PROGRAMMEZ.COM

PROGRAMMEZ!

Mensuel n°178 - Octobre 2014

le magazine du développeur

Cloud Computing
Les conteneurs Docker

Drupal : programmation avancée

Un code + **écologique**
= un code + **efficace**

02-14-07 © Inok

X-Files : les origines du processeur

Carrière : quel salaire pour le développeur ?

Créer des apps pour **Chrome**

JavaScript : utiliser des lambda !

Késako un **Program Manager** ?

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

M 04319 - 178 - F: 5,95 € - RD

WINDEV 19
DÉVELOPPEZ 10 FOIS PLUS VITE

WINDEV : Logiciel professionnel de développement multi-plateformes

N°1 en France

Vous méritez le meilleur, vos équipes méritent le meilleur pour être le plus efficace possible, quel que soit le matériel sur lequel vos applications vont fonctionner en natif sur: **PC**, **tablette**, **smartphone**, et **Internet**.

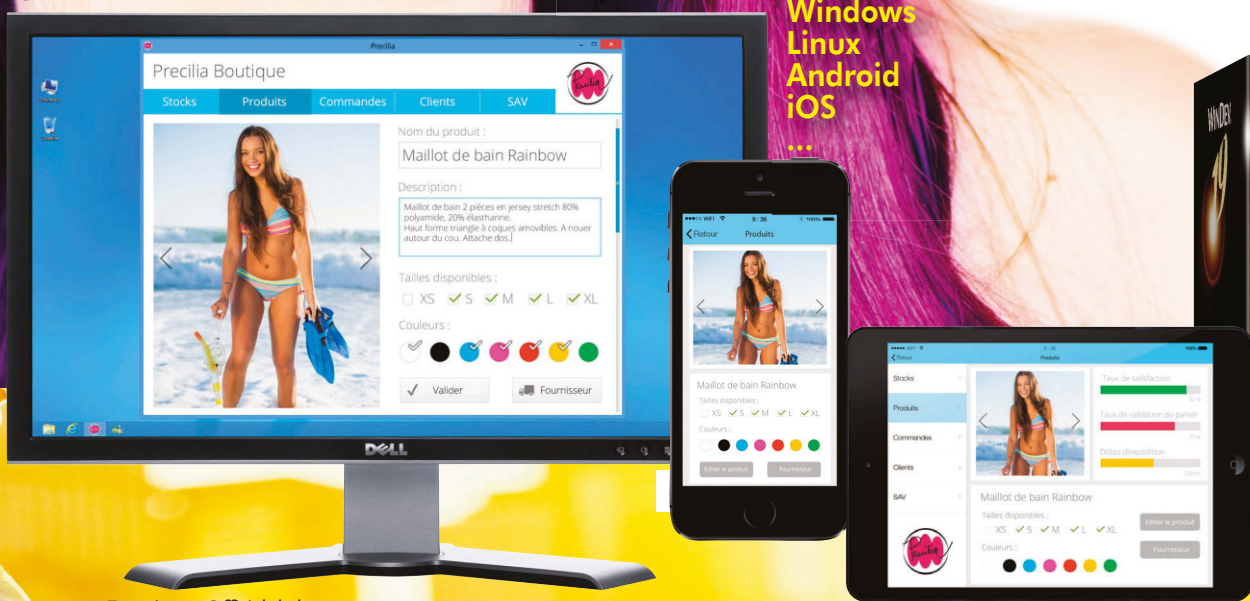
Incroyable : grâce à **WINDEV 19**, **WEBDEV 19** et **WINDEV Mobile 19**, **les applications que vous écrivez sont portables**.

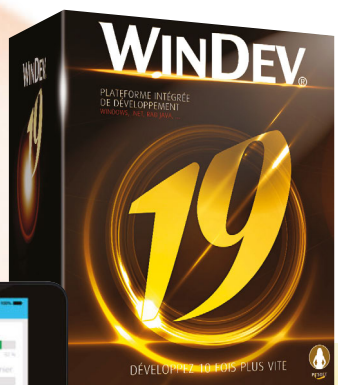
Vous réutilisez votre code, vos fenêtres, vos états... dans tous les environnements: **Windows**, Mac, Linux, **Android**, iOS (**iPhone**, **iPad**), **Windows Phone**, pour des applications natives et pour des sites **Internet**, avec les données en local, sur serveur local ou distant, ou encore dans le **cloud**.

Dans l'intérêt de votre entreprise, dans l'intérêt de vos utilisateurs et de vos clients, commandez aujourd'hui votre **WINDEV 19** !

Les applications que vous développez sont **natives** et **portables**

Windows
Linux
Android
iOS
...





Environnement de développement professionnel depuis 20 ans

Dossier + DVD + Témoignages sur simple demande (gratuit)

Fournisseur Officiel de la Préparation Olympique

WINDEV AGL N°1 en FRANCE

www.pcsoft.fr

Des centaines de références sur le site

document non contractuel

édito

Le code vieillit mal... ou pas.

J'ai toujours été intrigué par les « vieux » codes. Des millions de codes Cobol, Fortran fonctionnent aujourd'hui partout dans le monde. Certaines lignes remontent à 20-30 ans et elles tournent toujours, avec peu de modifications.

Peut-on dire la même chose des langages récents ?

On aurait presque tendance à l'oublier, mais nous produisons des millions de lignes de codes chaque année, soit des nouvelles, soit des anciennes modifiées et adaptées. Mais cela pose tout de même une question fondamentale : que fait-on du vieux code ? Un exemple très simple : essayer de faire fonctionner un site web HTML 3.2 sur la version la plus récente d'un navigateur. Vous aurez bien des surprises. Ou encore un site optimisé IE 6... Son fonctionnement sera tout ou partie altéré ou carrément non supporté.

A contrario, un code développé dans un langage normalisé (et à la condition de respecter rigoureusement la norme) fonctionnera sans réel problème. C'est notamment le cas pour le C et le C++, deux langages normalisés. Ces langages évoluent régulièrement (mais lentement) et un code vieux de 10 ou 20 ans pourra être compilé et exécuté. Les compilateurs récents pourront générer des warnings, si on pousse les options du compiloou si on utilise des librairies externes qui peuvent poser problèmes (recompilation ou version plus récente). Ainsi un sample code VC++ MFC de 1994 généré avec un Visual Studio 2013 tourne sur Windows 8.1 (merci à Christophe P. pour avoir fait le test).

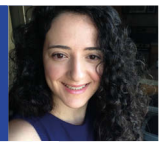
Du vieux code Java pourra fonctionner sur les plus récentes JDK (merci la rétrocompatibilité). Il y aura sans doute des adaptations et des modifications à faire selon l'ancienneté du code. Même chose pour du code C# 2.0. Et de nombreuses applications codées en VB6 tournent toujours même sous Windows 8.1. Je suis sans doute un peu médisant sur mon affirmation de départ : le code vieillit mal. Nous reviendrons sur cette question très prochainement...

Ce mois-ci, nous vous proposons un joli panorama technique : les applications Chrome, déployer PHP sur Azure, comment faire du fonctionnel en JavaScript, du perfectionnement sur Drupal, et la découverte d'un des projets open sources les plus actifs du moment, Docker.

Bon code !

François Tonic
Directeur de la publication & rédacteur en chef
ftonic@programmez.com

sommaire



À LIRE DANS LE PROCHAIN NUMÉRO
n° 179 en kiosque le 31 octobre 2014

CLOUD & DÉVELOPPEUR
Comment et pourquoi le cloud computing bouleverse le développement et les applications ?

Wordpress
Wordpress est une plateforme de CMS puissante et extensible. Comment utiliser toute sa puissance ?

iBeacon
Découvrez une technologie très prometteuse.

Quel salaire pour un développeur ?

Urban Linker, recrutement en ligne, a publié fin août une étude sur les salaires des métiers du numérique. Cette étude a été menée sur 1 an auprès de 300 recruteurs (toutes catégories de sociétés). Le recruteur a supprimé les technologies trop anciennes telles que CSS2. Ces salaires moyens constituent un indicateur, une tendance et non des statistiques officielles. Salaires exprimés en € / brut annuel. Globalement, le salaire du développeur reste quasi stable mais de nombreuses différences existent entre les profils.

Comme toujours, ces chiffres sont à nuancer et la différence Ile de France / province sera à considérer. D'autre part, le niveau salarial variera aussi selon l'employeur (PME, grande entreprise, SSII, etc.).
Source : http://www.urbanlinker.com/salaire-moyen/salaire-technique-2014/

Développeur Java, .Net, Ruby : bilan mitigé

Le développeur "standard" demeure une valeur sûre du marché. Le recruteur met en avant Ruby mais ce profil n'est pas le plus demandé.

DÉVELOPPEURS BIG DATA

Rémunération	Data Analyst	Data Scientist	Hadoop	Data Visualisation
Débutant 0 à 1 an	35-38 K€ =	50-60 K€ =	38-42 K€	36-38 K€
Intermédiaire 2 à 3 ans	38-45 K€ =	65-75 K€ =	42-48 K€	38-42 K€
Confirmé 4 ans et +	45-55 K€ =	> 75 K€ =	48-60 K€	42-50 K€

K€ brut/an – Source Urban Linker pour l'IDF

Pour un développeur arrivant sur le marché de l'emploi, Urban Linker propose une fourchette de 35 – 39 000 $, ce qui est tout de même une fourchette haute car on voit régulièrement des postes entre 28 – 33 000 €.
Les profils experts (+ 6 ans expérience / architecte / chef de projet) connaissent une petite baisse mais selon le profil et le langage, le salaire va de 46 à 85 000 €.

Développeur web : une baisse pour les profils basiques

Oui, le développeur demeure un profil incontournable mais attention à ne pas trop le surestimer, surtout pour le développeur – intégrateur HTML 5 / JS / CSS 3. Le salaire débutant démarre assez bas : 28 – 32 000 €. Ces profils subissent une baisse générale de 10 – 13 %. Ces profils ne sont pas les plus difficiles à trouver. Par contre, les compétences supplémentaires (Node.JS, responsive design, Angular, etc.) peuvent se monnayer un peu plus cher, jusqu'à + 10 %.

Des compétences à avoir ou à surveiller

compétences	tendance / opportunité	commentaires
cloud computing	●●●	incontournable pour le développeur, tous les développeurs. Compétence à acquérir rapidement si vous ne l'avez pas
architecte cloud computing	●●●	profil très expert, expérience d'architecte logicielle obligatoire
agilité	●●	aujourd'hui, tout développeur doit connaître les bases de l'agilité notamment Scrum
testeur / qualité logicielle	●	on en parle peu mais ces compétences et profils profitent d'une relative rareté
développeur web « classique »	●●	le développeur web est partout mais attention car on trouve tout et n'importe quoi dans les compétences et les profils. Démarquez-vous ! Montez en compétences.
responsive design	●●	Nous sommes mitigés sur cette fonctionnalité. On en parle. Il y a des postes et des demandes de compétences. A court terme, cette compétence peut être un + pour votre carrière, à long terme, nous sommes sans avis
développeur open source	●●●	l'open source est partout et la recherche de profils est toujours là. Sauf à être reconnu par vos pairs, cela ne va pas vous apporter une valorisation salariale forte

Développeur PHP : toujours un bon profil mais...

Selon Urban Linker, le développeur PHP demeure stable, à partir de 25-30 000 €, avec des sommets à 53 – 57 000 € pour les profils experts. Par contre, là encore, les compétences supplémentaires sur des outils et des frameworks peuvent faire la différence sur le niveau de salaire.

Développeur mobile : une tendance à la hausse

Urban Linker distingue les développements hybrides (multi-plateformes, technologies web, multi-terminal) et les projets natifs. Le natif reste légèrement plus rémunérateur que l'hybride qui utilise souvent des technologies web. Le profil natif voit des hausses, surtout pour les profils experts. Attention à ne pas sur-interpréter ces résultats. Oui, le développeur mobile est fortement demandé. Mais aujourd'hui, ce développeur doit être polyglotte et multiplateforme. Sa capacité à être un caméléon, à développer aussi bien sur Android, iOS que Windows Phone, sera un + indéniable (dans le monde du jeu mobile c'est même un profil standard). Un profil avec compétence, expérience, et à jour dans les technos (veille technologie obligatoire) pourra espérer un salaire plus élevé. L'étude estime qu'un profil senior peut atteindre 50 000 €, voir 60 000 € en natif. Le sommet se situe aux alentours de 80 000 € (profil architecte, + 6 ans expérience). Ce profil explose mais c'est maintenant qu'il faut se positionner si vous le souhaitez.

Développeur Big Data : un profil intéressant mais très limité

Dans Programmez !, nous avions déjà abordé les métiers liés aux données et généralement aux Big Data. Urban Linker liste plusieurs profils : data analyst, data scientist, Hadoop et data visualisation. Sans surprise, le data scientif est le haut de gamme, minimum 50 000 € (et encore, cela nous paraît un peu juste) et pou-

DÉVELOPPEURS PHP

	PHP	PHP + Framework MVC (Zend, Symfony, ...)
Débutant 0 à 1 an	25-30 K€ =	30-35 K€ =
Intermédiaire 1 à 2 ans	30-34 K€ =	35-40 K€ =
Confirmé 2 à 4 ans	35-40 K€ =	40-45 K€ =
Sénior 4 à 6 ans	40-45 K€ =	45-50 K€ =
Expert / Architecte 6 ans et +	45-53 K€ - 6.7 %*	50-70 K€ +4.3 %*
Chef de projet 8 ans et +		45-57 K€ +2%*

*K€ brut/an – Source Urban Linker pour l'IDF

DÉVELOPPEURS MOBILE

	Applications mobiles hybrides	Applications mobiles natives
Débutant 0 à 1 an	32-36 K€	34-40 K€ + 1.9 %*
Intermédiaire 1 à 2 ans	36-42 K€	38-44 K€ +2.5 %*
Confirmé 2 à 4 ans	42-46 K€	43-50 K€ +3.3 %*
Sénior 4 à 6 ans	46-50 K€	50-60 K€ +10%*
Expert / Architecte 6 ans et +	-	60-80 K€ +19.7 %*
Chef de Projet 4 à 8 ans	-	55-65 K€ +7 %*

*K€ brut/an – Source Urban Linker pour l'IDF

DÉVELOPPEURS FRONT END

Rémunération	Intégration / HTML5 / CSS3 / Javascript	HTML5 /CSS3 /Javascript / New frameworks JS + responsive web design	Dev fullstack JS NodeJs + framework front (Angular, backbone)
Débutant 0 à 1 an	28-32 K€ -10.4 %*	33-37 K€ =	36-38 K€
Intermédiaire 1 à 2 ans	32-38 K€ -9.1 %*	37-44 K€ =	38-45 K€
Confirmé / Chef de projet 2 à 4 ans	38-42 K€ -10.1 %*	44-49 K€ =	45-50 K€
Sénior 4 ans et +	42-45 K€ - 13 %*	49-46 K€ =	50-65 K€

*K€ brut/an – Source Urban Linker pour l'IDF

DÉVELOPPEURS JAVA . NET & RUBY

	JAVA	.NET	RUBY
Junior 0 à 1 an	35-39 K€ +1.4 %*	35-39 K€ +1.4 %*	32-36 K€
Intermédiaire 1 à 2 ans	39-42 K€ +1.3 %*	39-42 K€ +1.3 %*	36-45 K€
Confirmé 2 à 4 ans	42-46 K€ +1.4 %*	42-46 K€ +1.4 %*	45-50 K€
Sénior 4 à 6 ans	46-52 K€ +3.5 %*	46-55 K€ +1.4 %*	50-55 K€
Architecte / expert 6 ans et +	48-70 K€ - 1.7 %*	55-85 K€	
Chef de projet 4 à 8 ans	48-62 K€ - 4.3 %*	48-65 K€ - 1.7 %*	

*K€ brut/an – Source Urban Linker pour l'IDF

vant dépasser les 75 000 € (et même bien plus pour les profils les plus experts). Mais attention, le data scientist n'est pas un développeur; les postes sont peu nombreux et les candidats pas si fréquents. Nous parlons ici de statisticiens, de mathématiciens. Les profils orientés données ont toujours existé mais avec le Big Data, ces profils se multiplient. Les salaires sont plus élevés que pour un développeur classique. Nous sommes un peu sceptiques sur le poste purement Hadoop. Oui, Hadoop est le framework Big Data mais c'est un profil hybride développeur – données. Le salaire devrait être plus élevé. Compétence peu fréquente en France.

