécessite donc des classes pour interagir avec la base de données. Le builder vous propose de les générer automatiquement à partir de votre fichier de profils de connexion. Cliquez sur "créer la couche modèle". Vous voyez listés les profils de votre site, ici "bootstrapDb" (précédemment renseigné). Cliquez dessus. Le builder vous liste les tables visibles via ce profil, dans notre cas les tables note et member précédemment créées. Validez avec le bouton "générer". Il vous crée alors les fichiers model/model_note.php et model/model_member.php qui contiennent les classes model_note/row_note et model_member/row_member.
Un exemple de classe model :

```
<?php
class model_note extends abstract_model{

  protected $sClassRow='row_note';

  protected $sTable='note';
  protected $sConfig='bootstrapDb';

  protected $tId=array('id');

  public static function getInstance(){
    return self::_getInstance(__CLASS__);
  }

  public function findById($uId){
    return $this->findOne('SELECT * FROM '.$this->sTable.' WHERE
 id=?',$uId );
  }
  public function findAll(){
    return $this->findMany('SELECT * FROM '.$this->sTable);
  }


}
```

Un exemple de classe row :

```
<?php
class row_note extends abstract_row{

  protected $sClassModel='model_note';
```

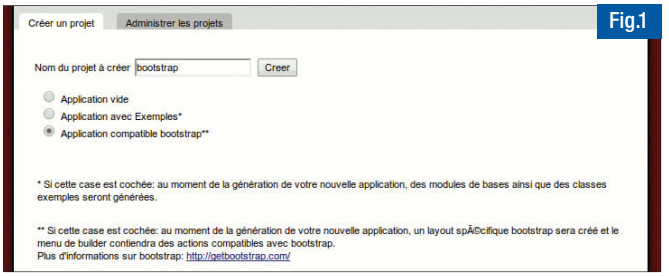


Fig.1

```
  /*exemple jointure
  public function findAuteur(){
    return model_auteur::getInstance()->findById($this->auteur_id);
  }
  */
  /*exemple test validation*/
  private function getCheck(){
    $oPluginValid=new plugin_valid($this->getTab());
    /* renseigner vos check ici*/
    $oPluginValid->isNotEmpty('text','Le champ ne doit pas
&ecirc;tre vide');

    return $oPluginValid;
  }

  public function isValid(){
    return $this->getCheck()->isValid();
  }
  public function getListError(){
    return $this->getCheck()->getListError();
  }
  public function save(){
    if(!$this->isValid()){
      return false;
    }
    parent::save();
    return true;
  }

}
```

Générer le module CRUD de la table note

CRUD signifie Create Read Update Delete. Ceci désigne les pages nécessaires pour administrer une table: un tableau listant les enregistrements, ainsi que les pages d'affichage, d'ajout de modification et de suppression. Cliquez sur "créer un module CRUD bootstrap".
Le builder vous liste les classes modèles disponibles : Fig.2.
Cliquez sur la classe "model_note.php". L'interface de génération de CRUD vous propose de nombreuses options pour vous faire gagner du temps. Validez avec le bouton "créer", le builder créera un module "note". Note: lorsque le builder génère le CRUD, il affiche un lien permettant de voir le module généré. Voici à quoi ressemble en partie le module note :

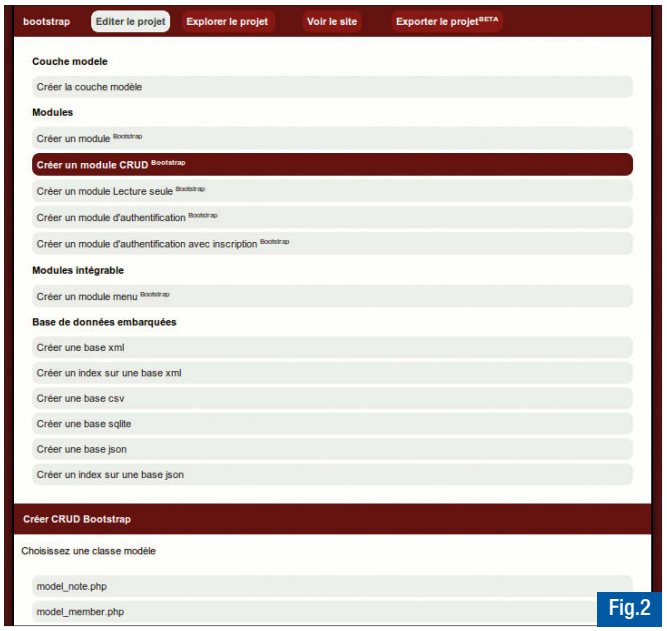


Fig.2

```
<?php
class module_note extends abstract_module{

  public function before(){
    $this->oLayout=new _layout('bootstrap');

    //assignez le menu à l'emplacement menu
    $this->oLayout->addModule('menu','menu::index');
  }
  public function _index(){
      //on considère que la page par défaut est la page de listage
       $this->_list();
  }
  public function _list(){

    $tNote=model_note::getInstance()->findAll();

 //création d'un objet vue
    $oView=new _view('note::list');
     //assignation du tableau des notes à la vue
    $oView->tNote=$tNote;

     //on envoie l'objet vue à l'emplacement "main" du layout
    $this->oLayout->add('main',$oView);

  }
  public function _new(){
    $tMessage=$this->processSave();

    $oNote=new row_note;
    $oNote=$this->fillRow($oNote);

    $oView=new _view('note::new');
    $oView->oNote=$oNote;

    $oPluginXsrf=new plugin_xsrf();
    $oView->token=$oPluginXsrf->getToken();
    $oView->tMessage=$tMessage;

    $this->oLayout->add('main',$oView);
  }
  public function _edit(){
    $tMessage=$this->processSave();
```

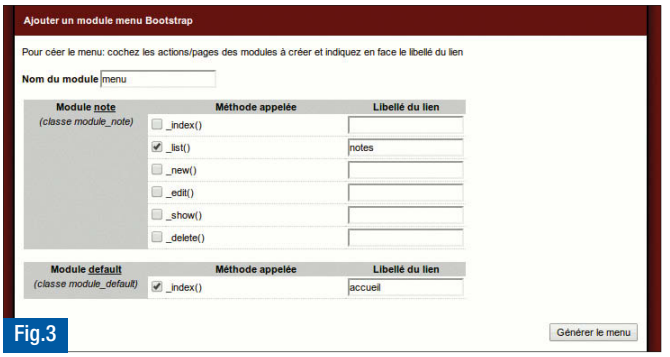


Fig.3

```
    $oNote=model_note::getInstance()->findById( _root::getParam('id') );
    $oNote=$this->fillRow($oNote);

    $oView=new _view('note::edit');
    $oView->oNote=$oNote;
    $oView->tId=model_note::getInstance()->getIdTab();

    $oPluginXsrf=new plugin_xsrf();
    $oView->token=$oPluginXsrf->getToken();
    $oView->tMessage=$tMessage;

    $this->oLayout->add('main',$oView);
  }
  private function fillRow($oNote){
    //si ce n'est pas une requête POST on ne soumet pas
    if(!_root::getRequest()->isPost() ){
      return $oNote;
    }
    //on remplit ici l'objet avec les paramètres du formulaire
 s'il est envoyé
    //pour ne pas perdre la saisie
    $tId=model_note::getInstance()->getIdTab();
    $tColumn=model_note::getInstance()->getListColumn();
    foreach($tColumn as $sColumn){
      if( _root::getParam($sColumn,null) === null ){
        continue;
      }else if( in_array($sColumn,$tId)){
          continue;
      }

      $oNote->$sColumn=_root::getParam($sColumn,null) ;
    }
    return $oNote;
  }
private function processSave(){
  //si ce n'est pas une requête POST on ne soumet pas
  if(!_root::getRequest()->isPost() ){
    return null;
    }

  $oPluginXsrf=new plugin_xsrf();
    //on vérifie que le token est valide
    if(!$oPluginXsrf->checkToken( _root::getParam('token') ) ){
      return array('token'=>$oPluginXsrf->getMessage() );
    }

    $iId=_root::getParam('id',null);
    if($iId==null){
      $oNote=new row_note;
    }else{
      $oNote=model_note::getInstance()->findById( _root::get
```

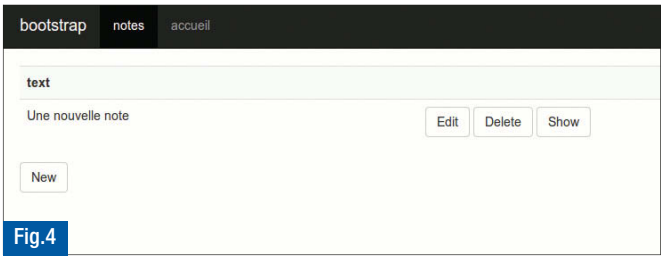


Fig.4

```
Param('id',null) );
    }

    $tId=model_note::getInstance()->getIdTab();
    $tColumn=array('text');
    foreach($tColumn as $sColumn){
       $oPluginUpload=new plugin_upload($sColumn);
      if($oPluginUpload->isValid()){
        $sNewFileName=_root::getConfigVar('path.upload')
$sNewFileName.=$sColumn.'_'.date('Ymdhis');

        $oPluginUpload->saveAs($sNewFileName);
        $oNote->$sColumn=$oPluginUpload->getPath();
        continue;
      }else  if( _root::getParam($sColumn,null) === null ){
        continue;
      }else if( in_array($sColumn,$tId)){
         continue;
      }

      $oNote->$sColumn=_root::getParam($sColumn,null) ;
    }

    if($oNote->save()){
      //une fois enregistré on redirige (vers la page liste)
      _root::redirect('note::list');
    }else{
      return $oNote->getListError();
    }

  }
```

Créons un menu à notre application

Le builder vous permet également simplement de générer un module menu pour vous. Cliquez sur “créer un module menu bootstrap” Fig.3. Vous voyez listée ici la liste des modules/actions disponibles dans votre application. Validez avec le bouton “Générer le menu”.
Le builder vous indique qu'il a bien généré le module menu et vous affiche le code pour l'inclure dans vos modules. Copiez ce code dans le module note, fichier module/note/main.php (en fin de méthode before)

```
<?php
class module_note extends abstract_module{

  public function before(){
    $this->oLayout=new _layout('bootstrap');

    //assignez le menu a l'emplacement menu
    $this->oLayout->addModule('menu','menu::index');
  }
```

Et ainsi en retournant sur votre application vous voyez votre application avec le menu Fig.4. Et en redimensionnant la fenêtre (simulant un smartphone).

Conclusion

Vous avez pu voir dans ce tutoriel comment créer une application bootstrap. Le builder facilite la tâche dans la création d'une application bootstrap. Vous avez l'exposant “bootstrap” pour vous indiquer les modules adaptés.

Premiers pas avec le framework AngularJS

AngularJS est un framework Javascript porté par Google. C'est une infrastructure MVVM (Modèle-Vue-Vue/Modèle) qui se positionne côté front-end. Sa finalité étant de construire des Web Apps en opposition à NodeJS qui se positionne côté serveur. Ce framework structure le code et impose une architecture au projet. Il se base sur la librairie open source JQuery.

Alice Barralon
Chef de projet mobilité
alice.barralon@gmail.com

Pour commencer avec NodeJS, il est préconisé de connaître les bases fondamentales d'HTML, CSS et Javascript. Des notions de tests automatisés (BDD : Behavior Driven Development et TDD : Test Driven Development) sont un plus. Il n'est pas nécessaire d'être expert en JQuery et autres langages Web. Cet article s'appuie sur le tutoriel en anglais présent sur le site officiel d'AngularJS (https://docs.angularjs.org/tutorial). Tous les exemples cités peuvent y être copiés.

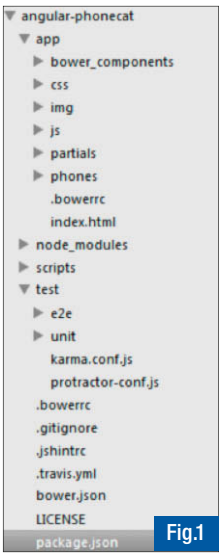
Les outils à installer pour travailler

- GIT (http://msysgit.github.io/)
- Editeur de texte de type : SublimeText (http://www.sublimetext.com/)
- Serveur Web local : NodeJS (http://nodejs.org/download/)

Une fois NodeJS installé, il est possible d'installer les dépendances avec la commande :
`[npm install]`. La commande va procéder à l'installation dans le répertoire courant `[node_modules]` des outils suivants :

- Bower (code package manager côté client),
- HttpServer (serveur local),
- Karma (test unitaire),
- Protractor (test end to end / E2E)

Elle va également sauvegarder les dépendances installées dans le fichier package.json. L'intérêt étant que les autres développeurs qui travaillent sur le projet pourront récupérer facilement toutes les dépendances associées.



Vue de l'architecture du projet

L'architecture du projet

Fig.1

- App : le répertoire de l'application qui comprend les templates HTML, vues partielles, CSS, images, js, etc.
- App/bower_components : les composants angular (camera, animation, webcam, router, etc.),
- App/js : le ou les controleurs JS,
- Test/ unit : les fichiers JS de tests unitaires,
- Test/e2e : les scénarios de tests fonctionnels,
- Node_module : les commandes angular installées par le projet (karma, protractor, bower, etc.)

Les principales commandes

Les commandes suivantes sont à exécuter depuis le répertoire de travail courant :

- `[npm start]` : démarre un serveur Web local qui écoute sur le port 8080 : http://localhost:8000/app/index.html
- `[npm test]` : démarre le test unitaire Karma. Le fichier de configuration est placé dans le dossier `[test/karma.conf.js]`. Les fichiers Javascript contenant les fractions de code à tester se trouvent dans le dossier `[test/unit]`.
  Une fois la commande lancée, les tests sont exécutés dans la fenêtre du navigateur et les résultats affichés dans la console. Karma "écoute" les fichiers de tests et dès qu'une ligne est modifiée, les tests sont rejoués automatiquement.
  Pour un développeur, c'est donc intéressant d'avoir une console ouverte avec les tests s'exécutant au fur et à mesure que le code évolue.
- `[npm run protractor]`: démarre les tests E2E (end to end) Protractor. Les fichiers Javascript contenant les tests de l'interface se trouvent dans le dossier `[test/e2e]`.
  Ces tests valident l'application dans son ensemble. Il faut lancer deux consoles. Dans une première console, démarrer le serveur `[npm start]` puis dans une seconde console lancer l'execution des tests `[npm run protractor]`.
  Une fois la commande lancée, les tests sont exécutés dans la fenêtre du navigateur, les résultats sont affichés dans la console. La fenêtre du navigateur est fermée à la fin des tests.

Pattern MVC

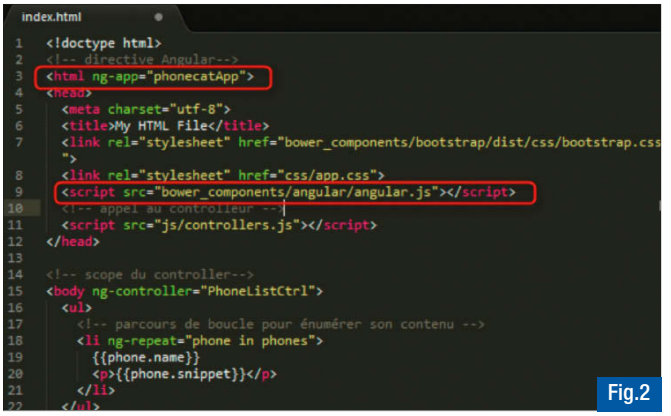
Angular encourage l'utilisation du pattern MVC (Model Vue Controller) pour rendre le code modulaire.

La Vue

La Vue est une projection du modèle à travers un template HTML. Par conséquent, lorsque le modèle change, la vue est impactée directement à travers le binding. L'appel d'Angular se fait directement dans le fichier HTML, par le biais de l'ajout de la directive `<html ng-app>` qui va informer le navigateur. Ajouter ensuite le tag d'appel dans le header :
`<script src="bower_components/angular/angular.js">`
L'appel se fait via des accolades de la manière suivante :
`<p>1 + 2 = {{ 1 + 2 }}</p>` Fig.2.

Le contrôleur

Il établit le data-binding entre le Modèle et la Vue. Il instancie via un constructeur le modèle de données. Le constructeur prend un `$scope`



Déclaration des directives AngularJS

en paramètre Fig.3. On retrouve dans cet exemple le module 'phonecatApp' déclaré dans la balise `<html>` de la vue et la déclaration du contrôleur présente dans la balise `<body>`.

Les tests

Le point fort d'Angular est la programmation par les tests. La séparation de la vue et du contrôleur facilite ainsi les tests unitaires. Dans cet exemple les tests sont écrits avec la librairie Javascript Jasmine (http://jasmine.github.io/). Le fonctionnement est simple : on écrit en langue naturelle les assertions et résultats attendus Fig.4 et 5.

> **Remarque :** *pour relancer les tests, il suffit simplement de changer le fichier javascript de tests ou la source, Karma repasse les tests automatiquement (si la console de lancement est toujours active) sans relancer la commande, ce qui est très pratique.*

Fonctionnalités Angular : la recherche textuelle

Si nous voulons pousser l'exemple plus loin en ajoutant un moteur de recherche, inutile de changer le contrôleur. Angular prévoit une directive qui effectue de la recherche textuelle sur la liste modèle : `Zone de recherche: <input ng-model="query">`
L'application du filtre donné dans la text box est immédiate et filtre les éléments du modèle en fonction du mot clé donné par la valeur [query] `phone in phones | filter:query` Fig.6
Pour tester ce type de filtre qui impacte directement le DOM, les tests unitaires ne sont pas idéaux (rappelons que le contrôleur n'a pas évolué). Il est préférable de mettre en place un scénario de test qui sera joué dans le navigateur.
Le scénario présenté ci-dessous va simuler 2 requêtes dans la zone de recherche et s'assurer que les résultats filtrés sont au nombre attendu Fig.7.

```
/* Controllers */
var phonecatApp = angular.module('phonecatApp', []);

phonecatApp.controller('PhoneListCtrl', function ($scope) {
  $scope.phones = [
    {'name': 'Nexus S',
     'snippet': 'Fast just got faster with Nexus S.'},
    {'name': 'Motorola XOOM™ with Wi-Fi',
     'snippet': 'The Next, Next Generation tablet.'},
    {'name': 'MOTOROLA XOOM™',
     'snippet': 'The Next, Next Generation tablet.'}
  ];
});
```

Fig.3

Controller et chargement du modèle de données

```
'use strict';

/* jasmine specs for controllers go here */

describe('PhoneListCtrl', function(){

  beforeEach(module('phonecatApp'));

  it('should create "phones" model with 3 phones', inject(function($controller) {
    var scope = {},
        ctrl = $controller('PhoneListCtrl', {$scope:scope});

    expect(scope.phones.length).toBe(3);
  }));

});
```

Fig.4

Test du contrôleur écrit dans l'exemple précédent

```
INFO [karma]: Karma v0.10.10 server started at http://localhost:9876/
INFO [launcher]: Starting browser Chrome
INFO [Chrome 35.0.1916 (Windows)]: Connected on socket TeUh5u31mYQfqx9ioiGy
Chrome 35.0.1916 (Windows): Executed 1 of 1 SUCCESS (0.197 secs / 0.018 secs)
```

Fig.5

Résultats des tests

> **Remarque :** *la syntaxe du test ci-dessus ressemble fortement au test du contrôleur écrit avec Jasmine et discuté précédemment.*

Pour lancer le test : `[npm run protractor]`

Fonctionnalité AngularJS : ajout d'une fonction de tri

En complément de la zone de recherche, nous allons ajouter la possibilité de trier la liste de résultats. Une nouvelle fois, les changements de code sont effectués dans le template HTML Fig.8. La partie recherche textuelle présentée précédemment reste inchangée. Une nouvelle directive `<select>` est ajoutée. L'impact au niveau du modèle se fait avec la directive `[orderBy]`. Le contrôleur est inchangé...

En conclusion, la prise en main du framework AngularJS est assez facile et rapide, les tutoriels sont nombreux et bien détaillés. D'autres technologies s'appuient sur AngularJS pour créer des applications mobiles Web ou natives, par exemple le framework Ionic (http://ionicframework.com/).

```
<div class="col-md-2">
  <!--Sidebar content-->

  Search: <input ng-model="query">

</div>
<div class="col-md-10">
  <!--Body content-->

  <ul class="phones">
    <li ng-repeat="phone in phones | filter:query">
      {{phone.name}}
      <p>{{phone.snippet}}</p>
    </li>
  </ul>
```

Fig.6

Filtre sur une text-box

```
'use strict';

/* http://docs.angularjs.org/guide/dev_guide.e2e-testing */
describe('PhoneCat App', function() {

  describe('Phone list view', function() {

    beforeEach(function() {
      browser.get('app/index.html');
    });


    it('should filter the phone list as user types into the search box', function() {

      var phoneList = element.all(by.repeater('phone in phones'));
      var query = element(by.model('query'));

      expect(phoneList.count()).toBe(3);

      query.sendKeys('nexus');
      expect(phoneList.count()).toBe(1);

      query.clear();
      query.sendKeys('motorola');
      expect(phoneList.count()).toBe(2);
    });
  });
});
```

Fig.7

Scénario pour tester le filtre sur le modèle de données

```
Search: <input ng-model="query">
Sort by:
<select ng-model="orderProp">
  <option value="name">Alphabetical</option>
  <option value="age">Newest</option>
</select>


<ul class="phones">
  <li ng-repeat="phone in phones | filter:query | orderBy:orderProp">
    <span>{{phone.name}}</span>
    <p>{{phone.snippet}}</p>
  </li>
</ul>
```

Fig.8

Ajout d'une fonction de tri

Les algorithmes de tri

1ère partie

En passant à la machine à café, vous avez sans doute déjà croisé des développeurs. Vous avez sans doute constaté qu'ils ont l'air passionnés par leur métier, ce qui rend leurs discussions animées. Et vous les avez sans doute entendu prononcer des mots comme « Bubble », « Quicksort », « logarithme » ou encore « complexité », qui semblent provenir d'une autre langue. Ce sont pourtant des notions primordiales en programmation. Dans cet article, nous allons tenter de démystifier ce charabia.

Thierry Leriche-Dessirier
Architecte JEE freelance / Team leader / Professeur à l'ESIEA
http://www.icauda.com

« Bubble », « Quicksort », « Insertion » ou encore « Fusion » (pour ne citer qu'eux) sont les noms donnés à des algorithmes de tri célèbres. Wikipedia indique(1) qu'un algorithme est « une suite finie et non ambigüe d'opérations ou d'instructions permettant de résoudre un problème ». Un algorithme de tri est donc la « recette de cuisine » permettant à un logiciel de ranger les éléments d'une liste.

On veut trier une liste lorsqu'on pense que ses éléments sont dans le désordre, ou plus précisément dans un ordre qui ne nous convient pas. L'objectif du tri, en tant qu'algorithme, est de mettre les éléments dans le bon ordre. Trier une liste suppose donc qu'on est capable d'établir une relation d'ordre entre ses éléments (ce qui n'est pas toujours simple et/ou possible). Par exemple, on peut dire si Nicolas est (objectivement) plus petit que David. Quand je parle de « plus petit », je fais instinctivement référence à la taille des personnes, mais on aurait tout aussi bien pu comparer leurs poids, leurs quantités de cheveux, leurs notes au baccalauréat, etc. Mais pour simplifier, disons juste qu'on se base sur le critère de la taille. Dans cet article, nous n'allons traiter que des entiers pour simplifier la discussion. La relation d'ordre sera donc établie.

Si ce sujet revient souvent chez les développeurs, c'est parce que le besoin de trier des données est omniprésent dans les programmes informatiques. Par habitude ou par inattention, on ne réalise pas qu'on manipule sans arrêt des données triées. Par exemple, disons qu'on s'intéresse à votre compte bancaire. Sur votre relevé, les opérations sont triées par date. Quand vous souscrivez à l'option Web, vous avez également la possibilité de trier selon les entêtes des colonnes (montant, date, numéro d'opération, bénéficiaire, type, etc.), mais vous ne vous êtes probablement jamais dit qu'il y avait un algorithme efficace caché derrière cette fonctionnalité.

> « *il faut choisir l'algorithme adapté au contexte...* »

En Java(2), on dispose de deux structures de données intéressantes à trier : les listes et les tableaux. Les algorithmes à base de listes sont plus faciles à programmer, mais sont généralement moins performants. Les tableaux demandent parfois de se retourner le cerveau pour coller à l'esprit de l'algorithme, mais le jeu en vaut la chandelle. C'est donc à ces derniers que nous allons nous intéresser. Dans cet article, nous allons discuter d'une poignée d'algorithmes. Pour que l'ensemble soit cohérent, je vous propose de créer l'interface « Tri » qui définit la méthode « trier() ». Cette méthode ne renvoie rien. Elle prend un tableau en paramètre et le trie « sur place ».