François Tonic

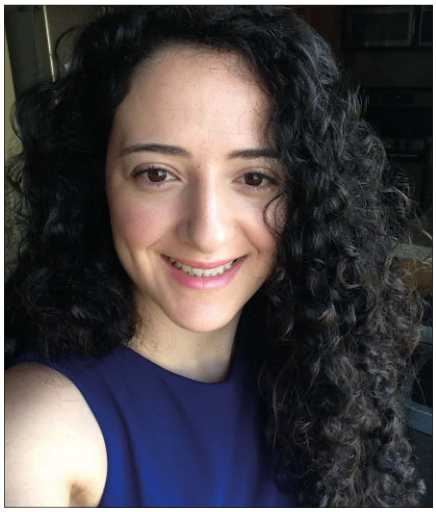
LES MÉTIERS UX & UI

	UX / UI	Architecte de l'information
débutant 0 à1 an	30-36 K€ =	35-40 K€ =
Intermédiaire 1 à 2 ans	34-40 K€ =	40-45 K€ +4.9%*
Confirmé 2 à 4 ans	39-46 K€ +2.4%*	46-42 K€ +5.4%*
Sénior 4 à 6 ans	43-42 K€ +6.5%*	52-65 K€ +6.3%*
Expert / Architecte 6 ans et +	48-70 K€ +4.4%*	65-75 K€ +3.7%*
Chef de Projet 4 à 8 ans	45-70 K€ =	60-75 K€ =

*K€ brut/an – Source Urban Linker pour l'IDF

Etats-Unis : des taxes mais un coût de la vie très différent

Souvent on cite en référence les salaires des ingénieurs logiciels américains. Mais qu'en est-il réellement ? Prenons un salaire médian de 100 000 $ brut annuel (bien entendu, le montant peut être largement au-dessus selon la société numérique et la région américaine).
Selon l'Etat où travaille notre ingénieur et la société, les charges et les taxes ne seront pas identiques. Les taxes fédérales s'élèvent à 19 % (le niveau des taxes est à nuancer selon l'Etat et la tranche de salaire).
Si notre ingénieur logiciel travaille dans la région de Seattle et dans une société généreuse en retraite et santé, il peut espérer un salaire net d'environ 70 000 $, si on considère une épargne retraite de 10 000 $ / an. L'assurance santé sera prise en charge par sa société. En Californie, notre développeur aura des taxes supérieures et donc un revenu moindre. Et s'il travaille dans une jeune startup, il devra payer lui-même les plans santés et retraites... Et le coût de la vie en Californie est plus élevé, surtout autour des grands pôles technologiques.
Il est cependant très difficile de comparer les salaires américains et français car le système des taxes n'est pas identique, ni pour la santé et la retraite. D'autre part, il n'y a pas de TVA mais chaque Etat applique un niveau de taxe local sur les marchandises.
Merci à Julien D. pour les chiffres et les précisions.



De « développeuse » à « program manageuse »

Ce que j'aime dans le développement, c'est de pouvoir sentir les résultats. Vous pouvez créer un produit ou un logiciel à partir de zéro. Vous pouvez suivre la vie de ce produit avec le code. C'est excellent !

C'est ce que j'ai fait depuis que j'ai eu mon diplôme d'ingénieur en informatique, mais il est temps pour moi de changer et de voir de l'autre côté de la vie d'un produit : le management. Après avoir été développeur pendant plus de 5 ans, j'ai décidé de passer à l'étape suivante dans ma carrière et de devenir « Program Manager », aussi appelé chef de projet/produit/programme en France. Ainsi, depuis quelques mois, je suis devenue Program Manager (ou PM) dans l'équipe en charge du noyau de Windows, Windows Phone et Xbox à Microsoft Redmond.

MAIS C'EST QUOI UN PROGRAM MANAGER ET QUE FAIT-IL EXACTEMENT ?

Pour expliquer les activités quotidiennes d'un PM dans le domaine du génie logiciel, je vais m'appuyer sur l'exemple d'un produit : une table à manger en bois. Nous construisons une table, nous avons une équipe de développeurs, de testeurs et de chefs de projet.
Je vais vous expliquer ce que le PM fait, mais cela ne signifie pas qu'un seul PM est en train de faire tout ce travail, ça pourrait être une équipe de PM, en fonction de la taille du projet et de l'entreprise. Chaque projet passe par différentes phases, même si les développeurs et les testeurs font exactement le même travail dans toutes ces phases le PM, de l'autre côté, fera un travail différent pour chaque phase. Voici les phases du projet : Fig.A.

Recherche et planification

Dans notre projet « construire une table », le PM commencera par une phase de « recherche et de planification » dans laquelle il répondra aux questions dont il a besoin pour faire la table que nous voulons :

- Quel type de bois devrait-on utiliser ?
- Quelle est la qualité que nous voulons ?
- Quelle est la qualité dont l'utilisateur a besoin ?
- Faut-il acheter du bois ou aller dans la forêt couper des arbres nous-mêmes ?
- Si nous voulons l'acheter, où l'acheter et pourquoi ?

Après avoir fait ces recherches, le PM passera à la phase de planification. Lors de la planification, il mettra toutes les grandes étapes nécessaires à la construction de cette table. Ces étapes doivent avoir des dates spécifiques.

Design

Dans cette phase, le PM demandera à l'équipe de design de designer la table en se basant sur ses recherches et son plan.
Dans la vraie vie, certaines équipes n'ont pas de designers et souvent le PM doit faire un design de base lui-même.
Cette phase doit également comporter la construction d'un prototype. Le prototype aidera l'équipe de développeurs à construire le produit convenablement.
Encore une fois, dans la vraie vie, certaines équipes n'ont pas le temps de construire un prototype et le PM finirait par construire le prototype lui-même.
Dans toutes ses enquêtes, le PM aura un plan avec une série de dates et de tâches à faire pour chaque personne de l'équipe et c'est à lui de gérer l'aboutissement de toutes les tâches comme il faut et à temps.

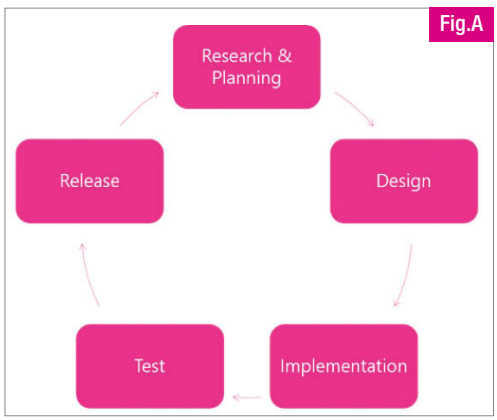


Fig.A

Implémentation et test

Cette phase est la plus longue. C'est la phase où l'équipe de développement codera le produit, et l'équipe de test testera le produit au fur et à mesure. Dans cette phase, le travail du PM consistera à :

- Créer un plan avec toutes les fonctionnalités qui doivent être faites,
- Prioriser les fonctionnalités importantes tout en restant en phase avec l'équipe de développement et en suivant leur progrès,
- Écrire un document de spécification afin de détailler les étapes de développement et de test.

Dans le monde réel, dans les grandes entreprises, tous les PM ont un ensemble de caractéristiques à gérer et auraient des discussions intéressantes afin de négocier les priorités de chacune de ces caractéristiques.

Lancement du produit

Cette phase est très différente d'un produit à un autre. Revenons à notre exemple de la construction d'une table. Avant le lancement de notre produit, nous pourrions inviter des clients aléatoires pour tester notre table et pour voir si le produit leur plaît ou pas. Sur la base de leurs évaluations, nous résoudrions les bugs et aurions une idée sur la façon dont nous pourrions procéder à la vente de la table. Le travail du PM dans cette phase sera de mener cette expérience utilisateur et de proposer des évaluations.

Qu'en est-il de la méthodologie Scrum agile ?

Dans les équipes qui suivent la méthodologie Scrum agile, ces phases se tournent autour et se divisent en petits cycles. Dans ce cas, le travail de suivi est moins important mais tous les autres aspects sont les mêmes.

Conclusion

Dans la vraie vie, il y a énormément de spécialités différentes pour un PM, de l'analyse de données BI à un PM technique (TPM) ou un PM business. Ce travail est également différent d'une société à l'autre, dans une très petite start-up, le PDG ferait parfois le travail du PM (et tout le reste aussi, s'il pouvait) alors que dans les grandes entreprises, nous trouvons un PM par fonctionnalité ou au moins un par équipe.

Soumow **Dollon**
http://soumow.dollon.net/blog

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

octobre

2 octobre : ReBuild à Nantes

Le 2 octobre, un grand événement communautaire autour des technologies Microsoft se tiendra à Nantes : 40 sessions techniques, 38 intervenants, 15 partenaires. C'est l'occasion de voir et de revoir des technologies et des outils Microsoft en dehors des événements parisiens. Cette année, on pourra voir Azure, SharePoint, Visual Studio Online, design d'interface, Kinect, SQL Server, Xamarin, Office 365, etc.
Site : https://lescommunautesms-public.sharepoint.com/rebuild

7 octobre : IBM SolutionsConnect 2014 à Paris

IBM organisera une journée de conférences le 7 octobre à la Cité des Sciences et de l'Industrie. De nombreux thèmes seront abordés : Big Data, Cloud, mobile, sécurité... Une partie développeur sera organisée : DeveloperConnect avec 3 grands thèmes (devops, ingénierie en continu et mobilité). Site : http://goo.gl/Co7dHk

10 octobre : dotGo

Vous connaissez peut-être le langage Go. Une grande conférence européenne se déroulera à Paris le 10 octobre prochain. Au menu : conférences, ateliers. Une excellente occasion de découvrir ce langage et d'échanger avec les développeurs l'utilisant. Site : http://www.dotgo.eu

17 octobre : Code of War 3e édition

Cette nouvelle édition se déroulera à Bordeaux le 17 octobre. « Code of War est une compétition de programmation en équipe,

Scala.IO : la conférence Scala !

Pour sa seconde édition, ScalaIO se déroulera les 23 et 24 octobre prochains et accueillera à Paris plus de 300 personnes venues de toute l'Europe pour assister aux présentations de près de 40 speakers talentueux, répartis comme l'an dernier sur 3 tracks. Au moins 30% des présentations seront données en Anglais et nous assurons une track pour un public anglophone.
Cette année, nous avons décidé d'investir un espace offrant un cadre convivial et chargé d'histoire, facile d'accès pour tous les participants d'où qu'ils viennent, et situé au coeur de Paris à proximité de ses quartiers les plus vivants. L'événement aura lieu dans les Salons du Tapis Rouge, un espace prestigieux situé au coeur de Paris en face de la mairie du Xème. Le lieu disposera d'une grande salle de conférence en sous-sol pour accueillir les keynotes et la track anglaise, d'une salle de 200 places et d'une salle de 80 places pour les 2 autres tracks.
ScalaIO est une conférence centrée principalement sur le langage Scala mais nous souhaitons une ouverture sur tous les ecosystemes connexes. Nous voulons que ScalaIO permettre aux gens d'apprendre et d'échanger autour de Scala mais aussi de partager avec les autres communautés. Par ailleurs cette année nous mettons particulièrement l'accent sur le domaine du (big)data processing avec des présentations, des ateliers et retours des experiences sur des mises en oeuvres d'architectures distribuées s'appuyant sur les frameworks de l'ecosysteme autour du langage. Le contenu de la conférence sera annoncé debut septembre. Les places sont actuellement en vente sur le site avec un tarif pour les 2 jours à 350 euros. Les préventes ont été écoulées en une matinée et nous espérons avoir dimensionné cette année la conference afin d'accueillir tout le monde.
Site : http://scala.io/

23 & 24 octobre : Forum PHP Paris 2014

Le grand événement de la communauté PHP revient fin octobre à Paris. Cinq thèmes seront abordés et montrés : agilité, devops, écosystème PHP, cloud et retour d'expérience. Des journées à ne pas manquer. Site : http://afup.org/pages/forumphp2014/

29 & 30 octobre : Blend Web Mix à Lyon

Durant deux jours, Lyon accueille une conférence Web : conférences, ateliers, rencontres avec les startups, speed démo, soirées, etc. C'est l'occasion de voir quelques projets de recherche autour du Web et du futur du Web. L'édition 2013 avait rassemblé + de 850 personnes ! site : http://www.blendwebmix.com

novembre

Green Code Lab Challenge 2014 :

l'événement de l'éco-informatique aura lieu du 26 au 28 novembre. Comment faire un code et un logiciel écoresponsable ? Toutes les réponses au Green Code Lab ! site : http://www.greencodelab-challenge.org/GCL2014/

Drupagora 2014 : l'évènement français

Drupal se déroulera le 14 novembre à Paris. Ce sera déjà la 4e édition. Site : http://www.drupagora.com/

dotJS + dotCSS : le 15 novembre, une

double journée technique sur JavaScript et les CSS. Les ateliers seront les vedettes de cette journée. Site : http://www.dotjs.eu/workshops

fOSSA 2014 : du 19 au 21 novembre

Cette nouvelle édition du Free Open Source Software for Academia aura comme fil rouge : les transitions en cours, leurs enjeux et le rôle de l'openness dans l'émergence de nouveaux écosystèmes. Si au départ fOSSa abordait essentiellement le logiciel open source et ses modèles, les dernières éditions élargissent résolument le spectre en abordant les technologies ouvertes, les écosystèmes qui y sont associés et les transformations de la société qu'elles engendrent dans leurs aspects économiques, sociaux, politiques, scientifiques, techniques, etc. On constate actuellement que les modèles ouverts investissent, réinvestissent ou s'apprêtent à investir tous les domaines possibles et imaginables, qu'il s'agisse bien sûr du logiciel mais aussi de la science ou voir même de la monnaie ;-) L'évènement se déroulera à Rennes. Site web : https://fossa.inria.fr/program/

Mobile Dev Day à Mons (Belgique)

Le 27 novembre, venez découvrir le développement mobile et embarqué sur les plateformes Microsoft. Les principaux thèmes seront : mobilité, interface et design, Cloud Computing et internet des objets. 11 sessions sont prévues dans la journée. Pour en savoir plus : ww.mobiledevday.be

Montée en charge linéaire et extrêmement performante

NCache™ 100% .NET

Pour applications .NET et Java
(supporté sur Windows Azure et Amazon AWS)

TayzGrid™ 100% Java

Les données en cache (via NCache), réduisent les accès coûteux en base, et permettent à vos applications de monter en puissance vers “extreme transaction processing” (XTP). TayzGrid est une implémentation native 100% Java de NCache.