(1) *Algorithme sur Wikipedia :* http://fr.wikipedia.org/wiki/Algorithme
(2) *On aurait pu écrire le même article dans un autre langage avec finalement assez peu de retouches.*
(3) *3T :* http://icauda.com/articles.php#3t

Fig.1

© Puyo / Podcast Science

Photo de classe.

```
public interface Tri {
  void trier(final int[] tab);
  ...
```

Il faut préciser qu'il existe des bons et des mauvais algorithmes de tri. Quand on trie une petite liste et qu'on le fait rarement, même le plus mauvais des algorithmes fera l'affaire. Par contre, quand on manipule des listes conséquentes et/ou qu'on les trie souvent, il faut choisir avec attention l'algorithme qui sera le plus adapté au contexte.

On teste en trois temps

Avant d'entrer dans le vif du sujet, nous allons écrire des tests simples, à l'aide de la bibliothèque JUnit, qui nous permettront de vérifier de manière automatique que les programmes fonctionnent correctement. Cela s'inscrit dans une démarche qualité. Pour cela, nous allons employer la méthode 3T (Tests en Trois Temps(3)) qui s'inspire du TDD (Test Driven Development). Durant l'écriture de cet article, cette démarche a permis d'identifier des erreurs qui seraient très certainement passées inaperçues sans. Tous les algorithmes doivent passer par la même moulinette de tri. On peut donc en écrire une version « abstract », comme suit, dont devront hériter toutes les classes de test :

```
public abstract class AbstractTriTest {
  protected Tri tri;

  @Test
  public void testTriTabVide() {
    final int[] tab = {};
    doTestTri(tab);
  }

  @Test
  public void testTriTabUnSeulElement() {
    final int[] tab = { 1 };
    doTestTri(tab);
```

```
}

@Test
public void testTriTabDeuxElements() {
  final int[] tab = { 2, 1 };
  doTestTri(tab);
}

...

@Test
public void testTriTabMelange() {
  final int[] tab = { 8, 6, 3, 9, 2, 1, 4, 5, 7 };
  doTestTri(tab);
}

@Test
public void testTriTabQuasiTrie() {
  final int[] tab = { 1, 2, 3, 4, 5, 6, 9, 8, 7 };
  doTestTri(tab);
}

...

protected void doTestTri(final int[] tab) {
  tri.trier(tab);

  int temp = Integer.MIN_VALUE;
  for (final int elt : tab) {
    Assert.assertTrue(temp <= elt);
    temp = elt;
  }
}
```

Ici, l'idée est de multiplier les cas de tests. Vous trouverez de nombreux autres tests dans le code source proposé avec cet article sur le site Web de Programmez. Un des tests (non présenté ici) a consisté à générer, selon diverses distributions (aléatoire, croissante, décroissante, quasi croissante, gaussienne, sinusoïdale, etc.), des fichiers (des petits et des gros, voire très gros) contenant des listes à trier. En entreprise, on entend souvent dire (à tort ou à raison) qu'il faut utiliser les fonctionnalités du langage pour trier des tableaux. Cela va donc nous fournir un premier algorithme de référence. On commence bien entendu par la classe de test qui sera relativement simple :

```
public class JavaTriTest extends AbstractTriTest {

  @Before
  public void doBefore() {
    tri = new JavaTri();
  }
```

Fig.2

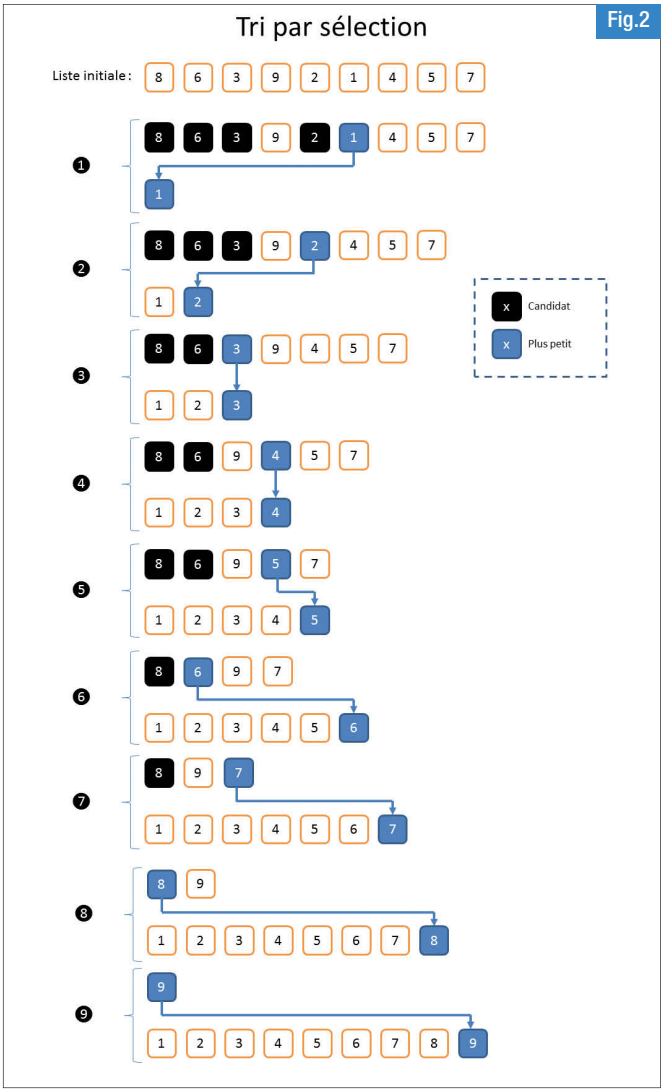
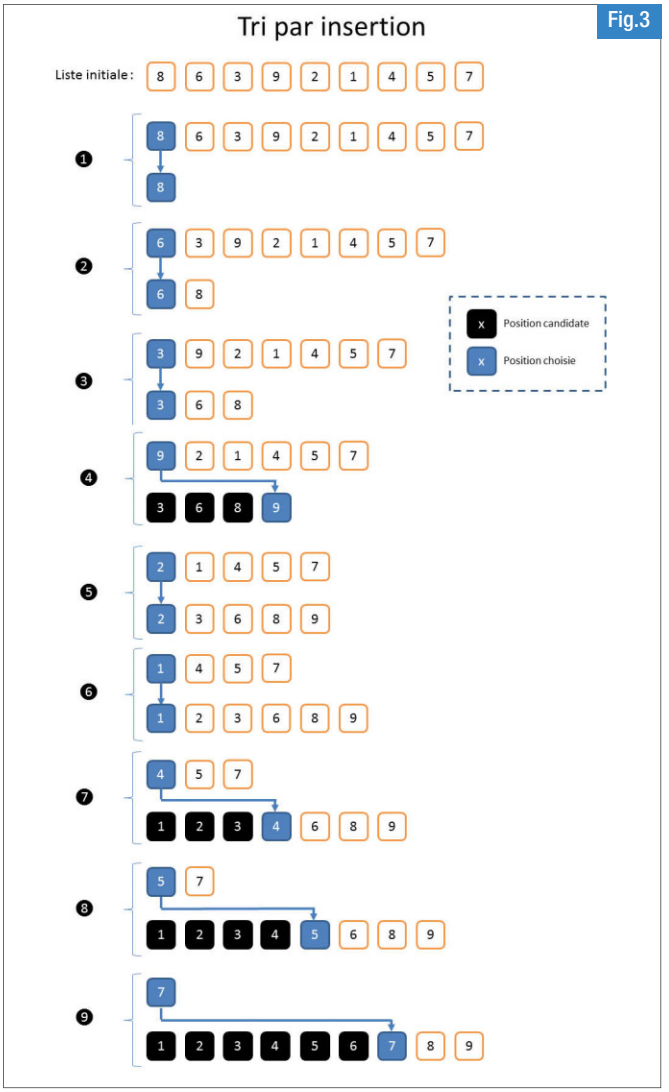


Fig.3



Dans la suite de cet article, les classes de test des autres algorithmes seront toujours calquées sur ce modèle. Elles ne seront donc pas présentées.

« il faut utiliser les fonctionnalités du langage... »

La classe qui met ça en pratique sera également très simple puisqu'il suffit de faire appel aux méthodes déjà existantes :

```
public class JavaTri implements Tri {

  @Override
  public void trier(final int[] tab) {
    Arrays.sort(tab);
  }
```

LES PREMIÈRES MANIÈRES QUI NOUS VIENNENT À L'ESPRIT

Tri par sélection

Sans le savoir, ou sans s'en rendre compte, on est confronté très jeune à des algorithmes de tri. Je dirais que cela se produit au plus tard en petite section de maternelle, vers l'âge de trois ans, à l'occasion du passage du photographe scolaire. Pour bien composer sa photo, le photographe doit placer les enfants sur le banc en fonction de leurs tailles Fig.1. Pour cela, le photographe passe en revue les enfants à la recherche du plus petit. Une fois trouvé, il le place sur le banc. Il réitère cette opération sur les enfants restant en plaçant à chaque fois l'enfant sélectionné à la prochaine place disponible sur le banc. L'algorithme se termine lorsqu'il ne reste plus d'enfant. Si vous avez bien suivi, vous aurez compris que le dernier enfant à être sélectionné, et donc à être placé sur le banc, est mécaniquement le plus grand Fig.2. Cet algorithme aurait pu s'appeler le « tri du photographe », mais on le trouve dans la littérature sous le nom de « tri par sélection ». Pour programmer ce tri sur un tableau, il suffit donc de le parcourir à la recherche du plus petit élément (ou du plus grand, ce qui revient au même) et de le positionner au prochain emplacement disponible :

```
public class SelectionTri implements Tri {

  @Override
  public void trier(final int[] tab) {
    for (int i = 0; i < tab.length; i++) {
      // Chercher la position (pos) du plus petit dans la zone restante
      int pos = i;
      for (int j = i; j < tab.length; j++) {
        if (tab[j] < tab[pos]) {
          pos = j;
        }
      }
      final int petit = tab[pos];

      // Delaler de i a pos
      for (int j = pos; i < j; j--) {
        tab[j] = tab[j - 1];
      }

      // mettre le plus petit trouve à la fin
      tab[i] = petit;
    }
  }
```

Dans la suite, on notera « n » le nombre d'éléments dans la liste à classer. Dans l'exemple, « n » correspond donc au nombre d'enfants dans la classe. À titre d'illustration, disons qu'il y en a dix-huit (cf. dessins). Pour sélectionner le plus petit, le photographe doit donc passer en revue les 18 élèves, ce qui nécessite de faire 17 comparaisons. Pour le deuxième, il reste 17 élèves à comparer, nécessitant 16 comparaisons. Pour le troisième il faudra faire 15 comparaisons, et ainsi de suite... Au final, le photographe va donc faire « 17+16+15+...+1 » comparaisons, ce qui est un peu long à calculer. Heureusement, on fait un peu de maths au lycée. Ça peut se factoriser à l'aide de la formule « n*(n-1) /2 », ce qui donne « 153 » comparaisons pour une classe de dix-huit bambins.

À la fin de la journée, le photographe doit donc avoir un sacré mal de tête. Et encore, il s'agit d'une classe relativement peu chargée. En école d'ingénieur, nous étions 150 dans ma promotion. Le photographe devrait donc faire plus de onze mille (11175) comparaisons pour préparer sa photo à l'aide du tri par sélection : autant dire qu'il y passerait la journée. Sans vendre la mèche, on devine déjà que le « tri par sélection » n'est pas réputé pour être très efficace.

Tri par insertion

Lorsqu'on est enfant, il y a un autre algorithme de tri qu'on apprend (ou découvre) assez vite. C'est celui que les joueurs de cartes utilisent pour organiser leurs mains. Le joueur pioche la carte qui est en haut du tas et la place (l'insère) à la bonne position dans la main. Cet algorithme se nomme le « tri par insertion » Fig.3.

En première approche, le « tri par insertion » est donc l'inverse du « tri par sélection ». Il est toutefois un peu plus efficace. Et, sans surprise, le code du programme est très proche de celui qu'on a déjà vu :

```
public class InsertionTri implements Tri {

  @Override
  public void trier(final int[] tab) {
    int pos;

    // On peut commencer à 1 directement...
    for (int i = 1; i < tab.length; i++) {
      final int temp = tab[i];

      // recherche position
      for (pos = 0; pos < i; pos++) {
        if (temp < tab[pos]) {
          break;
        }
      }

      // Décaler
      for (int j = i; pos < j; j--) {
        tab[j] = tab[j - 1];
      }

      // Insertion
      tab[pos] = temp;
    }
  }
```

Si on accepte de chercher la bonne position en partant de la queue, on peut en profiter pour réaliser le décalage du même coup.

```
    for (int i = 1; i < tab.length; i++) {
      final int temp = tab[i];
```

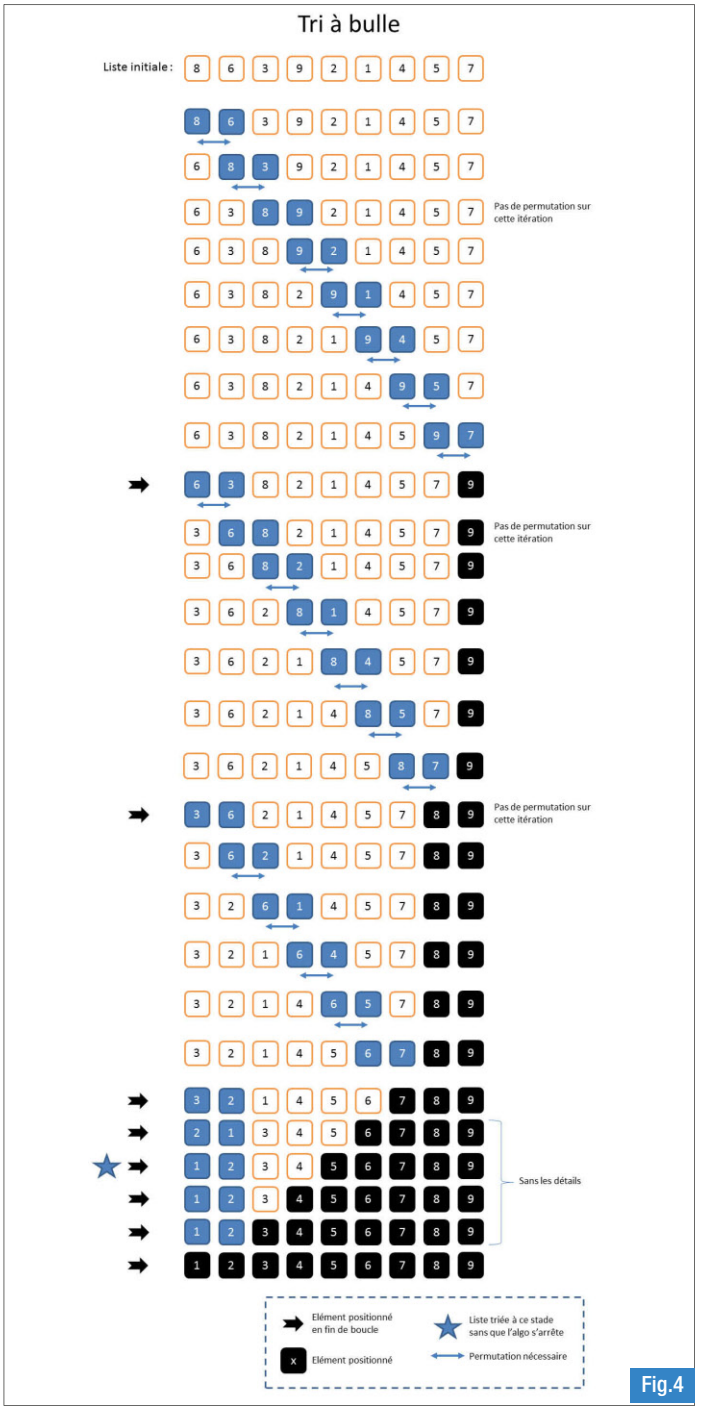


Fig.4

```
    // Recherche et décalage d'un seul coup
    for (pos = i; 0 < pos && temp < tab[pos - 1]; pos--) {
      tab[pos] = tab[pos - 1];
    }

    // Insertion
    tab[pos] = temp;
  }
```

Décider si on lance la recherche à partir de la tête ou de la queue n'est pas anodin. Selon que la liste initiale est plutôt croissante ou décroissante, l'une ou l'autre des recherches sera plus efficace. Et puisqu'on en est à se demander dans quel sens parcourir la liste, on pourrait aussi démarrer depuis la dernière position utilisée. Il suffit ensuite de monter ou de descendre le curseur selon le résultat des comparaisons et de s'arrêter lorsque ça s'inverse. Cela permet d'optimiser en moyenne le traitement des sous-séquences. Dans le pire des cas, cela sera équivalent à la recherche par une extrémité.

À titre d'illustration, prenons le jeu de la Belote(4) dans lequel chaque joueur reçoit huit cartes. Ici « n » vaudra donc « 8 ». Pour la première carte, il n'y a rien à faire puisque la main est vide. Pour la deuxième carte, il y a une comparaison à faire avec l'unique carte déjà en main pour savoir si elle doit aller à gauche ou à droite. Pour la troisième carte, il y aura « au maximum » deux comparaisons à effectuer. Il est important de comprendre qu'on arrête de faire des comparaisons dès que la bonne position est identifiée. On fait ainsi tant qu'il reste des cartes à piocher.
Au final, et dans le « pire des cas », on fera donc « 1+2+...+7 » comparaisons, soit « 28 » au total. C'est la même formule (à l'envers) de calcul que pour le « tri par sélection » et ce n'est donc pas mieux. En revanche, dans le « meilleur des cas » où le joueur pioche les cartes dans l'ordre idéal (ie. triées dans l'ordre croissant ou décroissant selon qu'on insère par la tête ou par la queue), il n'y aura que 7 comparaisons. Et bien entendu, plus le nombre « n » est grand et plus la différence est sensible.

Complexité

Comme vous l'avez surement déjà deviné, le nombre total d'opérations à effectuer pour « dérouler » un algorithme est un critère majeur pour dire si l'algorithme est efficace. Cela permet surtout de comparer l'efficacité de deux algorithmes et de choisir le plus adapté.
Il est relativement « simple » de dénombrer « précisément » le nombre d'opérations nécessaires pour « dérouler » un algorithme de tri lorsque le nombre d'éléments « n » dans la liste est petit. Par exemple, on a vu qu'il faudra réaliser « 153 » comparaisons pour trier « 18 » enfants à l'aide du « tri par sélection ». Le même algorithme nécessite de réaliser « 11.175 » comparaisons pour trier une classe de « 150 » élèves, et « 499.500 » pour trier les mille employés d'une PME. Et si vous vouliez trier les « 81.338 » spectateurs d'un match de foot au Stade de France, il faudrait « 3.307.894.453 » comparaisons.
Ces exemples mettent deux choses en évidence. Premièrement, la quantité d'opérations nécessaire au déroulement du « tri par sélection » croît rapidement lorsque le nombre « n » d'éléments dans la liste augmente. Deuxièmement, il n'est pas nécessaire d'avoir une grande précision sur des chiffres aussi importants. Quand on réalise « 3.307.894.453 » comparaisons, on n'est plus à une comparaison près, ni à dix, ou cent, ou même cent millions. D'ailleurs on aurait pu se contenter de dire qu'il fallait « environ trois milliards » d'opérations. Ce qui compte, avec des valeurs si conséquentes, c'est d'avoir un « ordre de grandeur », même vague. C'est ce qu'on va appeler la « complexité » d'un algorithme.
En fait, il n'est pas possible de discuter des tris sans parler de « complexité ». Ce mot à lui seul est synonyme de cauchemars pour bien des étudiants en école d'info. Et pour ma part, je dois avouer que j'ai mis longtemps à en comprendre les tenants et les aboutissants. Je me propose donc de vous l'expliquer simplement, en prenant quelques raccourcis. Ne vous inquiétez pas si tout n'est pas clair, ça va se clarifier au fur et à mesure.

« discuter des tris impose d'aborder la complexité... »

La complexité d'un tri de « n » éléments se note avec un « omicron » : dites « grand O ». Par exemple, la complexité du « tri par sélection » sera en « O(n*(n-1) /2) », où « n*(n-1) /2 » est la formule qu'on avait utilisée un peu plus tôt. Comme on s'intéresse à des valeurs importantes et qu'on ne veut qu'un « ordre de grandeur », on considèrera que cette « complexité » peut se « simplifier » en « O(n*(n-1)) », qui peut elle-même se simplifier en « $O(n^2)$ ». On dira que l'algorithme du « tri par sélection » est de « complexité quadratique » en « $O(n^2)$ ».

(4) *La belote se joue à 4 joueurs avec un paquet de 32 cartes, soit 8 pour chaque joueur.*

Pour faire simple, quand vous multipliez par « 10 » le nombre « n » d'éléments, vous multipliez par « 100 » le nombre d'opérations à réaliser lorsque vous utilisez un algorithme de tri en « $O(n^2)$ ». Dit comme ça, ce n'est pas très impressionnant alors essayons d'associer cela avec des durées pour voir à quel point c'est mauvais. Disons qu'une opération prenne une nanoseconde (ns) en moyenne (ce qui est très optimiste).

Nombre d'éléments « n »	Nombre d'opérations pour un tri en « $O(n^2)$ »	Durée pour un tri en « $O(n^2)$ »
10	100	100 ns
100	10 000	10 us
1 000	1 000 000	1 ms
10 000	100 000 000	100 ms
100 000	10 000 000 000	10 s
1 000 000	1 000 000 000 000	16 min 40 s
10 000 000	100 000 000 000 000	27 heures
100 000 000	10 000 000 000 000 000	115 jours
1 000 000 000	1 000 000 000 000 000 000	31 ans

Alors que le tri de « 10 » éléments prendra seulement « 100 » nanosecondes à l'aide d'un algorithme en « $O(n^2)$ », il faudra « 10 » microsecondes pour une centaine d'éléments et carrément une milliseconde pour mille. Et ces temps augmentent encore quand le nombre d'éléments augmente. Ca augmente même très vite. Pour un milliard d'éléments, ce qui reste raisonnable pour un programme informatique, notamment dans le domaine de la finance, on aura besoin d'un tiers de siècle. Autant dire que les données seront périmées depuis longtemps quand l'algorithme aura fini son travail. L'ordinateur sera peut-être même déjà passé à la poubelle. Et je ne vous parle même pas de la facture d'électricité qu'on aurait tendance à oublier (coût en énergie, en ressource naturelle, en entretien et en Euro).

Pour bien réaliser ce que ça représente, prenons un exemple plus concret. Pour trier la population mondiale(5), soit un peu plus de sept milliards de personnes (au début de l'été 2014), il faudrait plus d'un millier et demi d'années (1655 ans environ). Pour que le tri soit fini au moment où vous lisez cet article, il aurait donc fallu le commencer en l'an 350 de notre ère, soit plus d'un siècle avant le début du moyen âge (476 à 1453).

On parle de complexité en se plaçant dans la situation la plus défavorable pour l'algorithme. Ça permet en quelque sorte de majorer. On peut aussi se placer dans le cas le plus favorable ou dans le cas moyen. Par exemple et sans plus d'explications, le « tri par insertion » sera de « complexité linéaire » en « O(n) » dans le cas le plus favorable et en « $O(n^2)$ » dans les cas moyens et défavorables. Bon, je ne vais pas vous embêter plus que ça avec cette histoire de complexité. J'ai passé sous silence certains aspects (gestion de la mémoire, déplacements, permutations, création d'objets, flux, etc.) qui peuvent avoir un impact en termes de performance, en fonction de la plateforme et du langage utilisés, mais je pense sincèrement que l'essentiel y est... pour l'instant... En plus de la question des performances, on a pris l'habitude de classer (cf. tableau récapitulatif en fin d'article) les algorithmes de tri selon des caractéristiques importantes, comme la stabilité ou le fait de pouvoir trier sur place, c'est-à-dire qu'on réarrange les éléments directement dans la structure initiale. Un algorithme est dit « stable » lorsqu'il ne modifie pas l'ordre des éléments de même valeur. Enfin, une des principales contraintes concerne l'espace mémoire. Plus spécifiquement, on se demande si on est capable de trier l'ensemble des données en mémoire centrale. On dira que le tri est « interne » si c'est possible, et « externe » sinon.

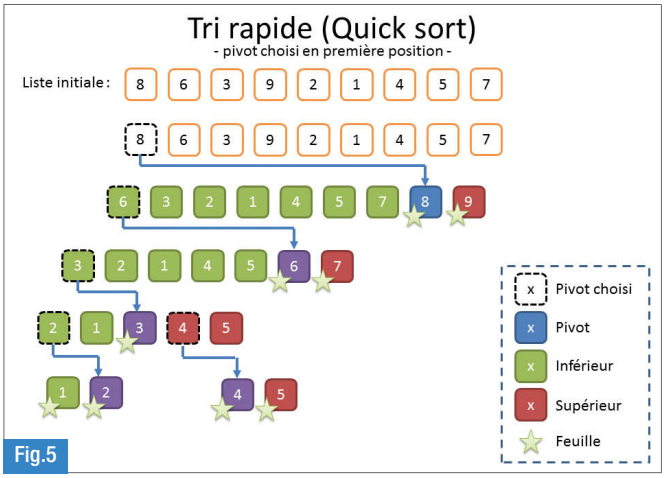


Fig.5

Comment faire mieux ?