Cache distribué en mémoire

- Extrêmement rapide et montée en charge linéaire avec 100% uptime
- Topologie en Miroir, Répliquée, Partitionnée et Cache Client
- NHibernate et Entity Framework cache niveau 2

Optimisation ASP.NET de Web Farms

- ASP.NET Session State cache
- ASP.NET View State cache
- ASP.NET Output Cache provider

Memcached Wrapper Fourni

Partage de données en mode Runtime

- Notifications d'événements puissants pour le partage de pub / sub données

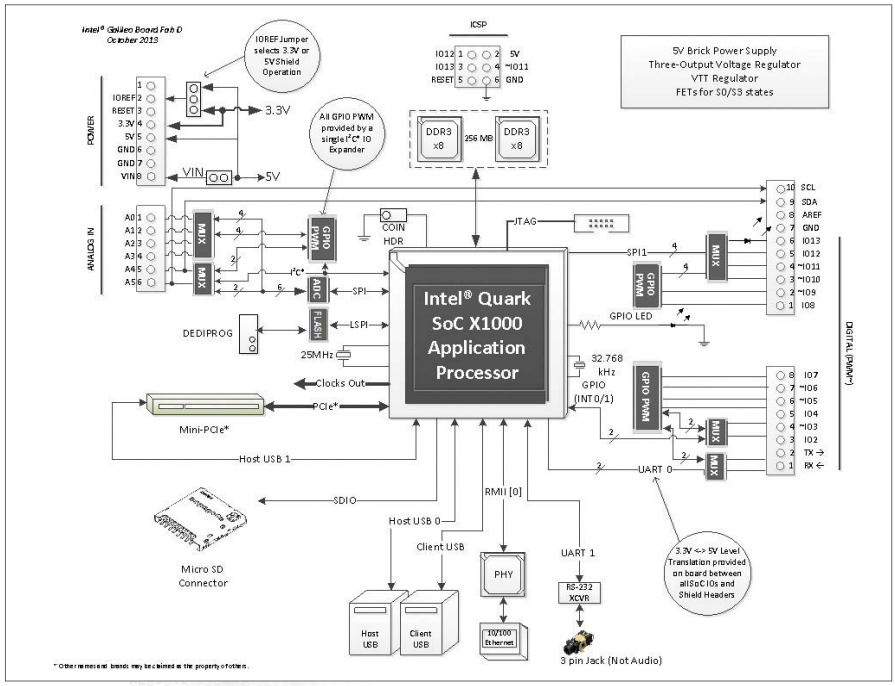
Alachisoft
sales@alachisoft.com
US: +1 (925) 236 3830
Siège de l'entreprise

Télécharger un essai GRATUIT!
www.alachisoft.com

RedFabriQ
info@redfabriq.com
Tel: +33 1 40 16 07 89
Distributeur et intégrateur en France

SoC : la pierre angulaire de l'informatique wearable et connectée

System on a Chip (système sur une puce), ou SoC, est sans doute une des innovations les plus marquantes des processeurs et des cartes mères. L'idée est d'intégrer dans une seule puce des processeurs (CPU / GPU), des interfaces de communication, des contrôleurs, des capteurs divers et variés, etc.



Ainsi, si on regarde le SoC du Raspberry Pi, un Broadcom BCM2835, il inclut notamment :

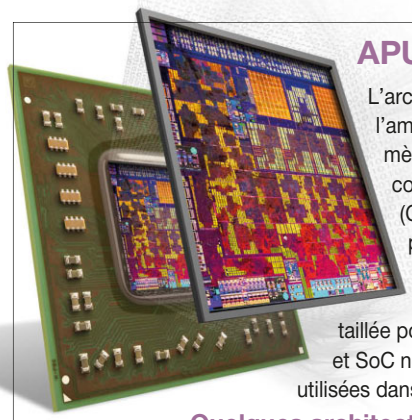
- Processeur ARM
- co-processeur VideoCore
- Encodage / décodage vidéo 1080 / Full HD / H.264
- OpenGL-ES
- Gestion sortie vidéo

La carte miniature Intel Edison repose sur un SoC maison : le Quark. Celui-ci intègre deux cœurs, le wifi et le bluetooth, mémoire DDR, GPU, support PCIe, USB, Ethernet, GPIO, SD, etc. Le tout en basse consommation, comme tous les SoC. Car un des enjeux est l'optimisation énergétique car le SoC doit fonctionner dans des environnements contraints.

Bref vous l'aurez compris, le système complet sur une puce joue un rôle crucial sur les terminaux mobiles et les cartes de type Arduino, Raspberry, Edison, les objets connectés. Plus le SoC intègre d'éléments, plus les ingénieurs pourront optimiser la taille de la carte mère. Dans le cas des objets connectés et du wearable en général, la taille de l'électronique est cruciale. Ainsi il est possible de limiter le nombre de contrôleurs externes au processeur (regarder la taille d'Edison). Cela signifie aussi que le SoC fournit des ressources que l'on ne peut pas étendre, telle que la mémoire vive, le GPU, les réseaux. Contrairement à des PC extensibles, le SoC est figé et il ne se remplace pas. Il est soudé à la carte. Cette miniaturisation impose donc des concessions et des compromis.

Et le développeur, il compte pour du SoC ?

Si le SoC devient très complexe, ce n'est pas forcément le cas pour le développeur. En réalité, les systèmes au-delà s'appuient sur une optimisation, mais aussi et surtout sur des pilotes pour assurer les échanges entre le matériel et le logiciel. Rien de nouveau sur ce point.



APU : une architecture magique ?

L'architecture APU (Accelerated Processing Unit) a l'ambition de proposer une autre architecture des cartes mères et des ordinateurs en général. Il s'agit de combiner le processeur et le processeur graphique (CPU + GPU). Cette approche doit supprimer les ponts entre CPU et GPU qui génèrent pertes de puissances et latences, les deux processeurs étant sur le même « support ». Cette architecture est taillée pour les calculs, la simulation. Mais les mondes APU et SoC ne sont pas éloignés. Surtout, ces approches sont utilisées dans les tablettes et certaines consoles de jeux.

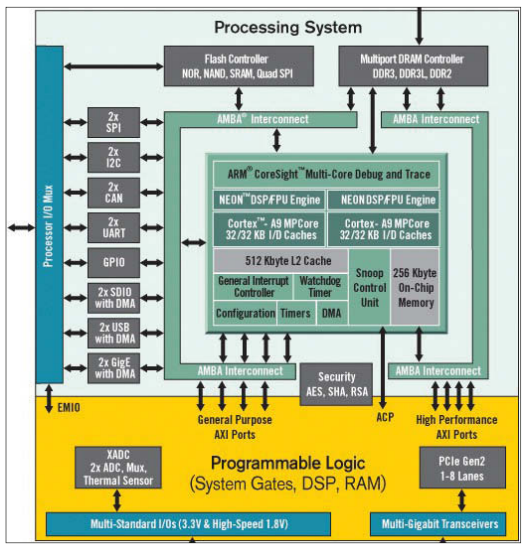
Quelques architectures :

- AMD Fusion : cette architecture évolue rapidement. Le fondeur annonce les APU Beema et Mullins
- NVidia Tegra K1

Côté développeur, les API, SDK et autres librairies cachent la complexité du SoC. Ainsi, Nvidia par son SoC Tegra propose aux développeurs :

- Une suite d'outils dédiés : Tegra Android Development Pack
- Une image système Linux optimisée et dédiée Tegra
- GameWorks : ressources et outils orientés jeux et 3D

Mais fondamentalement, le SoC seul n'a pas d'utilité. Ce n'est qu'une puce. Il faut tout l'environnement matériel pour qu'il soit utilisable et utile. Par exemple, pour pouvoir initier un développement Edison, il faut typiquement passer par une carte d'extension de type Arduino / Galileo pour flasher le système, installer les outils et développement les applications.



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

NEW HOSTING

AVEC LES MEILLEURES APPLICATIONS !

WordPress & 140 Apps populaires

- Installation facile avec l'assistant WP
- **Support dédié 24/7 assuré par les experts 1&1**
- Notifications de sécurité et mises à jour automatiques
- Performance optimale : 2 Go de RAM garantis
- Service complet pour 140 Apps & CMS réputés : WordPress, Joomla!™, Drupal™, TYPO3, Magento®
- Versions d'évaluation disponibles

Outils de référence

- **PHP 5.4/5.5/5.6**, Perl, Python, Ruby, tâches cron
- 1&1 Mobile Website Builder
- NetObjects Fusion® 2013 – 1&1 Edition

Marketing efficace

- 1&1 Référencement Pro
- 1&1 Newsletter Tool
- 1&1 WebStat
- Crédits Facebook®

Infrastructure High-Tech

- Disponibilité maximale grâce à la **géo-redondance**
- Connectivité > 300 Gbits/s
- 1&1 CDN powered by CloudFlare® (23 PoPs)

Tout inclus

- Nom de domaine : .fr, .com, .net, .eu, .org, .info
- Ressources illimitées : espace Web, trafic, comptes email et bases de données MySQL

PACKS COMPLETS POUR PROFESSIONNELS

À partir de **0,99** € HT/mois*

1 MOIS POUR ESSAYER

1 CLIC POUR CHANGER DE PACK

1 APPEL UN EXPERT VOUS RÉPOND

0970 808 911
(appel non surtaxé)

1&1

1and1.fr

*30 jours « satisfait ou 100 % remboursé ». Les packs hébergement 1&1 New Hosting sont à partir de 0,99 € HT/mois (1,19 € TTC) la 1ère année au lieu de 1,99 € HT/mois (2,39 € TTC) pour un engagement minimum de 12 mois. À l'issue des 12 premiers mois, les prix habituels s'appliquent. Offres sans durée minimale d'engagement également disponibles. Offres à durée limitée et soumises à conditions détaillées disponibles sur 1and1.fr. Rubik's Cube® utilisé avec l'accord de Rubik's Brand Ltd.

Bug : Gradle sur Eclipse

Gradle est un moteur de production fonctionnant sur la plateforme Java. Il permet de construire des projets en Java, Scala, Groovy, voire C++.

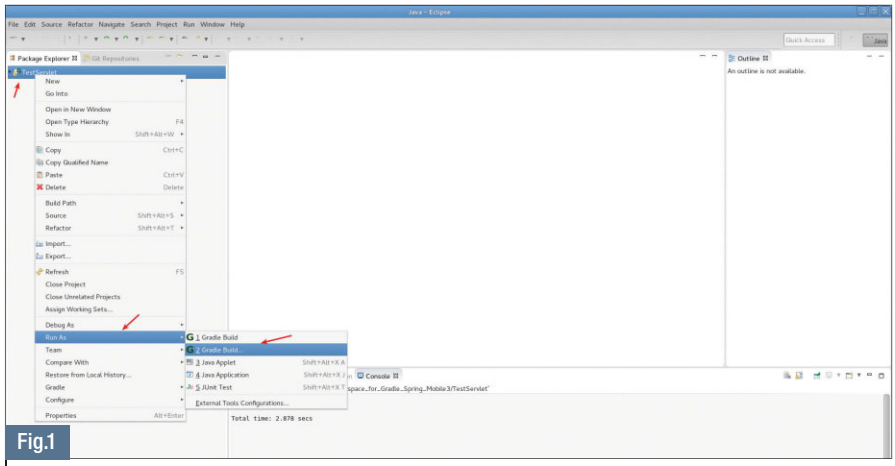


Fig.1

Exécuter un projet Gradle

Gradle allie les atouts de Apache Maven et Apache Ant : il combine l'utilisation de conventions à la manière de Maven (convention plutôt que configuration) avec la flexibilité de Ant pour décrire les tâches de construction, avec une cohérence forte dans l'interface de programmation des tâches. (source : http://fr.wikipedia.org/wiki/Gradle)

Logiciels utilisés

- Eclipse Standard 4.4
- Java SE 8u11
- Gradle IDE 3.6.0.201407080553-RELEASE
- Fedora 20 (Linux), 64 bits

Projet

Nous avons un projet Gradle "TestServlet" dans Eclipse.
Pour l'exécuter :
Clic droit sur le projet > Run As > Sélectionnez > Gradle Build (avec ou sans 3 points) **Fig.1.**
Dans le dialogue suivant, entrez des commandes destinées à Gradle (ici jettyRun) > puis Run, **Fig.2.**
Un message d'erreur est affiché. Apparemment Gradle n'a pas trouvé le JDK installé sur le système.
Note : *JDK est bien installé et JAVA_HOME est configuré !!!* **Fig.3.**

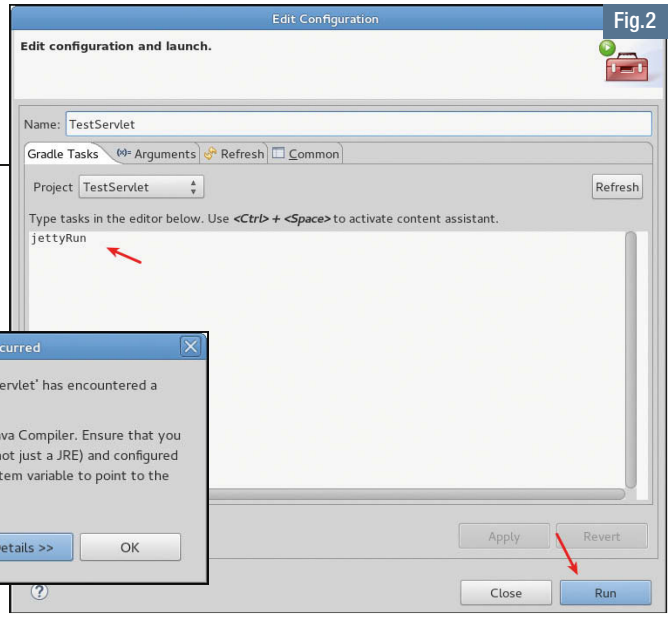


Fig.2

Executer jettyRun

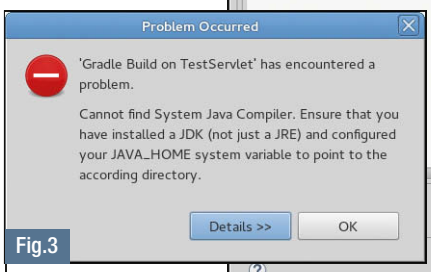


Fig.3

Le plug-in du Gradle n'a pas trouvé le JDK !!!

Solution

Fermez la fenêtre d'erreur (ou le message d'erreur, précédent) en cliquant sur " OK "
Recommencez l'exécution qui consiste à cliquer avec le bouton droit sur le projet > Run As > Gradle Build.
Maintenant, au lieu de cliquer sur le bouton " Run ", sélectionnez l'onglet " Arguments ". Désélectionnez " Use Gradle wrapper's default ".
Sélectionnez la case " Workspace JRE : " et sélectionnez le/un JDK installé sur votre système. Maintenant cliquez sur " Run " **Fig.4.**
Et voilà, **Fig.5.**

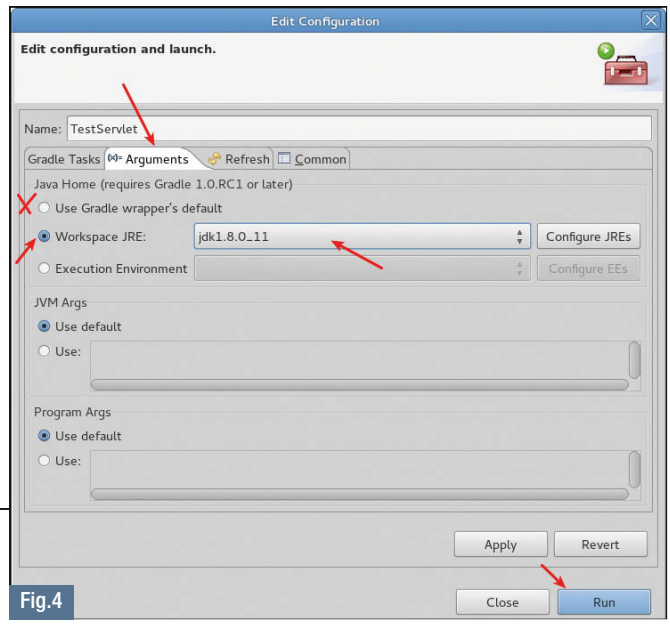


Fig.4

Configurer le JDK

Exécution OK

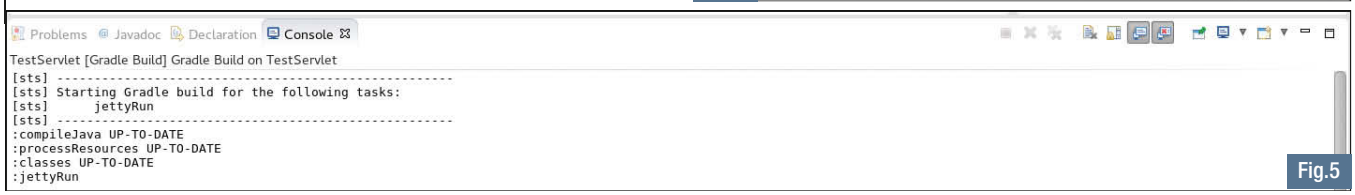
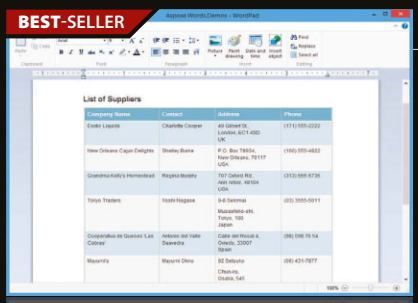


Fig.5

Kaesar **ALNIJRES**
Développeur Java
http://java-javafx-iipt.blogspot.fr/

ComponentSource®
The Definitive Source of Software Components
www.componentsource.com

BEST-SELLER

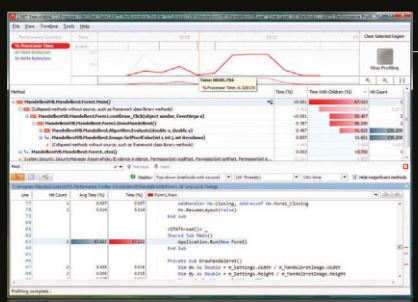


Aspose.Words for .NET | à partir de € 750

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



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

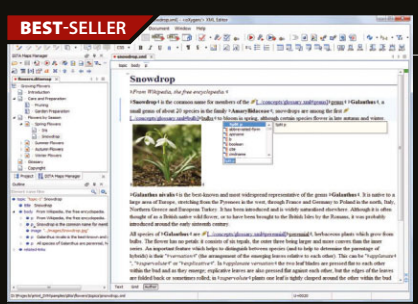
DevExpress DXperience | à partir de € 1 126

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

BEST-SELLER



oXygen XML Editor Professional | à partir de € 366

<oXygen/> XML Editor

Éditeur XML multiplateforme supportant la plupart des technologies XML.