On devine qu'il va falloir trouver comment faire mieux, notamment en utilisant des algorithmes beaucoup plus rapides. Rassurez-vous, je ne vais pas vous présenter tous les algorithmes de tri qui existent. Je vais me limiter aux plus connus en insistant sur leurs points forts et leurs faiblesses.

Tri à bulle (Bubble Sort)

Le tri à bulle est certainement celui qu'on apprend à programmer en premier en école d'informatique, avant même le « tri par sélection » et le « tri par insertion ». Pour effectuer un « tri à bulle », il faut parcourir la liste en permutant les éléments contigus qui ne sont pas dans le bon ordre. Par exemple, si je trouve « 9 » dans la case « 4 » et seulement « 2 » dans la case « 5 », alors j'échange leurs positions. Quand c'est fini, l'élément le plus grand se retrouve donc en queue de liste. Cet élément est arrivé à sa bonne position, mais le reste de la liste est encore potentiellement en désordre. On recommence autant de fois qu'il y a d'éléments dans la liste, ce qui fait donc à chaque fois remonter le plus grand élément restant Fig.4. Le nom du « tri à bulle » vient du fait que les plus gros éléments remontent comme des bulles dans une flute de champagne. D'ailleurs, on n'utilise jamais le « tri à bulle », car les bulles remontent très lentement.

```
public class BulleTri implements Tri {

  @Override
  public void trier(final int[] tab) {
    for (int i = 0; i < tab.length - 1; i++) {
      for (int j = 1; j < tab.length; j++) {
        if (tab[j] < tab[j - 1]) {
          // Permutation
          permuter(j, j - 1, tab);
        }
      }
    }
  }

  public static void permuter(final int indexA, final int indexB,
final int[] tab) {
    final int temp = tab[indexA];
    tab[indexA] = tab[indexB];
    tab[indexB] = temp;
  }
```

Pour une liste dans laquelle le nombre d'éléments « n » est de « 9 », on fera « 8 » (ie. « n-1 ») comparaisons pour mettre le plus grand élément à sa bonne position. Et on va répéter cela « 8 » fois également, ce qui nécessitera donc « 64 » comparaisons. La complexité associée au « tri à bulle »

(5) *Population mondiale estimée au 1er juillet 2014 : 7 226 376 025 personnes.*

sera donc en « O((n-1)n) » qu'on simplifiera en « O(n^2) ». On peut améliorer l'algorithme de base. En effet, à chaque passage, un élément supplémentaire se retrouve à sa place définitive. Il n'est donc plus nécessaire de l'inclure dans les comparaisons suivantes.

```
for (int i = 0; i < tab.length - 1; i++) {
  for (int j = 1; j < tab.length - i; j++) {
    if (tab[j] < tab[j - 1]) {
      permuter(j, j - 1, tab);
```

On n'aura alors besoin que de « (n-1)*n/2 » comparaisons, soit « 36 » comparaisons pour une liste de « 9 » éléments. La complexité restera toutefois en « O(n^2) », le multiplicateur «1/2» ne changeant rien à l'histoire pour les grandes valeurs, comme on l'a déjà vu.

« accusé d'être lent, le tri à bulle cache bien son jeu... »

En outre, on peut stopper le déroulement de l'algorithme dès lors qu'on détecte qu'aucune permutation n'a été réalisée durant une boucle. Si la liste s'y prête, cela peut devenir une sérieuse optimisation.

```
public class RapideDemiBulleTri implements Tri {

  @Override
  public void trier(int[] tab) {
    for (int i = 0; i < tab.length; i++) {
      boolean permutation = false;
      for (int j = 1; j < tab.length - i; j++) {
        if (tab[j] < tab[j - 1]) {
          // Permutation
          permuter(j, j - 1, tab);
          permutation = true;
        }
      }
      if (!permutation) {
        break;
      }
    }
  }
```

Le « tri à bulle » est souvent accusé d'être lent, mais comme vous pouvez le voir, il cache bien son jeu. À titre personnel, je dois avouer que j'ai beaucoup d'affection pour cet algorithme, même en sachant que ce n'est pas le meilleur.

Tri rapide (Quick Sort)

S'il y a un algorithme de tri dont vous avez forcément entendu parler à la machine à café, c'est bien le « Quick sort ». C'est l'un des algorithmes les plus utilisés et certainement celui qui présente le plus de variantes. Le « Quick Sort », aussi appelé « tri rapide », fait partie de la famille d'algorithmes dont le fonctionnement repose sur le principe « diviser pour régner ». Pour réaliser un « tri rapide », on doit choisir un élément dans la liste, qu'on appelle « pivot ». On divise ensuite la liste en deux sous-listes. La première, à gauche, contient les éléments inférieurs au pivot. La seconde, à droite, contient les éléments supérieurs au pivot.

« vous avez forcément entendu parler du Quick Sort... »

On reproduit alors récursivement ce choix du pivot et la division sur les listes de gauche et de droite précédemment construites jusqu'à n'avoir que des sous-listes de zéro ou un élément. Pour finir, il suffit de rassembler les éléments de toutes les sous-listes dans l'ordre gauche-droite Fig.5. Le principe du « tri rapide » est donc très simple : tout se passe pendant la construction de l'arbre.

En revanche, il est réellement (très) difficile de programmer cet algorithme dans la version utilisant des tableaux. Plus précisément, c'est la séparation en fonction du pivot qui pose problème à de nombreux développeurs. On trouve d'ailleurs de nombreux codes faux (qui ne passent pas mes tests) sur Internet. Je vous propose donc de commencer en douceur avec une version employant des listes.

```
public class RapideViaListeTri implements Tri {

  private List<Integer> trier(final List<Integer> liste) {
    if (liste == null || liste.isEmpty()) {
      return new ArrayList<>();
    }

    // Choix du pivot à gauche
    final int pivot = liste.get(0);
    // pour ne pas traiter le pivot dans la boucle
    liste.remove(0);

    final List<Integer> listeGauche = new ArrayList<>();
    final List<Integer> listeDroite = new ArrayList<>();

    // Séparation en deux sous listes
    for (final int elt : liste) {
      if (elt < pivot) {
        listeGauche.add(elt);
      } else {
        listeDroite.add(elt);
      }
    }

    final List<Integer> result = new ArrayList<>();

    result.addAll(trier(listeGauche));
    result.add(pivot);
    result.addAll(trier(listeDroite));

    return result;
  }
```

À la lecture de ce code, vous avez sans doute repéré plusieurs problèmes techniques. La méthode fonctionne correctement (elle passe tous mes tests), mais elle est lente. Il y a fondamentalement trois « erreurs ». D'abord de nombreuses listes sont créées et manipulées durant le traitement, ce qui coûte cher.

Ensuite, l'auto-unboxing réalisé lors de l'itération sur la (sous) liste à trier ajoute encore un peu à l'addition. Et l'auto-boxing lors de l'ajout aux sous-listes finit de « plomber » la note.

Cela peut se résoudre en travaillant directement sur des objets, et donc sur des références, en les comparant à l'aide de « compareTo() » en lieu et place de l'opérateur de comparaison.

Enfin, l'algorithme ne prend pas en compte la présence (éventuelle) de plusieurs valeurs égales au pivot. Quitte à avoir deux sous-listes, on peut bien en gérer une troisième pour ce cas spécifique. Le nombre total d'éléments restera inchangé et le coût du test supplémentaire sera largement compensé par le gain en nombre de récursions.

```
public class RapideViaListe2Tri implements Tri {

  private List<Integer> trier(final List<Integer> liste) {
    if (liste == null || liste.isEmpty()) {
      return new ArrayList<>();
    }

    // Pivot à gauche
    final Integer pivot = liste.get(0);

    final List<Integer> listeGauche = new ArrayList<>();
    final List<Integer> listeCentre = new ArrayList<>();
    final List<Integer> listeDroite = new ArrayList<>();

    // Séparation en deux sous listes
    for (final Integer elt : liste) {
      final int comp = elt.compareTo(pivot);
      if (comp < 0) {
        listeGauche.add(elt);
      } else if (comp == 0) {
        listeCentre.add(elt);
      } else {
        listeDroite.add(elt);
      }
    }

    liste.clear();

    liste.addAll(trier(listeGauche));
    liste.addAll(listeCentre);
    liste.addAll(trier(listeDroite));

    return liste;
  }
```

Voici maintenant une première version utilisant des tableaux. L'intérêt de cette structure est de pouvoir effectuer le tri « sur place », en manipulant des primitifs.

```
public class RapideTri implements Tri {

  @Override
  public void trier(final int[] tab) {
    trier(tab, 0, tab.length - 1);
  }

  private void trier(final int[] tab, final int gauche, final
int droite) {
    if (droite <= gauche) {
      return;
    }

    // Pivot à gauche
    int positionPivot = gauche;
    permuter(positionPivot, droite, tab);

    for (int i = gauche; i < droite; i++) {
      if (tab[i] <= tab[droite]) {
        permuter(i, positionPivot++, tab);
      }
    }
    permuter(droite, positionPivot, tab);

    // tri récursif des sous-listes
    // Bien entendu on ne trie pas le pivot
    trier(tab, gauche, positionPivot - 1);
    trier(tab, positionPivot + 1, droite);
  }
```

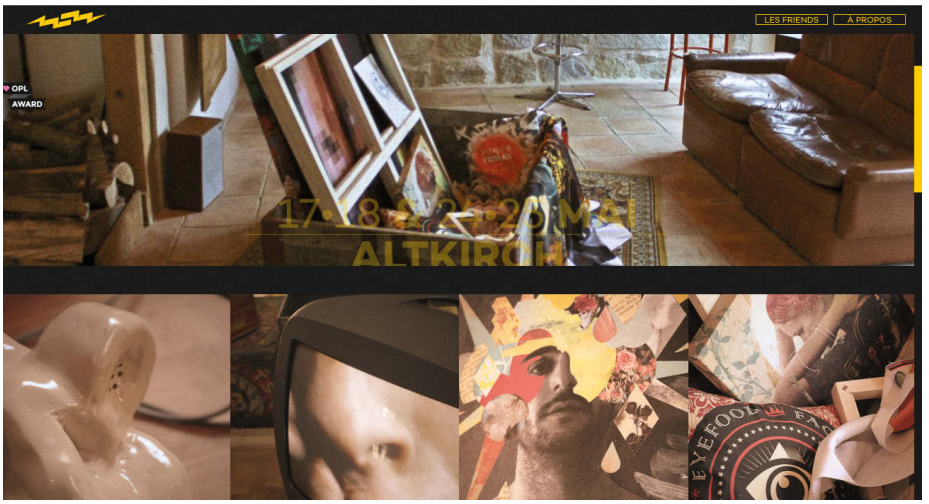
Comme vous le constatez, cette première version utilisant des tableaux est bien plus complexe. On arrive toutefois encore à distinguer la forme de base de l'algorithme. Un des gros défauts, difficiles à corriger, de cette version est de devoir gérer le décalage d'un grand nombre de cases. Pour gagner encore un peu en vitesse, on va devoir passer par des fonctions de « System », qui font des manipulations de la mémoire de façon native.

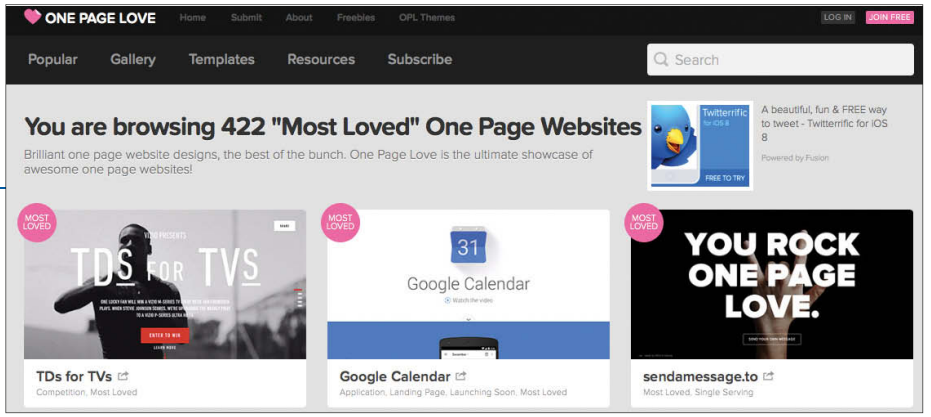
```
public class RapideTri2 implements Tri {

  @Override
  public void trier(final int[] tab) {
    if (tab.length < 2) {
      return;
    }

    int[] tab2 = new int[tab.length];
    trier(tab, tab2, 0, tab.length);
    System.arraycopy(tab2, 0, tab, 0, tab.length);
  }

  private void trier(final int[] tab, final int[] tab2, final
int gauche, final int droite) {
    // Choix du pivot à gauche
    final int pivot = tab[gauche];
    int posGauche = gauche;
    int posDroite = droite;

    // Séparation en deux sous listes
    for (int index = gauche + 1; index < droite; index++) {
      int elt = tab[index];
      if (elt < pivot) {
        tab2[posGauche++] = elt;
      } else if (elt > pivot) {
        tab2[--posDroite] = elt;
      }
    }

    Arrays.fill(tab2, posGauche, posDroite, pivot);

    if (posGauche > gauche) {
      trier(tab2, tab, gauche, posGauche);
      System.arraycopy(tab, gauche, tab2, gauche, posGauche - gauche);
    }
    if (posDroite < droite) {
      trier(tab2, tab, posDroite, droite);
      System.arraycopy(tab, posDroite, tab2, posDroite, droite
- posDroite);
    }
  }
```

Suite au prochain numéro.