- Distribuez/actualisez facilement des plug-in/frameworks pour Oxygen
- Simplifiez la configuration des projets XML avec Master Files
- Affichage des modifications et commentaires de révision dans légendes
- Support d'édition visuelle convivial pour DocBook, DITA, TEI, XHTML
- Validez les documents XML avec les schémas XML, Relax NG, DTD, NVDL et Schematron

© 1996-2014 ComponentSource. Tous droits réservés. Tous les prix sont corrects au moment de la presse. Prix en ligne mai différentes de celles décrites en raison de fluctuations quotidiennes et remises en ligne.

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

Nous acceptons les bons de commande. Contactez-nous pour demander un compte de crédit.

VISA MasterCard

Ecoconception Web
Architecture technique : retour aux sources

Les étapes qui précèdent la phase de développement sont les plus importantes pour réduire les impacts environnementaux d'un logiciel ou d'un site web, notamment le choix d'une technologie et d'une architecture adaptées qui compensent parfois un code peu efficient.

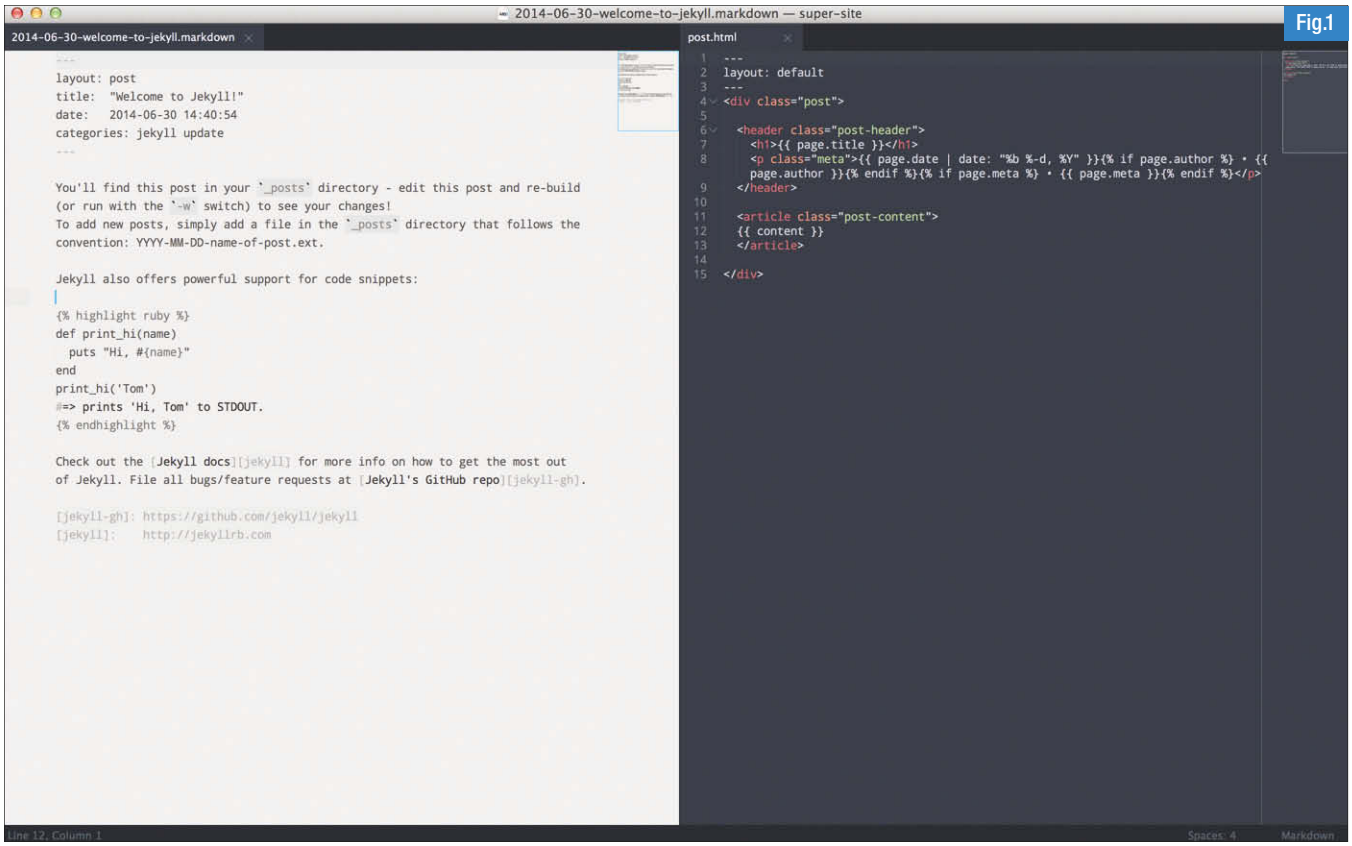
Une fois les besoins des utilisateurs détaillés dans les spécifications générales (voir notre premier article dans le numéro septembre de Programmez !), il est temps de choisir une plate-forme de développement et d'exécution ainsi qu'une architecture applicative adaptées aux objectifs du projet.

A cette étape, il est indispensable de ne pas tomber dans la facilité qui consiste à utiliser systématiquement le framework ou le CMS que l'on maîtrise par cœur. Car, la plupart du temps, cela revient à utiliser un éléphant pour accoucher d'une souris. Certains sites à très fort trafic tels que Culture.fr en France, HealthCare.gov aux Etats-Unis ou bien encore le site de financement de la campagne de Barack Obama ont appris cette leçon à leur dépend. Dans le cas de HealthCare.gov, le choix d'un CMS s'est révélé catastrophique. Malgré de nombreuses optimisations, cette architecture, trop complexe et gourmande en ressources, n'a pas permis de soutenir la montée en charge. Quelques mois plus tard, il a été reconstruit avec une approche complètement différente : la génération de pages statiques à l'aide de Jekyll, un outil spécialisé dans ce type d'approche. Résultat ? Le site supporte désormais de très grosses montées en charge tout en consommant 10 fois moins de ressources.

Privilégier les sites statiques

Si une page ne doit être modifiée que 2 fois par an, préférez des pages statiques, construites en dehors d'un CMS. Cela permettra d'économiser des cycles CPU, de la bande passante et réduira la consommation électrique ; tout en améliorant considérablement le temps de chargement du site puisque la page sera déjà présente dans le cache du navigateur de l'internaute ou dans les différents caches et proxies du réseau (fournisseur d'accès internet, CDN, proxies du site, etc.). Un fichier statique est directement lu et renvoyé à l'internaute par le serveur HTTP ou le serveur de cache sans solliciter le serveur d'applications et la base de données sous-jacente. A l'opposé, utiliser un système de gestion de contenu (CMS) requiert de charger les différentes couches logicielles pour servir le contenu demandé par l'internaute : serveur HTTP, système de gestion de contenu (Drupal par exemple), serveur d'application (PHP par exemple), système de stockage du contenu (base de données relationnelle), etc. De nombreux outils facilitent la génération de sites statiques. Par exemple : Jekyll, Octopress, Pelican Prose.io, etc. Pour Jekyll (http://jekyllrb.com/) par exemple, l'utilisateur installe ce programme écrit dans le langage Ruby sur son ordinateur. Après avoir choisi ou créé son thème, il contribue au contenu de son site en écrivant des fichiers textes respectant le format Markdown (http://daringfireball.net/projects/markdown/). Il peut aussi agrémenter le contenu de metadonnées écrites au format Yaml. Lorsque le contenu du site est écrit, l'utilisateur lance la commande `jekyll build` qui génère les fichiers HTML (statiques) du site. Il ne reste plus qu'à publier ces fichiers sur le serveur web (via FTP ou rsync par exemple) Fig.1.

Fig.1



Quelques conventions permettent de préciser la mise en forme attendue (Markdown). Le fichier HTML sert de template.

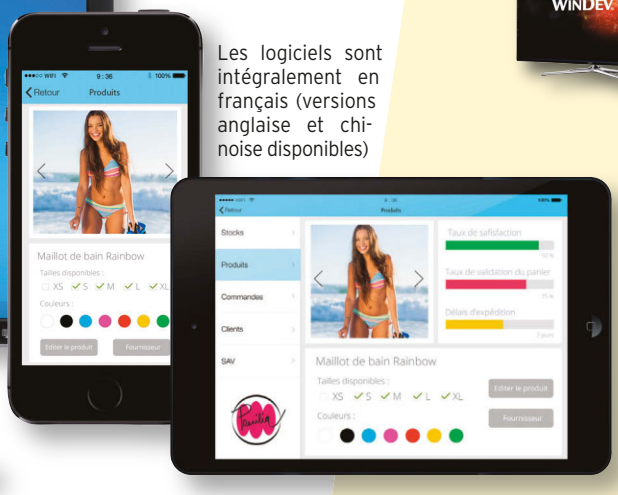
VOUS DIRIGEZ UNE SOCIÉTÉ, VOUS RECHERCHEZ L'EFFICACITÉ POUR VOS LOGICIELS : VOUS AUSSI, CHOISISSEZ WINDEV 19

Avec **WINDEV 19**, votre service informatique ou votre SSII développe plus vite vos logiciels et vos sites, de manière unique **pour toutes les plateformes**, en natif !
Avec **WINDEV 19**, le développement est effectué une seule fois pour tous les matériels: **PC, Mac, Tablette, Smartphone, Internet**,... Le déploiement s'effectue sans redevance.

Les applications que vous développez sont **natives** et **portables**

BÉNÉFICIEZ DE VOS APPLICATIONS SUR MOBILE !

Les logiciels sont intégralement en français (versions anglaise et chinoise disponibles)



applications natives

Les applications développées avec WINDEV s'intègrent parfaitement à votre S.I. existant, et à votre base de données actuelle: Oracle, SQL Server, MySQL, DB2 etc.

VU À LA TÉLÉ SUR M6

96%

47% autres WINDEV

UN TAUX DE SUCCÈS DES PROJETS SANS ÉQUIVALENT

WINDEV 19 permet de créer les applications dans tous les domaines, que ce soit des applications autonomes, ou des applications reliées à un existant, logiciel spécifique ou progiciel (ERP,...).

VOS DÉVELOPPEMENTS SONT MULTIPLAFORMES: Windows, Linux, Android, iOS (iPhone & iPad), Internet, Cloud... Vos projets réussissent à une **vitesse** et avec des **budgets** que vous n'osiez pas imaginer.

AVEC WINDEV VOUS RÉUSSISSEZ VOS PROJETS.

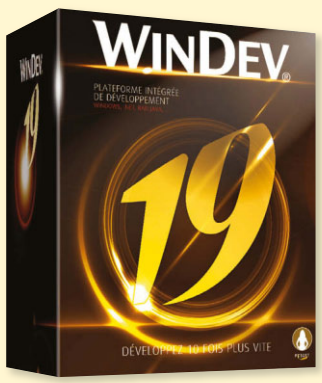
Vous gagnez en temps, en qualité, en fonctionnalités et en budgets.

N'hésitez pas à consulter notre site www.pcsoft.fr, à nous mailer (**info@pcsoft.fr**) ou à nous appeler :
Tél Paris: **01 48 01 48 88**, Tél Province: **04 67 032 032**.

Nous sommes à votre entière disposition !

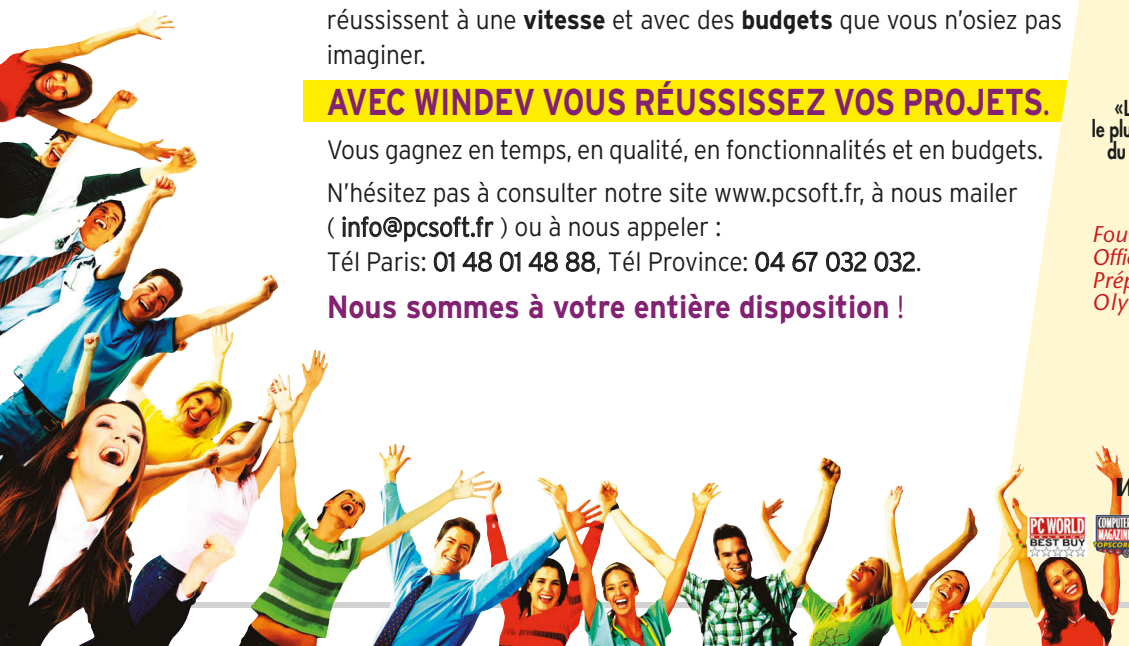
VOUS MÉRITEZ LE MEILLEUR

PC, Tablette, Smartphone, Internet
Développez avec WINDEV 19 !



Elu «Langage le plus productif du marché»

Fournisseur Officiel de la Préparation Olympique



www.pcsoft.fr

PC SOFT 25 ANNÉE Au service des développeurs

WINDEV AGL N°1 en FRANCE

Exemple Jekyll Fig.2.

2014-06-30-welcome-to-jekyll.markdown

```
---
layout: post
title:  "Welcome to Jekyll!"
date:   2014-06-30 14:40:54
categories: jekyll update
---

You'll find this post in your `_posts` directory - edit this post and re-build (or run with the `-w` switch) to see your changes!
To add new posts, simply add a file in the `_posts` directory that follows the convention: YYYY-MM-DD-name-of-post.ext.

Jekyll also offers powerful support for code snippets:

{% highlight ruby %}
def print_hi(name)
  puts "Hi, #{name}"
end
print_hi('Tom')
#=> prints 'Hi, Tom' to STDOUT.
{% endhighlight %}

Check out the [Jekyll docs][jekyll] for more info on how to get the most out of Jekyll. File all bugs/feature requests at [Jekyll's GitHub repo][jekyll-gh].

[jekyll-gh]: https://github.com/jekyll/jekyll
[jekyll]:    http://jekyllrb.com
```

post.html

```
---
layout: default
---
<div class="post">

  <header class="post-header">
    <h1>{{ page.title }}</h1>
    <p class="meta">{{ page.date | date: "%b %-d, %Y" }}{% if page.author %} • {{ page.author }}{% endif %}{% if page.meta %} • {{ page.meta }}{% endif %}</p>
  </header>

  <article class="post-content">
  {{ content }}
  </article>

</div>
```

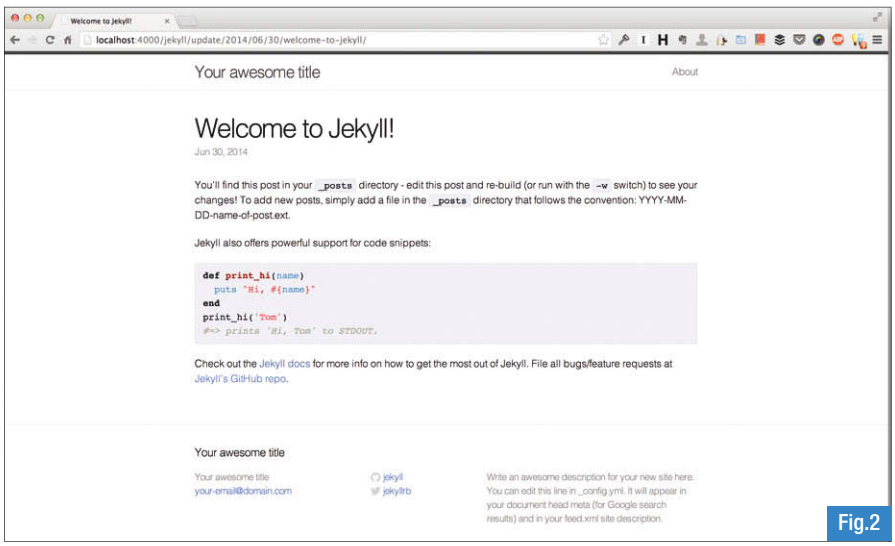


Fig.2

Jekyll transforme les fichiers textes (.md et .html) en une page HTML statique lisible par un navigateur web.

Choisir des forks orientés performance

Une fois le socle technologique retenu, il est possible de s'appuyer sur des versions optimisées. En effet, les logiciels open source sont souvent « forkés » pour des raisons de performance. En général, un gain de performance implique un gain de consommation de ressources et permet donc de réduire les impacts environnementaux associés au site. Si un fork optimisé existe et offre un périmètre équivalent, utilisez-le.
Voici une liste des principaux forks orientés performance :

- Drupal -> Pressflow
- MySQL -> Percona Server
- InnoDB -> TokuDB
- PHP + FPM -> HipHop de Facebook (HHVM)
- ...

Epurer l'infrastructure applicative

Il n'est malheureusement pas toujours possible d'utiliser un générateur de site. Il faut alors recourir à un outil plus évolué qui génère dynamiquement chacune des pages. Pour optimiser la quantité de ressources nécessaires au fonctionnement (cycles CPU, quantité de mémoire vive, nombre de serveurs, bande passante, etc.), il est essentiel de choisir l'outil le plus économe en fonction de ses besoins et contraintes métier. En effet, plus la solution retenue est "packagée" / de haut niveau, et plus elle empile des couches logicielles qui consomment des ressources. Au point qu'il vous faudra peut-être multiplier par deux, trois, ou plus, le nombre de serveurs.
Dans la plupart des cas, plus le développement est de bas niveau et meilleure sera l'efficience du code. On distingue trois niveaux possibles pour un site web dynamique :

- Développement sur mesure (PHP, JEE, .Net, etc.)
- Développement sur mesure au dessus d'un framework (Symfony, etc.)
- CMS (Drupal, Joomla, Wordpress, etc.)

Créer un site « responsive » oui, mais server-side

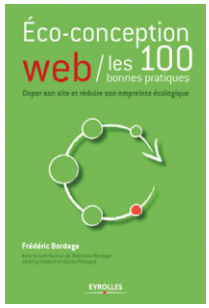
La plupart des nouveaux projets web reposent sur une architecture de type Responsive Web Design (RWD) pour s'adapter à la ribambelle de terminaux connectés, de l'écran 24 pouces d'un ordinateur de bureau en passant par un smartphone connecté en 3G à une tablette connectée en Wi-Fi. Face à cette diversité, seule l'adaptation dynamique aux propriétés physiques des terminaux garantit une bonne expérience utilisateur. Cependant, si la sélection des ressources s'effectue sur le terminal, cette architecture est un anti-pattern en termes d'efficience : car il faut transporter plus de ressources que le terminal n'en utilisera. D'où l'in-

térêt de l'architecture Responsive Design + Server Side Components (RESS). Cette dernière reprend les principes Responsive mais sélectionne côté serveur les ressources qui seront envoyées au terminal. On s'assure ainsi de ne pas consommer inutilement de la bande passante, ni de trop solliciter le processeur et la mémoire du terminal pour des traitements inutiles. Il s'agit ni plus ni moins que de pousser à l'extrême la bonne pratique qui consiste à fournir du code spécifique à un navigateur en particulier.

Parmi les solutions clés en main, RESS.io automatise la mise en œuvre d'un certain nombre de bonnes pratiques responsive orientées efficience. Par exemple, RESS.io propose de servir des images redimensionnées pour les petits terminaux, de compresser automatiquement (gzip) les pages et les ressources CSS et JavaScript en sortie de serveur HTTP, de fusionner les feuilles de styles, etc. L'intérêt est d'alléger au maximum les données envoyées vers les différents terminaux. Par exemple, un smartphone n'a pas besoin de télécharger la feuille de styles des écrans dont la résolution est supérieure à 960px. Cette sélection des ressources côté serveur (server-side) allège considérablement les échanges sans pour autant augmenter trop fortement les ressources serveurs nécessaires. En effet, RESS facilite la mise en place d'une architecture de cache plus performante, notamment via le langage Edge Side Include (voir ci-dessous).

Maximiser la possibilité de mise en cache

La mise en cache est le moyen le plus sûr de réduire le nombre de serveurs nécessaires à l'exécution d'un site web dynamique tout en réduisant le temps d'affichage pour l'internaute. Il faut donc concevoir les pages web et l'architecture sous-jacente en fonction de la capacité à les cacher avec des outils tels que Varnish et Squid. Evidemment, lorsque le site sert des pages personnalisées à un utilisateur identifié – Pole-Emploi.fr, banque en ligne, intranet, etc. – le travail de mise en cache est plus difficile. On peut cependant s'appuyer sur le langage ESI (Edge Side Include) pour construire des pages dynamiques par assemblage de fragments HTML au niveau du proxy et non plus au niveau du serveur d'application ou du serveur web. Les proxies les plus courants tels que Varnish et Squid, ainsi que des CDN comme Akamaï, supportent nativement ce langage. Les balises ESI sont insérées dans le code HTML lors de la conception. Elles sont ensuite interprétées par le proxy (à la manière des directives de pré processeurs) et déclenchent des traitements de construction (assemblage) de la page. Cette répartition des traitements et leur spécialisation garantissent un usage optimum des serveurs physiques.



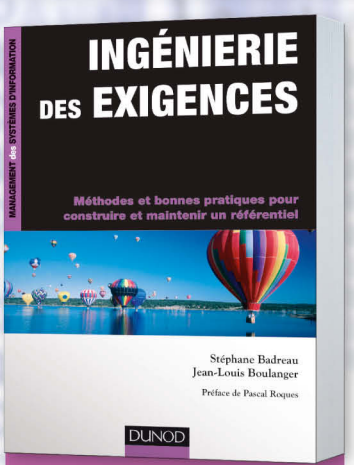
Eco-conception web les 100 bonnes pratiques
Frédéric Bornage
Editions Eyrolles
12€

Exemple de code ESI au sein d'une page HTML

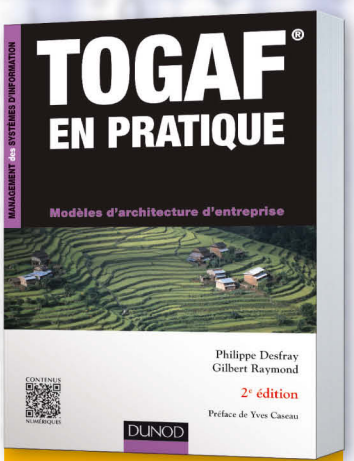
```
<esi:include
    src="http://example.com/1.html"
    alt="http://bak.example.com/2.html"
    onerror="continue" />
```

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

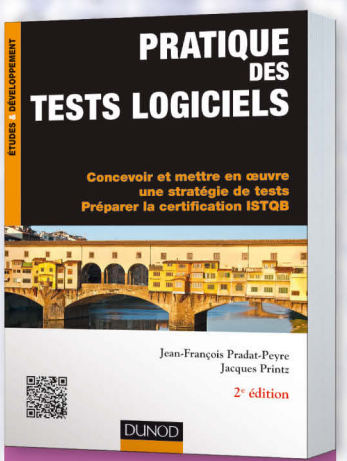
DÉVELOPPEZ VOS COMPÉTENCES



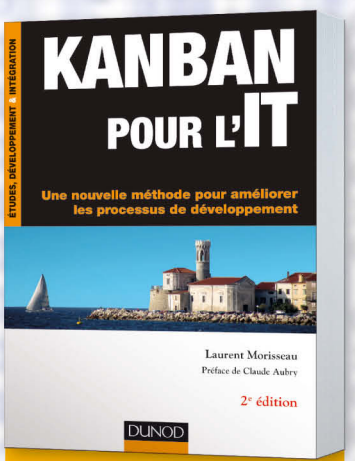
S. BADREAU, J.-L. BOULANGER
9782100706402 • 300 pages • **39,00 €**
Les connaissances de base pour développer des systèmes complexes à forte composante logicielle



P. DESFRAY, G. RAYMOND
9782100712823 • 288 pages • **35,00 €**
Très orienté solutions, ce livre apporte un point de vue de praticiens sur la modélisation d'architectures d'entreprise avec TOGAF®



J.-F. PRADAT-PEYRE, J. PRINTZ
9782100706082 • 240 pages • **29,50 €**
Les bonnes pratiques pour concevoir et mener à bien une stratégie de tests



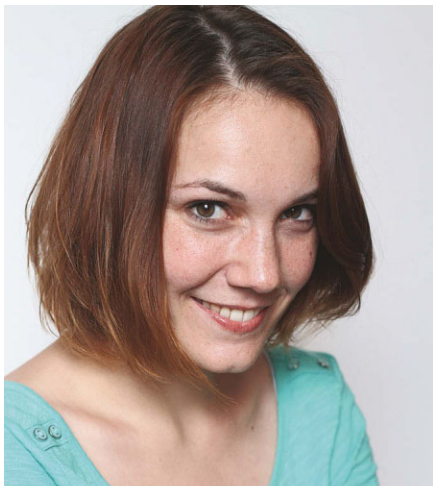
L. MORISSEAU
9782100710386 • 304 pages • **29,90 €**
Améliorer les processus de développement avec la méthode Kanban

www.dunod.com

Les actus du savoir

DUNOD
ÉDITEUR DE SAVOIRS

Réalisation : WIP



Ludwine : développeuse et experte en donnée

Ludwine, 30 ans, est Data Engineer chez Cityzen Data, après un Master en Mathématiques Recherche à l'Université de Franche-Comté. Aujourd'hui, elle imagine des méthodes et outils d'analyse de données issues de capteurs et les industrialise pour les proposer aux clients de Cityzen Data, incluant du Machine Learning et du traitement en temps réel.

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

Après mon Master, sans idées bien arrêtées de ce que je voulais faire, j'ai cherché un travail et je suis tombée sur une offre pour être développeur. Je n'avais pas de connaissances en programmation et je n'étais pas non plus particulièrement à l'aise avec les ordinateurs en général. Je me suis rendu compte que l'informatique est un outil qui permet de résoudre des problèmes, exige une certaine rigueur mais laisse auusi place à de la créativité. Au final j'ai retrouvé dans l'informatique ce que je pouvais aimer dans les mathématiques. Je suis dit que ce secteur pouvait me correspondre et répondre à mes attentes, alors je suis montée à Paris pour mon premier job dans ce milieu en 2010. Aujourd'hui avec plus de recul je suis très satisfaite de mon choix. J'apprécie particulièrement le travail en équipe avec la volonté de créer ensemble des projets/produits. Je suis contente d'être challengée intellectuellement et techniquement. J'apprends beaucoup de nouvelles choses chaque semaine. C'est l'une des choses d'ailleurs qui me plaît dans l'informatique en général : c'est un domaine assez récent, en constante évolution, qui laisse beaucoup de possibilités et de place à la créativité et à l'innovation. Si j'ai un message à faire passer c'est celui-ci : les réorientations, c'est tout à fait possible, quel que soit le background de départ. Je pense que la motivation et la curiosité sont les principales clés pour réussir.

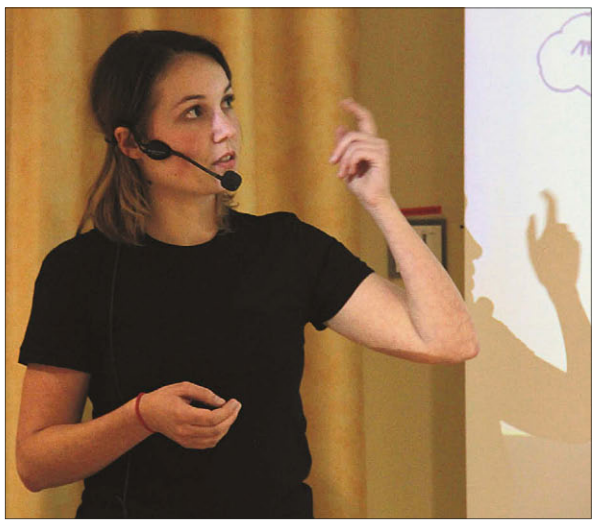
Pour toi, qu'est-ce qui fait que l'on aime toujours et encore le développement, la technique ?

Je pense que l'on peut voir le développement comme un langage de création : la programmation permet de créer des jeux, des applications mobiles, web, des sites....destinés à tous et utilisables par tous afin de rendre la vie plus pratique, ou encore de s'amuser, jouer, apprendre...Comme je le disais plus haut, l'informatique est un monde en mouvance permanente. Il reste encore beaucoup à faire, découvrir et à imaginer. Je pense que cette place à la créativité et au challenge peut rendre ce domaine attractif et passionnant. La technique évoque pour mois le défi intellectuel, l'envie de maîtriser un sujet, d'en comprendre les tenants et aboutissants.

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

Oui je suis "geek", autrement dit passionnée de manière générale. La veille techno prend une partie de mon emploi de temps sous des formes différentes : lecture d'articles de blog, news, évènements techniques, hackathon, conférences, implication dans des associations. Je vais très régulièrement aux divers évènements organisés par les communautés informatiques. C'est l'occasion pour moi d'apprendre de nouvelles technos/langages/outils. Et ces soirées sont toujours très enrichissantes autant professionnellement qu'humainement. D'autre part je suis leadeuse de l'association Duchess France et membre active du Google Developer Group (GDG), une association qui met en avant les technologies Google. Partant du constat que peu de femmes, en France du moins, sont présentes dans les milieux techniques en informatique, nous avons envie avec Duchess France de créer une dynamique autour du métier de développeur et de montrer que des femmes sont bien présentes et reconnues pour leurs expertises : nous voulons mettre en avant des rôles modèles. Nous encourageons aussi les femmes à participer à des conférences en tant que participantes mais aussi en tant que « speakeuses », puis à coopérer sur divers concours, projets open source et à s'investir d'avantage dans la communauté informatique. Avec ces diverses actions, nous espérons que des femmes, filles, jeunes filles se diront "Pourquoi pas moi ?" et investiront d'avantage les métiers du développement.

Concernant les "gadget", fan des figurines et miniatures en tout genre, j'ai récemment été bluffée par les réalisations de figurines à partir d'imprimantes 3D. Je vois de belle perspectives d'évolution notamment dans ce qui concerne le stop-motion.



Au final, oui la communauté est très importante pour moi : elle m'apporte la possibilité de rencontrer des passionnés et passionnantes dans divers domaines de l'information, dont certains que je ne connaissais pas. De fait c'est toujours très enrichissant puisque cela ouvre potentiellement sur de nouvelles perspectives professionnelles et humaines.