Le design « One Page »

Le développement One Page est un sujet simple à aborder et rapide, je vais commencer par expliquer à quoi ça sert, comment il fonctionne et enfin pourquoi le faire.
Le but du développement One Page est un concept simple à comprendre : il s'agit d'afficher toutes les informations de votre site (ou de vos sites si vous voulez en faire plusieurs) sur une seule et même page avec une navigation qui peut être verticale ou horizontale.





De nombreux templates (gratuits ou payants) sont disponibles.

Nicolas Le Bot

Je parlerai aussi des avantages et des contraintes autour de ce mode de développement, car il y a un peu des deux ; vous verrez par vous-même lors de votre développement que cela peut être une bonne idée comme une mauvaise, tout dépend de ce que vous cherchez à faire sur votre site.

Pourquoi faire un site One Page ?

Commençons déjà par le fait que le One Page se passe surtout côté design, car c'est à ce niveau-là que se joue le rôle le plus important du développement. Le but du développement One Page est surtout de faire un site où toutes les informations seront sur la même page avec la possibilité de naviguer rapidement et facilement sur cette même page.
Car si le but est de faire un site rapide, ce sera sur une durée limitée; vous n'allez probablement pas garder un site One Page pour votre site tout le temps, sauf si bien sûr c'est par exemple pour un site vitrine où il faut juste des informations importantes pour les utilisateurs.
Maintenant il est important de comprendre qu'il faut faire une différence entre une bonne et une mauvaise explication de ce concept; il ne s'agit pas de faire un site One Page horrible qui ne donne pas du tout envie de le visiter ou même de le parcourir.
Prenons par exemple ce site : http://friends.overblitz.com/
Comme vous pouvez le voir, il y a une scrollbar différente des sites habituels ainsi qu'un développement intégral sur une seule page.
Pour la mise en prod de votre "produit", c'est très simple à mettre en place, puisque comme tout site, vous pouvez le tester directement en local ou bien sur votre serveur ! Les sites One Page fonctionnent sur tous les navigateurs (Sauf IE 6).

Comment faire un site One Page ?

Il faut d'abord partir du principe qu'un site One Page est un site où les contenus seront peu mis à jour. Dans cet article, le but est de faire du One Page rapidement, peu importe le contenu. Ensuite si on le souhaite (cela est toujours utile), il est assez facile via du PHP de faire un affichage multilingue (via le PHP vous pourrez toujours faire la mise à jour des contenus, mais encore une fois ce n'est pas le but direct d'un site One Page).
Nous partirons donc sur l'utilisation du HTML/CSS, avec du PHP pour les langues et le contenu.

Les avantages d'un site One Page

En premier, nous allons partir du principe qu'il est simple à réaliser et ne devrait pas vous occuper pendant très longtemps dans votre vie de développeur. En second, l'utilisation du HTML/CSS et l'évolution des technologies va vous permettre assez facilement de faire votre code de façon à pouvoir avoir votre site optimisé aussi pour les navigateurs mobiles (ce qui est de nos jours utile).
Enfin, en dernier lieu, la facilité d'utilisation du PHP pour créer une version multilingue de votre site va pouvoir permettre l'ouverture sur l'international ainsi que la mise à jour des contenus, cela si jamais vous en avez vraiment besoin. J'allais oublier que le fait de faire votre site en One Page, permettra à une personne ayant de bonnes bases en CSS, de pouvoir faire des modifications assez simplement.
Ce genre de site étant assez simplement modifiable et c'est bien plus simple pour un designer d'ailleurs de travailler sur un site One Page.

Les contraintes d'un site One Page

On va passer à l'une des parties les moins amusantes, les contraintes, car en effet il y a des contraintes. Je vais énumérer quelques problèmes aux-

quels je n'ai malheureusement pas forcément toutes les solutions.

- En premier, le souci de ce genre de site est le temps de chargement de la page; certes de nos jours les connexions sont plutôt excellentes mais cela peut s'avérer parfois long quand on doit tout charger en même temps. Du coup nous pouvons éventuellement précharger les images si besoin (par exemple si votre site contient trop d'images, ou qu'elles sont trop lourdes)
- En second, nous allons avoir le souci du référencement par les moteurs de recherche; si vous voulez toujours avoir un minimum de référencement, le mieux sera d'aller jusqu'à donner des noms "physiques" aux images (cela permettra si jamais vous nommez une image src="j'aime la mousse au chocolat", d'être mieux référencé sur ce thème lorsque quelqu'un le cherchera sur son moteur de recherche.
- Et enfin, (sûrement l'un des plus importants), c'est de faire quelque chose de "joli" pour que la navigation de l'utilisateur devienne un moment d'exception, pour qu'il prenne plaisir à naviguer sur votre One Page. Du côté des exemples, vous pouvez par exemple refaire une scrollbar, ou encore faire votre propre scroll au moment du clic de boutons (avec une descente particulière). Bref, c'est au développeur de faire quelque chose de sympathique pour permettre à l'utilisateur d'apprécier le site.
- Une dernière contrainte peut être liée à l'utilisation des navigateurs. En effet, Internet Explorer pose souvent des souci aux devs car les versions 6 (essentiellement) ont été mal optimisées; ne pas oublier du coup de vérifier sur tous les navigateurs votre CSS !

Il va falloir faire attention au développement lors de la création, il est un petit peu plus compliqué de faire du One Page que du développement normal (ce qui peut paraître logique mais pas toujours), il faut penser à tout bien prendre en compte.

On va probablement bientôt attaquer la partie la plus agréable, celle du développement avec quelques exemples de codes qui peuvent vous servir pour vos sites One Page.

Il ne faut pas oublier, que le développement One Page concerne le CSS surtout, donc il y faut un minimum de bases avant de se lancer dans ce genre de site, sinon ça ne sera pas forcément une partie de plaisir (même si dans le pire des cas, apprendre le CSS peut être parfois amusant).

LE DÉVELOPPEMENT ONE PAGE

Nous allons commencer par faire des petits trucs sympas en CSS, comme la scrollbar !

Dans le HTML nous allons avoir ça :

```
<div class="scrollbar" id="ex3">
    <div class="content">Example 3</div>
</div>
```

Et dans le CSS :

```
.scrollbar{
width:150px;
height:300px;
background-color:lightgray;
margin-top:40px;
margin-left:40px;
overflow-y:scroll;
float:left;
}
.content{
height:450px;
}
```

(Ceci est la scrollbar de base, c'est pour le début !)

Dans le CSS, on va maintenant changer la scrollbar et mettre ceci :

```
#ex3::-webkit-scrollbar{
width:16px;
background-color:#cccccc;
}
```

Cela va nous faire une scrollbar de 16px avec un background-color: #cccccc (gris donc)

Dans la suite, nous allons mettre de la couleur à notre barre, ce qui va la rendre un peu plus jolie ! (ce n'est pas encore parfait, mais ça avance)

```
#ex3::-webkit-scrollbar-thumb{
background-color:#B03C3F;
border-radius:10px;
}
#ex3::-webkit-scrollbar-thumb:hover{
background-color:#BF4649;
border:1px solid #333333;
}
#ex3::-webkit-scrollbar-thumb:active{
background-color:#A6393D;
border:1px solid #333333;
}
```

Après avoir fait ceci, nous allons définir le "border" de la barre pour un peu plus d'esthétique.

```
#ex3::-webkit-scrollbar-track{
border:1px gray solid;
border-radius:10px;
-webkit-box-shadow:0 0 6px gray inset;
 }
```

Et voilà, avec ceci, vous devriez avoir une scrollbar fort différente de celle d'avant, rouge, et qui change quand vous cliquez dessus (avec la délimitation de la scrollbar).

Conclusion

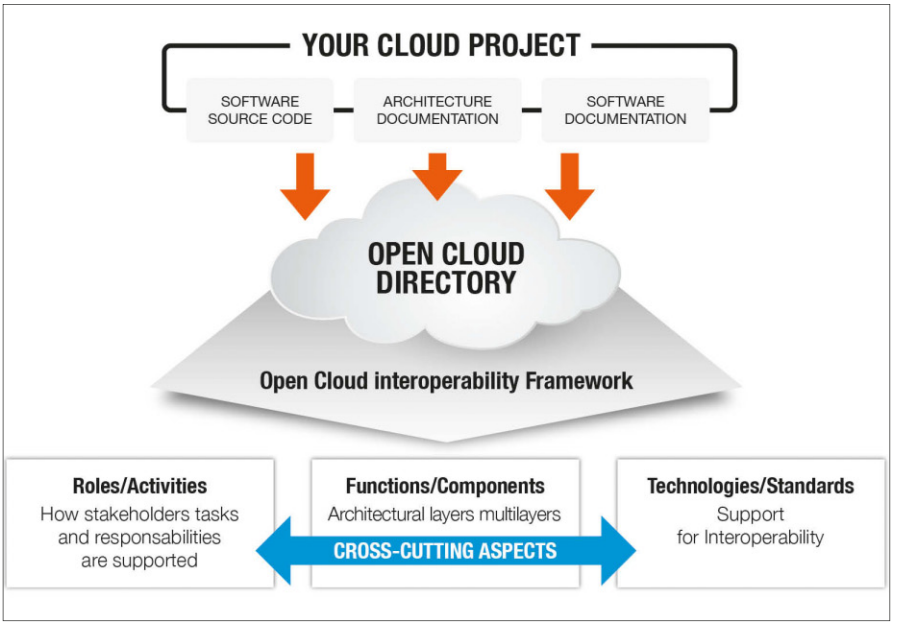
Cet article se termine. Nous aurons vu ce qu'est le développement « One Page Design », les avantages et les contraintes, ainsi qu'un exemple en CSS pour changer la Scrollbar, et, bien sûr, pourquoi nous pouvons utiliser le développement One Page. Il va sans dire que c'est une solution très agréable à utiliser, rapide et peu coûteuse.

Très prochainement la suite.

Open World Forum révèle l'innovation collaborative

L'innovation ouverte, une utopie ? Pas pour les projets collaboratifs, soutenus par la Commission Européenne, qui produisent des résultats open source tangibles.

Olivier Bouzereau



Hélas, ces programmes restent encore sous-utilisés car méconnus. Pourtant, ils sont réalisés par des groupes talentueux composés d'industriels, de chercheurs et de PME innovantes. Conçus sur une période de 24 à 36 mois, ils représentent l'état de l'art scientifique en termes de connaissances et de programmation pour le Cloud Computing, l'Internet des objets ou encore l'analyse de données massives.

Lors de l'Open World Forum de Paris, fin octobre, sept projets invités par la communauté OW2, spécialisée dans les logiciels d'infrastructure open source, répondaient à des problématiques actuelles ou aux usages concrets d'entreprises et de collectivités.

1 La plateforme RISCOSS (riscoss.eu) fournit une approche méthodologique visant à réduire les risques d'inclure des composants open source dans le développement d'un nouveau produit ou service. En cours d'intégration sur X-Wiki, elle réunit des algorithmes i* et des connaissances issues d'une corrélation d'indicateurs sur les risques juridiques, techniques et organisationnels liés aux communautés produisant des logiciels open source. En améliorant la gestion des risques, les développeurs pourront implémenter des solutions open source plus durables.

2 Pour mieux explorer les composants open source destinés au Cloud Computing, le projet OCEAN (ocean-project.eu) fournit un catalogue raisonné et plusieurs outils de navigation. On peut ainsi détecter des codes récents issus de projets FP7 soutenus par la commission européenne ou d'initiatives locales. Développé sous le portail CKAN – très apprécié dans le domaine de l'open data -, l'Open Cloud Directory facilite l'examen des documentations, architectures et programmes open source, rapports de tests inclus. Ce référentiel décrit plus de 77 projets et 157 actifs que l'on peut dénicher par arborescence, couches fonctionnelles ou activités Cloud soutenues.