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

Durant ces 4 années d'expérience(s), j'ai

RÉUSSISSEZ votre CODE avec PROGRAMMEZ! le magazine du développeur

ABONNEZ-VOUS !

Offre spéciale rentrée 2014 (1)

Programmez! et les éditions ENI vous plongent dans l'univers du JavaScript

pour 1€ de plus

(valeur du livre : 22,43 € livre numérique, email indispensable)

(1) Valable uniquement pour la France métropolitaine. Quantité limitée. L'abonnement doit être envoyé avant le 31 octobre

© 09-26-05 © caracterdesign

Offre réservée à la France Métropolitaine

Spécial étudiant 39€
1 an 11 numéros

1 an 11 numéros 49€ seulement (*)

2 ans 22 numéros 79€ seulement (*)

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 + livre numérique ENI "JavaScript" :*** 50 € au lieu de 65,45 € / prix au n° + 29,90 €) : 50 € / abonnement seul : 49 €

☐ ***Abonnement 2 ans au magazine + livre numérique ENI "JavaScript" :*** 80 € au lieu de 130,90 € / prix au n° + 29,90 €) : 80 € / abonnement seul : 79 €

☐ ***Abonnement spécial étudiant 1 an au magazine + livre numérique ENI "JavaScript" :*** 40 € au lieu de 65,45 € / prix au n° + 29,90 €) : 40 € / abonnement seul : 39 €
Photocopie de la carte d'étudiant à joindre

☐ 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 178

***Offre limitée**, valable jusqu'au 31 octobre 2014*

travaillé sur divers projets : en SSII en forfait, chez des clients, et en startup. Cela m'a permis d'être face à un éventail de situations avec des organisations internes et du management de projets différents. J'ai beaucoup appris de ces expériences, et elles m'ont aidé à comprendre ce qui me correspondait le mieux et ce que j'attendais de mon job.

La notion d'équipe, dans laquelle on construit ensemble et où chacun apporte à chacun, est très importante pour moi. Selon moi, le "chef" ne doit pas représenter qu'une figure d'autorité et se contenter d'imposer ses choix et points de vue. Dans ma vison, le "chef" est là pour faire ressortir le meilleur de chacun de son équipe et doit s'adapter aux différentes personnalités qu'il a en face de lui pour optimiser à la fois le travail personnel de chaque employé et le travail du groupe. Quel que soit le statut (manager, stagiaire, leader tech, dev...), je pense que chacun a à apprendre des autres. Aujourd'hui, ce sont ces choses-là que je recherche et attends de l'équipe dans laquelle je travaille. Les développeurs ne se résument pas à de simples exécutants ou "pisseurs de code" (même si dans certains cas c'est nécessaire et suffisant), mais comme plus créateurs, innovateurs. Du moins c'est comme cela que j'envisage mon job. Beaucoup d'idées sont encore à trouver pour innover, améliorer le quotidien des utilisateurs ou tout simplement créer les technologies et outils du futur. Et je pense que les développeurs ont ce pouvoir de faire avancer, évoluer, changer les choses.

Je n'ai que 4 ans d'expériences, donc peu de recul. J'ai pu noter quelques engouements concernant quelques buzz word comme Big Data et Machine Learning. Ces domaines sont bien plus présents aujourd'hui et de nouvelles formations ouvrent dans les écoles.

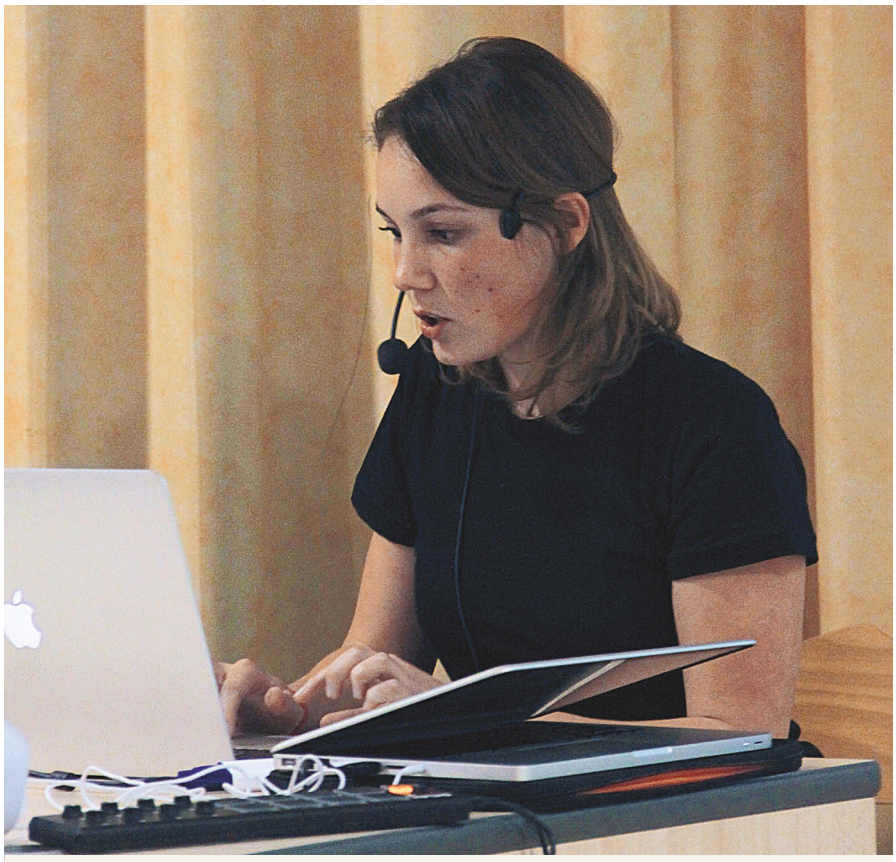
J'ai l'impression que l'esprit startup est plus encouragé, notamment grâce aux incubateurs et diverses initiatives axées entrepreneurs.

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

Ancienne nageuse en compétition, je continue à nager en alternant avec la course à pied. Je suis fan de pâte à modeler depuis toujours et suis régulièrement quelques ateliers de poterie et dessine pour mon blog. J'aime beaucoup voyager, et j'en profite pour m'exercer à la photographie.

Peux-tu nous présenter ton quotidien en quelques mots ?

J'ai rejoins la startup Citizen Data le 1er septembre dernier. Mon quotidien va donc un peu changer. Je vais devoir me familiariser avec mon nouvel environnement et ma nouvelle équipe. Puis monter en compétence sur leurs technos et commencer à implémenter de nouvelles features propres à mon poste chez eux.



Mon bureau

Je travaille avec un mac book pro personnel. Le portable est pour moi idéal puisqu'il me permet d'être flexible quant au lieu où je peux travailler. En effet, j'aime bien changer d'endroit, de bureau ou même de pièce, ça me permet de m'aérer l'esprit, de prendre un peu plus de recul (très bénéfique lorsque je travaille sur des tâches prenantes) sur mon travail en cours. On repart avec un regard différent et plus neuf. En général je travaille sans musique. J'ai la chance d'avoir une bonne concentration et donc n'utilise pas mes écouteurs pour m'isoler du bruit. J'ai tendance à écouter de la musique lorsque je suis sur une tâche assez pénible à faire, histoire de décompresser. Sinon j'aime bien lorsque j'en ai la possibilité, aller travailler dans des espaces de co-working et profiter d'une toute autre ambiance avec des personnes issues d'univers et pays différents.

Comment vois-tu ton job évoluer ?

Dans mon job, la dimension relationnelle, le partage, la communication, l'ouverture d'esprit, l'envie de créer ensemble sont les points clé qui me semblent indispensables pour y être heureuse. Pour le moment j'ai envie de me consacrer à du développement. D'ici quelques années, j'aurai envie tout en restant dans de l'expertise technique, d'avoir la possibilité de la partager via des conférences par exemple et avoir une partie relationnelle plus importante.

Des conseils aux étudiants et dévs qui nous lisent ?

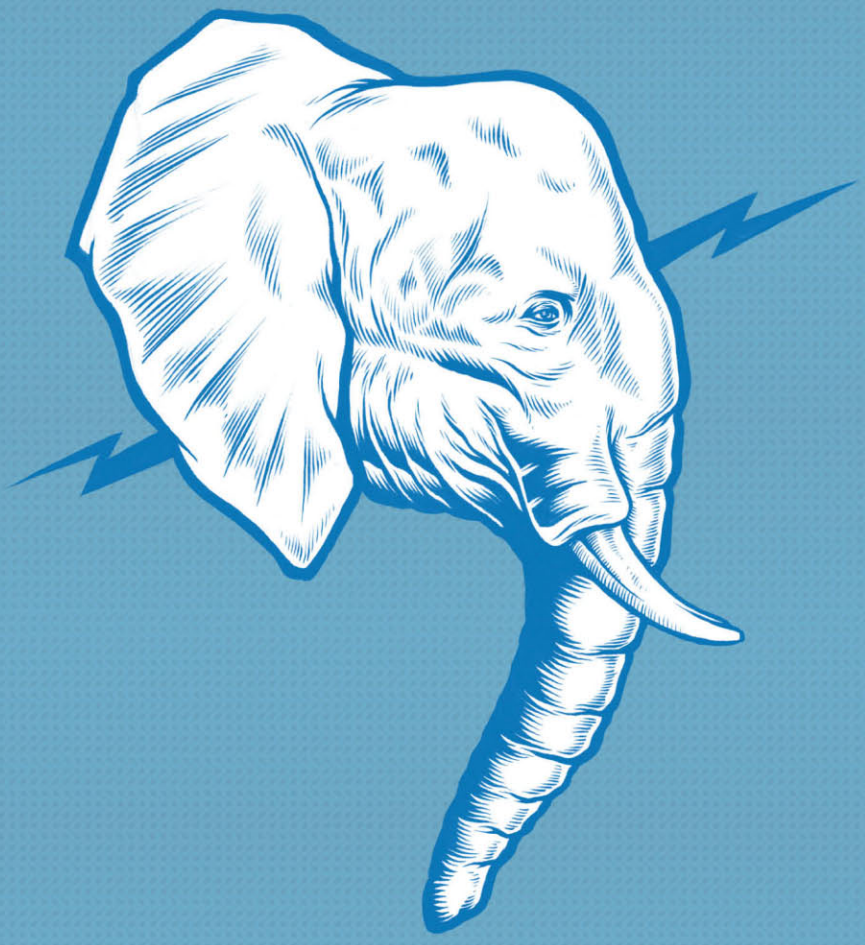
En étant étudiant, je pense que c'est un plus de commencer à faire de la veille technique en allant à des événements de la communauté par exemple. Partout en France, et pas seulement à Paris, il existe beaucoup de groupes meetup ou google group sur tous les langages et technos possibles. C'est un très bon moyen de rencontrer des développeurs plus expérimentés, avec des parcours différents, pas forcément linéaires. Cela permet aussi d'être conscient des différentes possibilités qui s'offrent à vous en terme d'opportunités professionnelles, technologies ou encore domaines informatiques.

Rester ouvert, se détacher de ses préjugés, tenter, prendre des risques, je pense que c'est la meilleure attitude à avoir autant professionnellement que personnellement.

Un conseil ? Ne pas négliger l'anglais et les maths. L'anglais pour les opportunités professionnelles à l'étranger notamment. Les maths, pour le raisonnement mathématique avant tout, mais aussi parce qu'aujourd'hui l'intersection entre les domaines des maths et l'informatique est grande et prometteuse.

FORUM PHP
PARIS 2014

l'événement majeur
de la communauté PHP francophone
depuis plus de 10 ans



23 et 24 octobre, Beffroi de Montrouge

Deux jours de conférences présentées par les plus grands experts mondiaux, des ateliers pratiques gratuits, des retours d'expérience de grandes entreprises, des cliniques-conseil tenues par des spécialistes, un apéro communautaire, pour amener la communauté PHP à son top !

Informations et réservations : www.forumphp.org

afup

Coding goûters : seulement pour les geeks?

Pour un coding goûter réussi, il faut :
- Des ordinateurs, avec des logiciels d'apprentissage préinstallés,
- Des parents, informaticiens de préférence,
- Des enfants modèles qui se destinent à une brillante carrière de développeur,
- Des animateurs motivés et patients,
- Et un goûter aussi.

Les ordinateurs, c'est facile à trouver, il suffit d'aller au supermarché. De plus, des logiciels éducatifs de programmation sont accessibles gratuitement sur Internet. La première étape est donc validée.
Mais les participants, ça paraît plus complexe: le public semble restreint. Finalement quel profil ont-ils vraiment? Et pourquoi sont-ils présents dans ces ateliers de programmation?

QUI SONT LES PARTICIPANTS?

Quels parents?

Les parents sont-ils toujours des informaticiens? A vrai dire, s'ils ne sont pas forcément des développeurs, ils ont pour la plupart soit un métier en rapport avec l'informatique, soit un proche qui est de la partie.
Sinon, comment savoir que ces ateliers éducatifs existent ? Et comment motiver ses enfants? «Samedi, tu n'as pas école, alors je t'ai inscrit à un cours de programmation informatique. Il faut penser à ton avenir plutôt qu'à aller jouer dehors». Pas sûr que ça fonctionne.
Alors, même si les ateliers de programmation sont accessibles à tous, et ont vocation à toucher toutes les catégories, le public reste encore un public d'initiés.
Il s'élargira progressivement plus les ateliers se multiplieront, et plus les parents avertis et convaincus de leur intérêt proposeront d'inscrire un camarade de l'école, une copine du club de sport, le fils de la nourrice... Ils ouvriront ainsi leur monde informatique à tout un chacun.
Pour l'anecdote, une grand-mère non-informaticienne qui amène ses petits-enfants pour leur proposer de nouvelles activités, c'est possible aussi!
Reprenons, nous avons donc des parents majoritairement informaticiens. Pour quelles raisons inscrivent-ils leurs enfants à ces sessions? Ils pourraient très bien leur apprendre à la maison, ils savent comment faire. Connaissez-vous beaucoup de parents qui savent lire? Un certain nombre. Et une deuxième question: connaissez-vous beaucoup de parents qui savent lire et qui apprennent à lire à leurs enfants? Beaucoup moins sans doute ...
Ces parents veulent expliquer leur métier à leur progéniture : "Je travaille sur un projet informatique: je programme des logiciels sur des ordinateurs."
"C'est quoi un logiciel? Ça veut dire quoi programmer? Papa, tu joues à l'ordinateur toute la journée, alors que je n'ai droit qu'à une heure d'écran par jour." Quelle injustice!
En inscrivant leur enfant, ils lui permettent de comprendre ces mots un peu trop professionnels, et de réellement pratiquer le développement d'un logiciel, même simple.
Les parents souvent passionnés sont alors fiers que leurs enfants aient pu approcher de façon concrète leur métier, abstrait pour beaucoup d'entre nous.
Mais cela ne signifie pas que ce métier qu'ils ont choisi pour eux-mêmes sera celui de leurs enfants. Qui dit être parent, dit éduquer et donner aux enfants le maximum de chances pour leur avenir.
Ces ateliers ont un but éducatif, et permettent aux enfants d'accéder à un nouvel apprentissage, qu'ils pourront approfondir par la suite s'ils le veulent.
Les parents ne souhaitent pas imposer un futur métier: "Tu seras informaticien mon fils, comme ton père et ta mère avant toi", mais leur proposer un choix supplémentaire.
L'école ne donne pas encore des cours de programmation, malgré l'omniprésence de l'informatique.
Pourtant les enfants dès 8 ans sont capables de programmer, avec des logiciels ludiques et adaptés à leur âge. Les parents ont confiance dans les capacités de l'enfant, et veulent leur donner une clé supplémentaire pour comprendre le monde de technologies dans lequel ils évoluent.

QUI ANIME ?

Des parents? Oui souvent, mais pas nécessairement.
Des professeurs confirmés? Et bien non justement; certains le sont, mais la plupart sont des professionnels de l'informatique, formateurs ou pas.
Qu'est-ce qui les regroupe? Essentiellement l'envie de transmettre un savoir, de partager leurs connaissances avec des enfants.
Tous sont conscients de l'absence de la programmation dans le cursus scolaire primaire, même si l'informatique fait partie intégrale de notre quotidien, et ce dès le plus jeune âge.
Peut-être veulent-ils casser les préjugés qui entourent leur métier.
L'informaticien ne serait pas ouvert aux autres, dans sa bulle, il préfèrerait rester dans son monde virtuel. Difficile de promouvoir une activité réputée asociale à des parents. Mais en pratique, avec ces ateliers, les enfants avancent ensemble:
- En binôme: ils apprennent à travailler à deux,
- La démonstration finale leur permet de présenter leur propre travail devant un auditoire, de voir ce que les autres ont fait et d'en tirer parti.

Où est le côté asocial?
L'informatique serait trop compliquée à comprendre, et l'ordinateur appartiendrait à une espèce de bête féroce, avec des bugs qui vous polluent la vie.
Et pourtant, en une simple après-midi, les enfants progressent et comprennent les concepts de la programmation avec plaisir.
Sans appréhension face à cette machine, ils lui donnent les instructions adéquates pour corriger ces fameux bugs.
L'informatique n'a pas forcément bonne presse, surtout pour des parents soucieux du bien-être de leurs enfants. En société, une parole comme "ma fille a appris à faire un programme informatique" peut vite couper court à la conversation: "Tu ne préfères pas l'emmener au parc plutôt que la coller devant un ordinateur ! "
Mais dans nos métiers actuels, quel que soit le domaine, dans nos loisirs, ne passons nous pas des heures et des heures à taper sur un clavier ? Ces ateliers proposent d'aller plus loin que la simple consommation de l'ordinateur. Il s'agit d'une démarche intellectuelle et éducative pour

permettre aux enfants de comprendre la technique, tout en s'amusant.

DES ENFANTS STÉRÉOTYPÉS ?

Des geeks, des geeks, et encore des geeks ... Commande de Lego Mindstorms pour Noël, première compilation à 4 ans, principal loisir, les jeux vidéos ... Ou pas ...
Effectivement, certains sont déjà embrigadés dans les nouvelles technologies, et détiennent déjà les clés de la programmation. Mais d'autres ne connaissent l'informatique que pour effectuer des recherches sur Internet et écrire quelques mails.
Peu importe, chacun est présent à l'atelier, et prend plaisir à programmer, à son niveau.
Les enfants viennent pour apprendre et surtout comprendre. La force de la programmation tient à cela: avec un même logiciel de développement, les façons de programmer sont infinies, et les réalisations informatiques le sont aussi. Les enfants de niveaux disparates ne produiront bien-entendu pas les mêmes programmes, mais tous progresseront.
N'allez pas croire que les enfants veulent participer parce qu'ils sont fans d'informatique et veulent en faire leur métier. Ils sont juste curieux, et toujours prêts à tenter une nouvelle expérience intellectuelle ludique. S'ils savent en quoi programmer consiste, ils seront plus à même de choisir: oui je veux en faire mon métier, j'aime programmer, ou non, je suis content de comprendre la machine et son raisonnement, mais ce que je veux vraiment faire quand je serai grand, c'est pompier, ou maîtresse. Les préjugés ont la vie dure.

Scolaires ou créatifs ?

Voici un autre stéréotype: l'informaticien manque de créativité. Ces ateliers ne doivent donc pas être proposés aux enfants qui ont un esprit un peu artiste, ils n'accrocheront pas.
Pourtant, programmer ça peut être:
- Suivre des exercices: trouver la solution d'un problème donné,
- Ou tout créer: inventer un scénario et le développer en suivant son imagination.

Il suffit d'un peu de concentration pour apprendre les bases. Ensuite, libre à l'enfant de développer selon son caractère et son envie du moment. Il peut tout à fait se laisser guider par sa créativité, plutôt que de suivre pas à pas une procédure. C'est l'un des avantages de réaliser cet apprentissage hors temps scolaire, sans exercice à respecter absolument pour avoir une note finale. L'inventivité de l'enfant peut s'exprimer librement.

Que des garçons?

Non, non, non. C'est la parité ici!
Et félicitations aux parents pour ça, ce sont eux qui permettent cette mixité: les frères et les sœurs participent, les copines sont amenées.
Les plateaux projets regorgent de population masculine, la moindre développeuse est presque une extra-terrestre. Faire participer des filles aux ateliers de programmation peut être un moyen pour ouvrir les études en informatique aux femmes. Les enfants n'ont pas encore cette notion de matière réservée aux garçons et s'en sortent brillamment qu'ils soient programmeurs ou programmeuses.
En fin de compte, les participants ont-ils tous le même profil? Les adultes, parents ou animateurs, ont de nombreux points communs en informatique. Mais les enfants ?
Tous différents.
Alors n'hésitez pas à inscrire vos filles, garçons, neveux, nièces, cousins, cousines, voisins, voisines, etc ... Tous les enfants sont les bienvenus et tireront un bénéfice de cet apprentissage, qu'ils deviennent programmeurs à leur tour, ou pas !

Aude **Lemar-Verrier**
VISEO
aude.lemarverrier@viseo.com

Abonnez-vous !

PROGRAMMEZ! le magazine du développeur

Offre spéciale rentrée 2014 (1)
Programmez! et les éditions ENI vous plongent dans l'univers du JavaScript pour 1€ de plus
(valeur du livre : 22,43 € livre numérique, email indispensable)
(1) Valable uniquement pour la France métropolitaine.
Quantité limitée.L'abonnement doit être envoyé avant le 31 octobre

1 an 11 numéros **49€** *seulement (*)*

2 ans 22 numéros **79€** *seulement (*)*

Offre réservée à la France Métropolitaine
Spécial étudiant **39€** *1 an 11 numéros*

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 + livre numérique ENI "JavaScript" :*** 50 € au lieu de 65,45 € / prix au n° + 29,90 €) : 50 € / abonnement seul : 49 €
☐ ***Abonnement 2 ans au magazine + livre numérique ENI "JavaScript" :*** 80 € au lieu de 130,90 € / prix au n° + 29,90 €) : 80 € / abonnement seul : 79 €
☐ ***Abonnement spécial étudiant 1 an au magazine + livre numérique ENI "JavaScript" :*** 40 € au lieu de 65,45 € / prix au n° + 29,90 €) : 40 € / abonnement seul : 39 €
Photocopie de la carte d'étudiant à joindre

☐ 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 178
***Offre limitée**, valable jusqu'au 31 octobre 2014*

Tour d'horizon du développement Chrome Apps

1ere partie

En Septembre 2013, Google annonçait un nouveau type d'applications Chrome beaucoup plus intégrées qui agissent comme des applications natives sur l'OS. Ces premières applications ont été mises à disposition pour les versions de Chrome Desktop dans un premier temps, puis annoncées sur plateformes mobiles en janvier dernier. De quoi entrer dans l'ère des applications natives multi-device.

Une application Chrome, c'est quoi ?

Une application Chrome ("*Chrome App*") est avant tout une application développée avec les technologies du Web : HTML5, JavaScript et CSS. Ces langages vont constituer la base de l'application. On tire ensuite parti des possibilités nouvelles d'HTML5 et des API spécifiques de Chrome pour enrichir l'application.

Et c'est surtout sur ce dernier point que Chrome va apporter tout son intérêt; l'API est ouverte et propose un éventail de fonctionnalités permettant d'aller beaucoup plus loin que des applications Web puisqu'elle donne l'accès aux routines plus proches de l'OS et du matériel. L'application va donc pouvoir se comporter comme une application de bureau, tout en tirant parti des facilités de développement en HTML et JS.

Avant d'aller plus loin et éviter les confusions, nous allons d'abord faire un rappel sur la différence entre une application Web, application Chrome, et extension Chrome.

Application Web vs Application Chrome ?

Une application Web est un site "enrichi" en tirant parti de JavaScript et des fonctionnalités d'HTML5 pour offrir une expérience améliorée à l'utilisateur, mais s'exécutant dans le navigateur. L'application Web est "connectée" et standard (indépendante du navigateur).

Une Chrome App est une application de bureau et offre une grande immersion à l'utilisateur en tirant parti de l'OS et des capacités matérielles. L'application Chrome est prévue pour fonctionner en mode déconnecté et s'appuie fortement sur les fonctionnalités du navigateur.

Application Chrome ou extension Chrome ?

Une extension est un complément de Chrome, elle n'a pas de fonction propre en dehors de Chrome. Elle permet d'ajouter des fonctionnalités supplémentaires à Chrome (par exemple accessibles via un nouveau bouton dans la barre d'adresse).

Les applications sont, quant à elles, indépendantes et autonomes. Une application s'appuie sur Chrome seulement pour son exécution, mais elle en est indépendante Fig.1.

Généralités

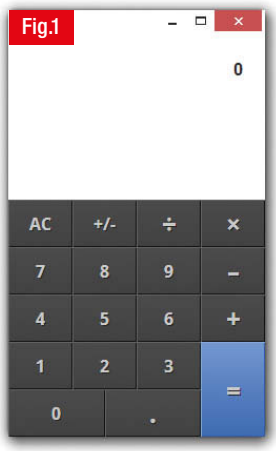
Avant d'entrer plus en détails, voici quelques généralités des Chrome Apps.

Mode Offline

Une *Chrome App* est déconnectée par défaut. Cela permet de mettre en avant l'approche *offline first* et concevoir l'application pour fonctionner non seulement en étant déconnecté, mais surtout pour répondre plus facilement aux cas pour lesquels le réseau est lent, pas fiable, ou si les serveurs sont indisponibles temporairement.

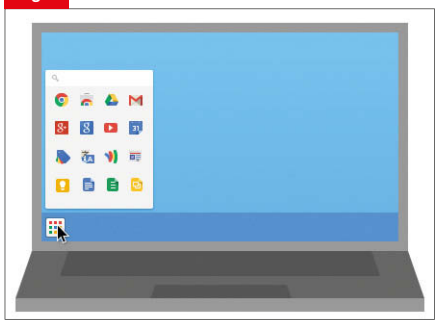
L'application Chrome apporte un plus ici car l'intégralité du package de l'application est téléchargée et disponible sans nécessité de connexion ultérieure pour son exécution.

Fig.1



Exemple d'application Chrome « Calculette ». Elle utilise le thème natif de l'OS sur lequel elle s'exécute.

Fig.2



Multi-device, multi-plateforme

Une Chrome App fonctionne sur la plupart des plateformes populaires sur lesquelles Chrome existe : Windows, OS X, GNU/Linux, mais aussi Android et iOS.

Immersif

L'application Chrome interagit avec l'OS hôte et avec le matériel (USB, Bluetooth, socket TCP, etc.). L'application est intégrée à l'OS (raccourcis, style de fenêtre adaptée, mode plein-écran)

Distribué

L'application Chrome est distribuée et mise à jour via **Chrome Web Store**. Cette approche permet également de simplifier la monétisation de l'application mais également des feedbacks rapides et facilités des développeurs.

Cloud

Une application Chrome est "cloud-enabled" par défaut. C'est-à-dire, que l'API propose des accès aux services Google permettant de faciliter le travail du développeur qui n'a rien de plus à coder pour les accès aux données, notamment les services Storage Sync, syncFileSystem, pushMessaging API.

Sécurisé et isolé

Les Chrome Apps implémentent Content Security Policy : en Javascript, pas d'utilisation de `eval()`, `new Function`, etc.

L'exécution des processus est isolée (principe de Sandboxing) : un plantage d'une application Chrome ne fera pas planter l'application voisine. Même principe de *Sandboxing* concernant le stockage des données applicatives qui est, lui aussi, isolé.

Autre point de sécurité : une application Chrome n'a accès qu'aux API Web standard et *Chrome Apps*, mais ne dispose pas des accès aux API de *Chrome Extensions*.

Le lanceur d'applications Chrome

L'intérêt de toutes ces fonctionnalités est évidemment l'intégration à Chrome OS. Toutes applications de Chrome OS sont des *Chrome Apps*, mais il est possible d'en tirer parti dès aujourd'hui directement au sein de votre OS habituel. Dès la première installation d'une application Chrome depuis le Web Store (https://chrome.google.com/webstore/category/collection/for_your_desktop), le lanceur d'applications Chrome va s'installer. Ce lanceur répertorie toutes les *Chrome Apps* et constitue un point central pour le lancement des applications, sans avoir à lancer le navigateur Chrome Fig.2.
Il permet également de créer des raccourcis directement sur le bureau ou la barre des tâches pour faciliter d'avantage l'accès aux applications.
Maintenant que l'on connaît les généralités sur les applications Chrome, voyons en détail comment elles sont constituées et comment en programmer une Fig.3.

Organisation des sources du projet

Une Chrome App est organisée comme un projet Web classique (index.html, assets) et comporte plusieurs fichiers complémentaires, le Manifest, le background page et les icones :

```
assets/
       images/
       scripts/
       css/
 manifest.json
 main.js
 index.html
 icon16.png
 icon48.png
 icon128.png
```

Le Manifest

Le Manifest (manifest.json) est un fichier important nécessaire à l'application Chrome. Il décrit son point d'entrée, ses ressources, ses autorisations d'accès. Il est décrit au format JSON.
Les informations minimales sont le nom et la version. Ainsi, une version minimale pourrait être définie comme :

```
{
  "manifest_version": 2,
  "name": "My App",
  "version": "versionString",
  "default_locale": "en",
  "description": "A plain text description",
  "icons": {...},
  "app": {
    "background": {
                "scripts": ["background.js"]
    },
  },
  "permissions": ["<all_urls>"]
}
```

On retrouve ces principales informations :

- Le descriptif de l'application : `nom`, `version`, description, `locale` (langue),
- Le fichier "point d'entrée" de l'application (`background`),
- Les icones associées (`icons`),
- Les permissions (`permissions`),
- Informations et paramétrages spécifiques aux API (storage, webviews, usb, bluetooth), au système de mise à jour, version minimale de Chrome (sur lesquels on ne va pas s'étendre dans cet article).

Pour plus de détails sur le contenu du manifest, n'hésitez pas à aller sur le site https://developer.chrome.com/apps/manifest. Revenons plus en détails sur les permissions, icones et point d'entrée dans les 3 prochains chapitres.

Les permissions

Pour utiliser la plupart des API, l'application doit déclarer explicitement ses intentions dans le manifeste. Les permissions sont ensuite affichées à l'utilisateur permettant de savoir quels sont les besoins de l'application. Les permissions aident à limiter les dommages si l'application est compromise par des malwares. Certaines permissions peuvent être facultatives. Les permissions peuvent être affichées à l'utilisateur avant l'installation (ou à tout moment sur la page des extensions chrome://extensions). Parmi les permissions les plus utiles, le pattern "`[scheme]:[host]/*`" spécifie les URL vers lesquelles l'application souhaite communiquer. En effet, pour éviter les attaques de types XSS, il est nécessaire d'indiquer explicitement toutes les URL auxquelles on souhaite accéder. Heureusement ici, il est possible de faire du Pattern Matching pour nous faciliter la tâche.
Quelques exemples :

- `http://*/*` : toutes les URL qui matchent le protocole http. Par exemple http://www.google.com/,
- `file:///foo*` : tous les accès aux fichiers locaux dont les fichiers commencent par foo*,
- `<all_urls>` : matche toutes les URL.