3 CompatibleOne, le premier broker Cloud open source a généré plus d'une trentaine de solutions professionnelles déjà. La jeune pousse Cloud Orbit permet ainsi de bâtir un Cloud hybride répondant aux critères économiques et techniques de chaque client. Les standards supportés par CompatibleOne – OCCI et WS Agreement – facilitent l'interopérabilité entre les nuages d'Amazon, Google, Microsoft, OpenStack et VMware. L'allocation automatique de ressources sur des Clouds publics ou privés s'effectue conformément aux seuils déterminés par le client, un suivi de la qualité de services et des engagements contractuels étant mené en tâche de fond.

4 Le projet BigFoot (bigfootproject.eu) vient sur l'écosystème Hadoop et OpenStack. Il aide à bâtir des fermes optimisées pour l'analyse de grands volumes de données. Sur un Cloud privé, BigFoot propose un planificateur de tâches pour Hadoop et un déploiement automatique de traitements pré-configurés. La plateforme PaaS (Platform-as-a-Service) s'apprête donc à gérer les interactions avec les données issues de Smart Grids ou de systèmes de sécurité par exemple.

5 Avec OSSMeter, la mesure et l'analyse des bugs présents dans les logiciels open source gagnent de nouveaux automatismes. « *Cette plateforme Cloud – dont une première démonstration est prévue en novembre – cherche à faciliter la découverte et la comparaison de projets open source. Elle peut également assurer la surveillance et l'analyse de la qualité des projets de développement en interne* », explique Alessandra Bagnato, ingénieur chez Softeam R&D qui participe au projet.

6 ModaClouds est un projet de gestion DevOps, autrement dit un outil rassemblant les utilisateurs métiers des développeurs et exploitants informatiques. Sa mission consiste à réduire les performances imprévisibles dans le Cloud ainsi que le verrouillage imposé par les fournisseurs d'infrastructures et d'applications distribuées. ModaClouds cherche à fournir un ensemble de services élastiques, souples pour un déploiement multi-cloud rapide. Des outils de surveillance, de migration d'applications et d'automatisation de la mise en production sont intégrés autour de métriques unifiées et de règles SLA définies par l'utilisateur.

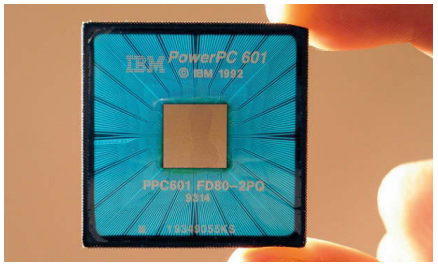
7 Enfin, ClouT (clout-project.eu) veut apporter une architecture de référence aux objets intelligents et aux villes connectées. Réunir Cloud et IoT (Internet of Things) permet de collecter des informations en temps réel, de stocker et de traiter ces informations massives avec fiabilité et souplesse puis de les transmettre à partir de mécanismes collaboratifs. Trois couches sont proposées dans cette architecture afin de fournir des services interdépendants en toute sécurité.

Timeline : 1993
Objet : l'aventure PowerPC

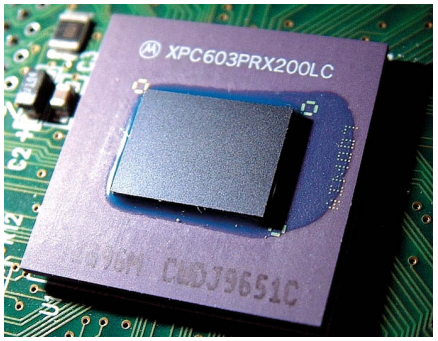
Processeur RISC concurrent du Pentium, le PowerPC a failli stopper la carrière d'Intel. Sauf que tous les acteurs concernés ont petit à petit jeté l'éponge. A part IBM.

En avril 1991, convaincus qu'ils doivent faire front commun contre les cloneurs de PC, les dirigeants d'Apple et d'IBM se rencontrent à Dallas. Du côté d'Apple, le PDG John Sculley, le DG Michael Spindler et l'ingénieur en chef Hugh Martin apportent dans leurs valises le projet Jaguar. Jaguar doit incarner le successeur du Mac : un système d'exploitation objet exécuté par des processeurs RISC. Le principe du RISC - un microcode simplifié pour être sûr que chaque instruction s'exécute en un seul cycle d'horloge - fut inventé par IBM en 1974, dans le cadre de son projet 801. Il s'agissait alors de mettre au point un processeur pour les commutateurs téléphoniques, sans performance de calcul extraordinaire, mais capable de copier ou d'additionner très rapidement des registres. Problème, Pink (Projet logiciel orienté objet, NDLR) n'est développé que par une équipe réduite et aucun processeur RISC monopuce testé (le Motorola 88100, l'Acorn ARM2, le MIPS R3000) n'est assez puissant pour l'exécuter. A défaut, ils présentent une version de Pink avec l'interface de MacOS 7 (mai 1991) émulée sur un IBM PS/2 Model 70 à base de processeur 486.

Du côté de la délégation IBM, le vice-président Jack Keuler s'accompagne de l'ingénieur Phil Euster, lequel a en charge la division des processeurs RISC. Depuis 1974, IBM a d'abord fait évoluer son 801 en le dotant de plusieurs puces fonctionnant en parallèle pour lui donner de la puissance de calcul (projets Cheetah en 1982, puis America/ROMP en 1985) afin d'aboutir, en 1990, à un POWER1 32 bits doté de 8 puces. Dans ses cartons, Phil Euster a maintenant un processeur RSC (Risc Single Chip) qui consiste à intégrer les 8 puces du POWER1 dans une seule. Pour IBM, il est financièrement intéressant que ce RSC soit utilisé ailleurs que dans ses propres machines Unix. Jack Keuler propose alors de vendre des stations avec le système Pink, à la condition qu'Apple vende des Macintosh avec le processeur RSC. L'affaire est intéressante, le POWER1 équipe les stations et les serveurs RS/6000 d'IBM, lesquelles sont quatre fois plus puissantes qu'un PC ou qu'un Mac. Mais Apple doute des capacités d'IBM à produire cette puce à grande échelle. John Sculley n'accepte le deal qu'à la condition que Motorola soit dans la boucle. L'affaire est entendue : le 2 octobre voit la naissance officielle de l'alliance AIM (Apple-IBM-Motorola), dont l'objet est de construire des ordinateurs à base de processeurs RSC. Pour gagner du temps, Motorola propose d'adapter au RSC le bus de son 88100, sur lequel Apple avait déjà travaillé. Le nouveau processeur est baptisé PowerPC, littéralement le POWER des ordinateurs personnels, même si, officiellement, le « PC » signifiait ici Performance Computing. La première puce, le PowerPC 601 cadencé à 50 et 66 MHz, est livrée en avril 1993.



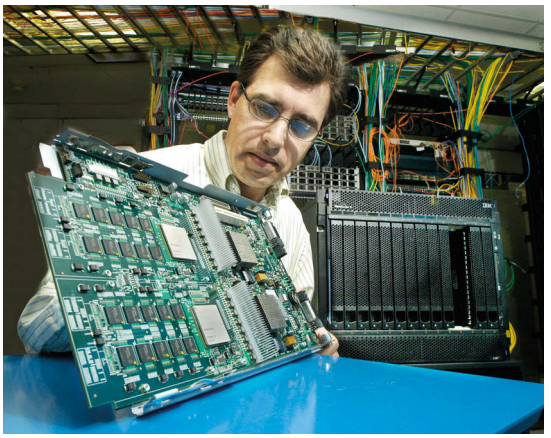
Le premier PowerPC, le PPC601.

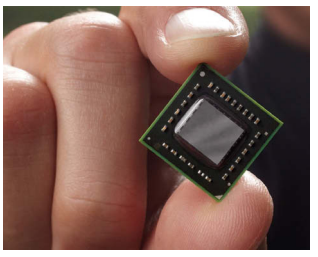


Pour concurrencer le Pentium II, le PPC 603 grimpait à 200 MHz

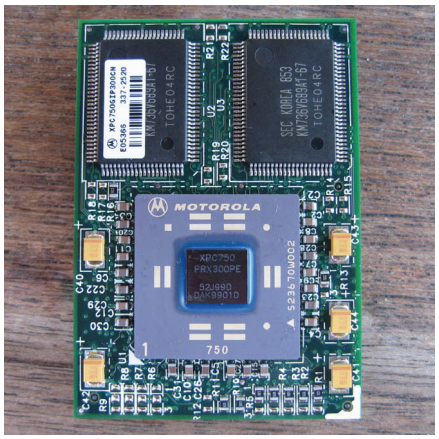
L'aventure des clones PowerPC fait long feu

En octobre 1993, IBM est le premier à proposer des stations RS/6000 (sous Unix) basées sur PowerPC. OS/2 serait déjà en cours d'adaptation pour fonctionner dessus, tout comme les prochains Windows NT 3.51 de Microsoft et Solaris 2.51 de Sun. Motorola fabrique des clones dans la foulée pour éponger la demande. Il faut dire que les machines dotées de ce PowerPC affichent dans les bancs d'essai des performances deux fois supérieures à celles des premiers PC équipés du dernier Pentium d'Intel à 66 Mhz. De plus, le PowerPC 601 a deux fois moins de transistors et coûte deux fois moins cher que son concurrent pour qui veut fabriquer des ordinateurs personnels « compatibles IBM. » Chez IBM, il y a l'idée de remplacer les compatibles PC, qui ont trop enrichi Intel, par des compatibles PowerPC. C'est le standard PReP (PowerPC Reference Platform), qui correspond à une carte mère de PC avec un socket PowerPC. Problème, quand Apple sort ses premiers PowerMac en 1994, ceux-ci ne sont pas compatibles PReP ; leurs cartes mères relèvent en effet plutôt du chipset historique des Macintosh que de celui des PC. IBM propose donc un second standard en 1995, le CHRP (Common Hardware Reference Platform), indépendant du chipset, mais reposant sur un firmware - OpenFirmware - qui fait l'interface entre la carte mère et l'OS installé. Dans la logique de CHRP, Apple vend des 1995 des licences de MacOS pour que des fabricants tiers puissent proposer des compatibles CHRP sous MacOS. Des clones de PowerMac, en somme. Hélas, il devient rapidement évident que les clones de PowerMac sont une erreur stratégique pour Apple : plutôt que d'étendre le nombre d'utilisateurs de MacOS, ils cannibalisent les clients du Mac original. De retour aux commandes d'Apple en 1997, Steve Jobs interdit leur fabrication. Pendant ce temps,

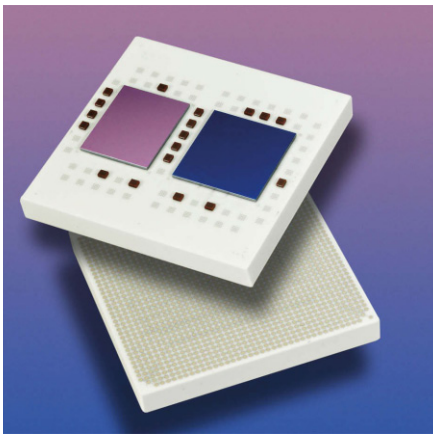

Le processeur Cell sera utilisé dans la Playstation 3 et les serveurs Unix en lame d'IBM.


Le G4 d'IBM est en fait un PowerPC 750 avec unité vectorielle pour consoles Nintendo.

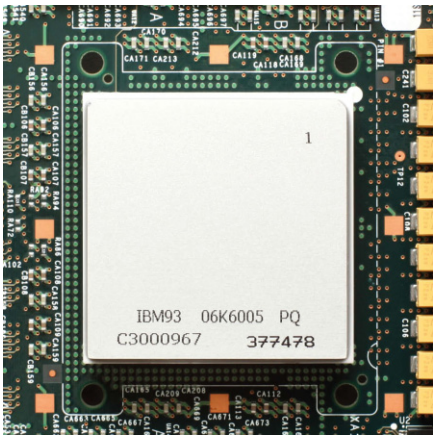
Le POWER1 avait huit puces.



Le PowerPC 750 était connu sous l'acronyme G3.



Le PowerPC 970, premier PowerPC 64 bits pour machines grand public.



Le POWER3, premier PowerPC 64 bits en une seule puce.

les carrières d'OS/2, de Windows NT et de Solaris sur machines CHRP font long feu : les utilisateurs n'achètent pas ces configurations tant que toutes les applications n'ont pas été recompilées en code PowerPC et celles-ci ne le sont pas tant qu'il n'y a pas assez d'utilisateurs sur ces machines. Le marché des clones CHRP meurt d'autant plus prématurément en 1997 que le fameux OS Pink (entretemps devenu Taligent) est avorté.

Trois types de PowerPC pour trois marchés

De 1993 à 2001, le PowerPC est divisé en trois familles. Pour les machines grand public, il s'agit d'un processeur 32 bits qui connaît, comme le Pentium d'Intel, quatre générations et dont les performances restent systématiquement les meilleures. La génération G4, connaît deux implémentations distinctes : les PPC7400 de Motorola (1999) à 1GHz apportent une unité vectorielle aux Mac, alors que les PPC750 d'IBM (1999) implémentent une unité vectorielle différente pour les trois prochaines générations de consoles Nintendo : la GameCube (PPC750CXe à 486 MHz, en 2000), la Wii (PPC750CL à 729 MHz, en 2006) et la Wii U (3 PPC750CL à 1,2 GHz, en 2012).

Simultanément, Motorola et IBM déclinent le PowerPC en modèles simplifiés, sans FPU (calculs flottants) ni MMU (mémoire virtuelle) pour l'électronique embarquée, en particulier celle des voitures, de l'aérospatiale, des imprimantes et des contrôleurs de disques. Ce sont les Motorola PowerQUICC et les IBM PowerPC 400 ; ils ne sont pas interchangeables dans leurs sockets et sont, par conséquent, concurrents. Du côté des machines Unix haut de gamme, IBM remplace le POWER1 par un POWER2 (1993) pour continuer à proposer les stations et les serveurs les plus rapides du monde. Les huit puces de ce POWER2 sont également combinées en un seul processeur P2SC en octobre 1996. Mais il coûte trop cher pour faire partie de la famille PowerPC. En 1997, IBM équipe ses machines AS/400 du processeur RS64 monopuce à 125MHz, un PowerPC 64 bits auquel est ajouté le jeu d'instruction 48 bits historique de ces machines. Ses RS/6000 passent la même année au processeur POWER3 à 200 MHz, un PowerPC 64 bits monopuce intégrant les instructions POWER qui n'ont pas été implémentées dans les PowerPC. Les RS64 et POWER3 sont regroupés en un seul POWER4 à 1,3 GHz en 2001.

Baroud d'honneur pour le PowerPC 64 bits

En 2002, il n'y a guère plus qu'IBM qui croit encore au PowerPC. Motorola n'a pas les fonds pour fabriquer un PowerPC 64 bits et songe déjà à se séparer de son activité processeurs. Celle-ci deviendra autonome en 2004, sous le nom de Freescale ; elle ne fabriquera plus que des puces pour électronique embarquée. De son côté, Apple voit rouge face à la montée en puissance d'Intel : son Pentium 4 Northwood grimpe à 2,2 GHz, soit deux fois la fréquence du PowerPC G4. En réponse, IBM débarrasse son POWER4+ à 1,9 GHz des instructions historiques POWER pour proposer à Apple le PowerPC 970 à 2 GHz, ou « G5 », le premier PowerPC 64 bits sur machine grand public. Apple devra encore attendre 2004 pour un PowerPC 970FX à 2,5 GHz - refroidi par liquide tellement il chauffe - alors que le Pentium 4 Prescott dépasse déjà les 3 GHz. C'en est trop : Apple jette l'éponge et annonce passer ses Mac aux processeurs x86 en 2005. Pourtant IBM se démène. Dès 2003, il met au point le processeur Cell pour la Playstation3 de Sony (2005). Il s'agit d'un PowerPC 970 à 3,2 GHz, dit PPE, qui intègre 8 unités 128 bits vectorielles. IBM reprendra cette puce dans ses lames serveurs sous Unix et Linux. Ce qui lui inspirera de relancer une plateforme pour cloneurs : le standard PAPR (Power Architecture Platform Reference) censé servir de référence pour construire des serveurs PowerPC sous Linux. En vain. Dans le même temps, Microsoft commande à IBM un PowerPC pour que sa future console Xbox 360 (2005) rivalise avec la Playstation 3. IBM lui fournira le Xenon, à savoir une puce disposant de trois cœurs PowerPC 970 à 3,2 GHz, mais sans les 8 unités vectorielles du Cell. En 2014, Sony et Microsoft ne renouvelleront pas le contrat ; leurs Playstation 4 et Xbox One sont désormais équipées de processeurs AMD x86 à 8 cœurs. Continuant d'améliorer seul son POWER (dont la version POWER8 à 12 cœurs et 5 GHz est sortie durant l'été 2014), IBM retente depuis 2013 de susciter l'envie des cloneurs autour d'une plateforme PowerPC ouverte. Cela s'appelle désormais l'OpenPOWER consortium...

Yann **Serra**

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 exclusivement sur www.programmez.com

Une publication **Nefer-IT**
7 avenue Roger Chambonnet
91220 Brétigny sur Orge
redaction@programmez.com
Tél. : 01 60 85 39 96

Directeur de la publication & rédacteur en chef : François Tonic

Ont collaboré à ce numéro : S. Saurel, Yann Serra, F. Bordage, J. Chatard
Secrétaire de rédaction : Olivier Pavie
Experts : H. Carnicelli, L. Ellerbach, S. Civetta, F. Dupont, T. Lebrun, J. Guittard, O. Guilloux, J. Antoine, T. Desmas, M. Frappat, F. Hebrard, M. Hubert, C. Villeneuve, S. Goudeau, B. Talmard, E. Briand, G. Egron, J-B Claramonte, P. Gilot, S. Vidouse, J. Devillard, M. Labelle

Crédits couverture : © 10-18-13 © FulopZsolt

Maquette : Pierre Sandré

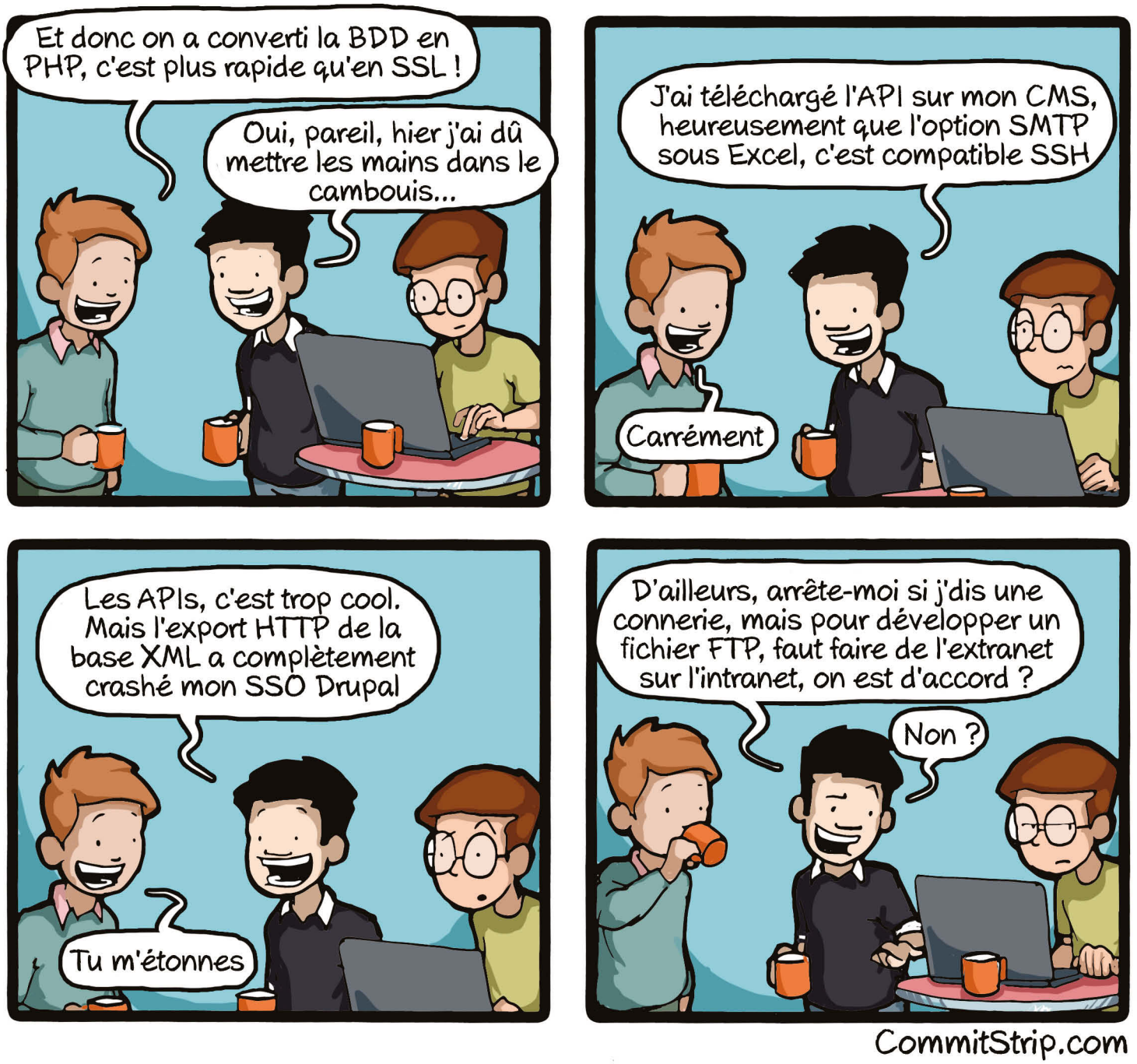
Publicité : PC Presse,
Tél.: 01 74 70 16 30, Fax : 01 41 38 29 75
pub@programmez.com

Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles, Belgique.
Marketing et promotion des ventes :
Agence BOCONSEIL - Analyse Media Etude
Directeur : Otto BORSCHA oborscha@boconseilame.fr
Responsable titre : Terry MATTARD
Téléphone : 09 67 32 09 34

Contacts
Rédacteur en chef : ftonic@programmez.com
Rédaction : redaction@programmez.com
Webmaster : webmaster@programmez.com
Publicité : pub@programmez.com
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1215 K 78366 - ISSN : 1627-0908

“Quand un chef de projet parle avec un chef de projet.”



Chaque semaine, de nouvelles aventures ! www.commitstrip.com

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

Mais aussi sur le web

Ou encore sur votre tablette

PROLONGATION JUSQU'AU 18 DÉCEMBRE 2014

UNE TÉLÉ POUR 1 EURO DE PLUS

INCURVÉ

Télé SAMSUNG

Pour 1 Euro de plus, choisissez :

Télé Samsung écran **Incurvé**
121 cm HD Réf: UE48H6800

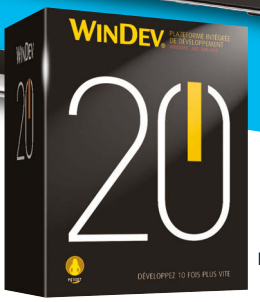
ou

Télé Samsung 4K ultra HD
127 cm, **LED**
Définition d'image fabuleuse: 3.840 x 2.160
Réf: UE50HU6900

ou

Télé Samsung 140 cm 2D/3D HD
Réf: UE55H6400

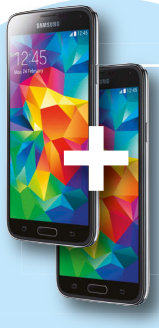
COMMANDEZ WINDEV 20 (OU WEBDEV OU WINDEV MOBILE) CHEZ PC SOFT ET RECEVEZ UN SUPERBE MATÉRIEL AU CHOIX POUR «1 EURO DE PLUS»



Elu
«Langage le plus productif du marché»

WINDEV®

Ou vous pouvez également choisir :



(x2) Nouveau smartphone Samsung Galaxy S5

Configurations détaillées sur www.pcsoft.fr

(x2) Nouvelle tablette Samsung Galaxy Tab S 10,5p

Configurations sur www.pcsoft.fr

Le tout nouveau Samsung Galaxy Alpha

Configurations détaillées sur www.pcsoft.fr

Samsung Galaxy Gear 2 Lite + Smartphone Galaxy Note 4

Configurations sur www.pcsoft.fr

Ou encore un PC portable DELL ou un PC de bureau DELL ou une station de travail DELL

Pour bénéficier de cette offre exceptionnelle, il suffit de commander WINDEV Mobile 20 (ou WINDEV 20, ou WEBDEV 20) chez PC SOFT au tarif catalogue avant le 14 décembre 2014: pour 1 Euro de plus, vous recevrez alors le ou les magnifiques matériels que vous aurez choisis. Offre réservée aux sociétés, administrations, mairies, GIE et professions libérales, en France métropolitaine. **L'offre s'applique sur le tarif catalogue uniquement.**
Voir tous les détails et des vidéos sur : **www.pcsoft.fr** ou **appelez-nous.**
Le Logiciel et le matériel peuvent être acquis séparément. Tarif du logiciel au prix catalogue de 1.650 Euros HT (1.973,40 TTC). Merci de vous connecter au site www.pcsoft.fr pour consulter la liste des prix des matériels et les dates de disponibilité. Tarifs modifiables sans préavis.

Descriptif technique complet des matériels sur www.pcsoft.fr

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

Fournisseur Officiel de la Préparation Olympique

www.pcsoft.fr