Parmi les autres permissions, on retrouve les accès aux alarmes (`alarms`), notifications (`notifications`), périphériques USB (`usb`), synthèse vocale (`tts`) stockage (`storage`), video (`videoCapture`), etc. La liste des permissions complète est disponible à cette adresse : https://developer.chrome.com/apps/declare_permissions. Dans tous les cas, la documentation de l'API de Chrome indiquera toujours quelles permissions accorder pour qu'une API soit utilisable.
Les permissions apparaissent dans le manifest :

```
"permissions": [
  "serial",
  "storage",
  "videoCapture"
],
```

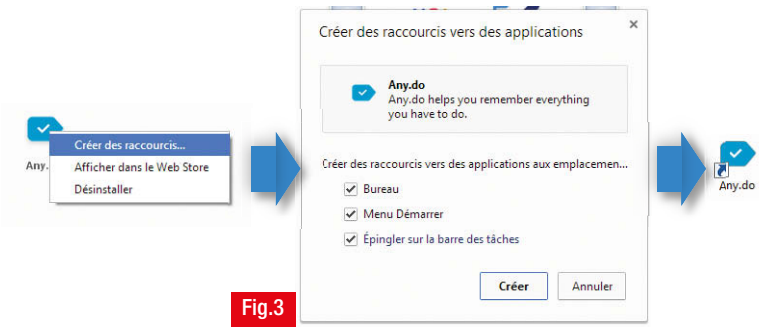


Fig.3

Le point d'entrée de l'application

L'Event page est référencée dans le *Manifest* par le mot-clé "`background`" et indique un fichier Javascript associé. C'est ce fichier JS qui va constituer le point d'entrée de l'application en agissant sur les

évènements systèmes liés à l'application (lancement, redémarrage). Nous préconisons de bien prendre en compte le nouveau modèle de chargement évènementiel **Event page** dont le détail est expliqué ici (https://developer.chrome.com/apps/event_pages). L'ancienne façon de faire, appelée, « Background page » (https://developer.chrome.com/apps/background_pages), bien qu'elle soit toujours opérationnelle, n'est pas préconisée car moins performante. Les évènements de pages ne sont lancés qu'en cas de besoin et respectent le cycle de vie applicatif. L'évènement `chrome.app.runtime.onLaunched` est appelé au lancement de l'application. Nous lui associons une fonction grâce à `addListener`. Dans la fonction associée, nous créons une nouvelle fenêtre, avec la page `"index.html"` en indiquant la taille et position de la fenêtre. Le cœur du traitement de l'application est ensuite fait depuis la page `"index.html"`, comme une application web standard.

```
chrome.app.runtime.onLaunched.addListener(function() {

  chrome.app.window.create('index.html', {
    id: "helloWorldID",
    bounds: {
      width: 500,
      height: 300,
      left: 0,
      top:0
    }
  });
});
```

Les icônes

Une ou plusieurs icônes représentant l'application sont à fournir. Au moins une : idéalement en format 128x128 pixels. C'est cette icône qui sera utilisée pour représenter l'application sur la page des applications, page de management, barre d'outils, etc. Il est préférable d'utiliser le format PNG car il offre une gestion de la transparence sans perte de qualité. La taille 48 est utilisée sur la page d'administration de l'application (celle qui apparaît sous `Chrome://extensions`).

```
"icons": { "16": "icon16.png",
           "48": "icon48.png",
          "128": "icon128.png" }
```

Attention toutefois, pour la phase de publication sur le Google Web Store, il sera également nécessaire de fournir une autre série d'images, les screenshots. Il est conseillé de mettre au moins la version 128. Chrome ajustera les autres tailles en se basant sur l'image en meilleure résolution s'il ne trouve pas les tailles plus réduites. Attention à bien respecter les tailles documentées et leurs tailles associées (128x128 pixels pour "128", 48x48 pixels pour "48") afin de ne pas avoir de surprise lorsque Chrome tentera de redimensionner ces images. Pour plus d'info : https://developer.chrome.com/apps/manifest/icons

Quelles API ?

Le nombre d'API disponibles pour les Chrome Apps est conséquent : Bluetooth, menu contextuels, système de fichiers, fenêtres sans cadre, plein-écran, géolocalisation, UserMedia (capture de son, vidéo), HID, identité (OAuth2), monétisation in-app, gallerie de média, Google Cloud Messaging, Impression, notifications enrichies, port série, Socket UDP/TCP, stockage, SyncFileSystem, Synthèse vocale, informations systèmes, USB, WebStore, WebView.

Une page d'exemples répertorie des cas types pour un grand nombre de ces API : https://github.com/GoogleChrome/chrome-app-samples
A contrario, certaines API Web ont été dépréciées, car elles sont considérées comme des mauvaises pratiques d'interactions, peu performantes ou tout simplement pas adaptées aux applications natives. Ainsi, on ne pourra pas utiliser les `window.alert`, `window.confirm`, `showModalDialog`, `document.cookie`, `document.close`, `document.open`, `document.write`, la navigation avec URL ou ancres, Flash, la soumission de formulaire, le *local storage*, WebSQL ou encore les requêtes `XmlHttpRequest` synchrones. On notera aussi que la sélection de texte par l'utilisateur n'est plus possible par défaut. Comment faire alors ? Faire sans ! La page https://developer.chrome.com/apps/app_deprecated liste cependant des contournements possibles pour ces cas précis.
Pour l'édition, comme il s'agit de ressources Web, il n'y a pas d'IDE dédié spécifiquement aux sources d'un projet Chrome app, il est préférable cependant d'utiliser un IDE Web avancé tel que Sublime Text, Brackets, ou encore WebStorm.

Exemple d'API : Sync Filesystem

Nous allons voir un exemple de codage en utilisant une des API phares : Sync FileSystem. Cette API est utile pour sauver et synchroniser les données vers Google Drive. Elle fournit un stockage spécifique à l'application synchronisable à tout moment, le développeur n'ayant pas à se soucier de savoir si l'application est connectée ou non. L'intérêt de ce type de stockage est de partager les données entre plusieurs device. L'appel de la fonction syncFileSystem.requestFileSystem retourne un filestystem synchronisable récupérable dans notre fonction de callback onInitFs.

```
chrome.syncFileSystem.requestFileSystem(onInitFs);
```

L'objet `fs` est un `DOMFileSystem` qui peut être opéré comme n'importe quel fichier distant ou local. Le filesystem ne supporte pas les opérations sur les répertoires : on peut obtenir la liste des fichiers depuis la racine d'un répertoire, mais pas en créer de nouveaux. Dans notre cas, la fonction de callback `getEntryCallback()` récupère notre fichier une fois accessible.

```
function onInitFs(fs) {
    fs.root.getFile('test.txt', {create:true}, getEntry
Callback, errorHandler);
}
```

On récupère l'objet représentant le fichier, puis on utilise un FileReader pour lire son contenu. La fonction `reader.onloadend` est appelée une fois le contenu disponible. Il est disponible via la variable

```
this.result.
Function getEntryCallback(fileEntry) {
    fileEntry.file(function(file) {
        var reader = new FileReader();
        reader.onloadend = function(e) {
          doSomething( this.result);
        };
        reader.readAsText(file);
}
```

Pour plus d'information sur l'API fileSystem HTML 5, se reporter à http://www.html5rocks.com/en/tutorials/file/filesystem/

Tester son application

Une fois que notre application est correctement organisée et que les principaux fichiers sont présents, elle va pouvoir être chargée pour son exécution. Pendant la phase de développement de l'application, on charge l'application dans Chrome en mode "non-empaquetée".
Pour charger une application, il faut ouvrir Chrome et se rendre sur la page des extensions (en tapant l'URL `chrome://extensions`) et cocher l'option "Mode développeur". Une fois cette option activée, il est maintenant possible de charger l'application en indiquant le répertoire racine de l'application (celui qui contient le manifest.json)
Dans ce mode, les sources sont chargées à partir de leur emplacement actuel qui permet un rechargement des sources, ce qui s'avère beaucoup plus productif en phase de développement. Pour le rechargement des sources, il suffit de se positionner sur la page des extensions et faire `Ctrl+R` (ou *cmd+R* sous Mac OS)
Pendant la phase de débogage, la console de développement de Chrome est accessible. Il s'agit bien d'une console propre à notre application et pas celle de Chrome. Pour afficher la console de développement, Clic-droit sur l'application puis "Inspecter l'élément".
Autre point à savoir : la console de développement Web n'est accessible que si l'application est chargée en mode "non-empaquetée".

Empaqueter l'application.

Bien que les *Chrome Apps* et *Chrome Extensions* soient différentes comme on l'a vu précédemment, elles partagent néanmoins le même système d'empaquetage et de distribution. L'application doit être empaquetée pour pouvoir être distribuée. Le packaging est un "zip" accompagné d'un certificat PEM qui constitue au final un package au format CRX. Le certificat PEM est obligatoire en cas de distribution sur le Google Web Store, mais facultatif en cas de distribution hors Store. S'il n'est pas présent il sera généré automatiquement lors du premier empaquetage. Pour empaqueter l'application, deux approches :

Approche manuelle

Depuis Chrome, aller sur la page des extensions, puis cliquer sur "Empaqueter l'extension". Sélectionner le répertoire source de la Chrome App et le certificat (si présent) puis "Empaqueter l'extension".
L'extension sera empaquetée dans le répertoire parent de l'application.

Approche scriptée

Il est possible de scripter l'empaquetage de plusieurs manières. La plus simple est d'utiliser Chrome en ligne de commande. Attention à bien indiquer les chemins complets (avec le c:\...) sinon, l'empaquetage ne fonctionne pas et il n'y a pas de message d'erreur pour nous l'indiquer.

```
chrome.exe --pack-extension=C:\myapp --pack-extension-key=C:\myapp.pem
```

L'autre approche est d'utiliser l'outil de scripting de son choix ! Le format CRX est un format ouvert dont les spécifications sont sur http://developer.chrome.com/extensions/crx. On peut donc très facilement créer son propre script d'empaquetage.

> Installer l'application.

Une fois le package CRX constitué, l'application peut être installée très facilement : il suffit de glisser-déposer le fichier CRX sur la page des extensions.
Chrome propose alors de l'installer. Elle est ensuite disponible automatiquement depuis le lanceur d'applications Chrome (ou ou depuis Chrome via l'URL chrome://apps).
Pour la distribution de l'application et sa mise à disposition sur "Desktop", il faudra, par la suite, la rendre disponible sur le Google Web Store. Nous verrons dans un prochain article comment aller plus loin en ciblant également les plateformes mobiles. La suite le mois prochain.

Florent **Dupont** (fdupont@sqli.com)
Expert Web & Mobilité chez SQLI
http://florent-dupont.blogspot.fr/

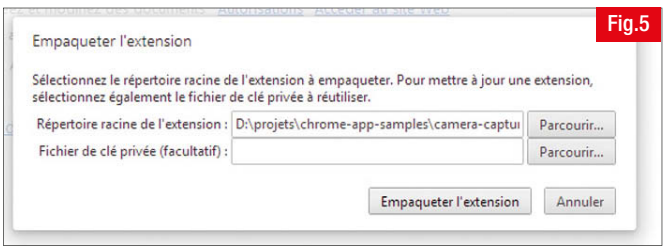


Fig.5

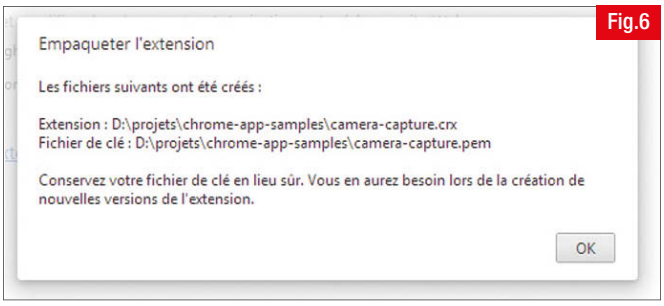


Fig.6

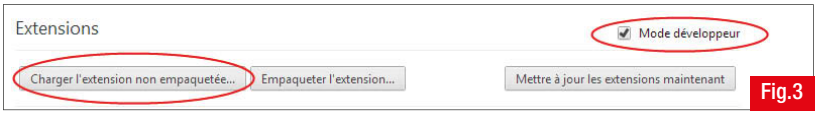


Fig.3

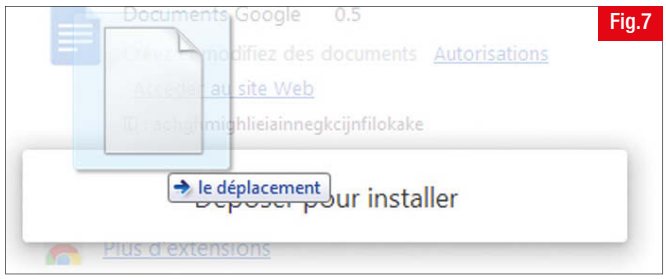


Fig.7

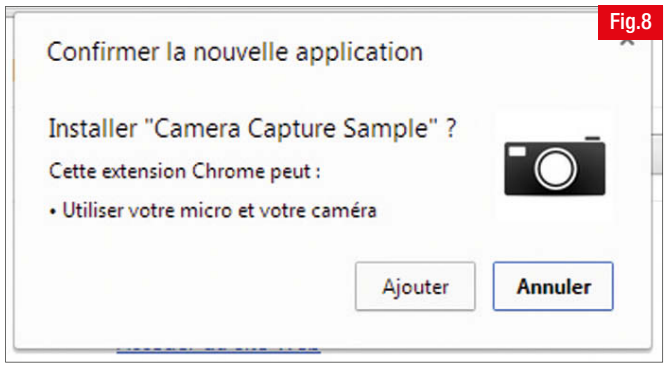


Fig.8

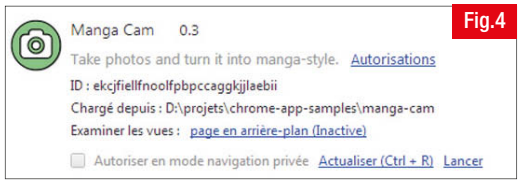


Fig.4

Code Legacy en C#, quels outils pour le tester ?

Dans cet article, nous vous proposons de revenir sur l'utilisation des Frameworks de Mocking : Pourquoi et quand doit-on les utiliser, quels sont les cas où nous devons utiliser des Frameworks avancés ? Ensuite nous présenterons certains scénarios où l'utilisation de JustMock s'avère utile.

Avant tout, nous allons rappeler quelques éléments de vocabulaire que nous utiliserons tout au long de cet article :

- On appelle System Under Test (SUT) la partie de code qui doit être testée. Cela peut être une ou plusieurs méthodes, une propriété ou encore une classe (dans ce cas, on parle même de Class Under Test).
- Pour isoler notre SUT de ses dépendances (base de données, classes, services, ...) on utilise des "Test Double". C'est un terme générique utilisé pour désigner tous les dispositifs que l'on met en place pour substituer des dépendances (appel à des méthodes, services, bases de données, etc.). Il en existe plusieurs types mais on utilise en général :
  - Les **fakes** : objets opérationnels uniquement pour les tests car ils utilisent certains raccourcis qui les rendent inutilisables en production. Les raccourcis sont en général les dépendances que l'on souhaite isoler,
  - Les **stubs** : objets que l'on utilise à la place des dépendances et que l'on configure pour renvoyer les données attendues par le test. Par exemple, si notre SUT fait appel à des données stockées en base de données, nous utiliserons un stub pour renvoyer des données à la place de celles-ci,
  - Les **mocks** : ce sont des stubs sur lesquelles nous allons pouvoir valider une interaction entre notre SUT et notre dépendance. Par exemple, si on utilise un mock pour une dépendance chargée de valider un résultat produit par une méthode (SUT), nous vérifierons que le SUT fait bien appel à notre mock.
- On utilise des frameworks de Mock pour fabriquer simplement ces "Test Double". Cela a eu pour conséquence d'utiliser le terme "mock" pour tous les types de "test double" sans aucune distinction.

Pourquoi et quand utiliser des Frameworks de Mocking

Ecrire des tests permet d'avoir une certaine maîtrise sur notre code : nous allons ainsi pouvoir détecter les régressions rapidement, refactorer notre application de manière sécurisée, corriger les erreurs de manière adaptée (sans faire de rustine dans le code de peur de tout casser). Bref, de la maintenir dans le temps sans que cela ne devienne un cauchemar pour les développeurs.

Mais cela a un prix : cela nous force à avoir un code « testable », ce qui signifie du code découplé (principe DIP), avec des méthodes et des classes avec peu de lignes de code et spécialisées (principe SRP), une complexité cyclomatique faible... En d'autres termes, du code de qualité. La principale difficulté que l'on rencontre est souvent liée au couplage. Lorsque l'on fait du TDD par exemple, nous construisons du code très peu couplé et cela ne pose pas de problème. Dans ces cas, l'utilisation de frameworks de mock "classiques" tels que Moq ou NSubstitute est parfaitement adaptée et a fait ses preuves. Le fait que ses frameworks soient gratuits et open source est un argument de plus en leur faveur.

Cependant, il existe certaines limites à ces frameworks : ils sont utilisables uniquement sur du code testable ce qui est loin d'être le cas de la majorité des applications en production actuellement. Les applications Legacy sont de plus en plus nombreuses et volumineuses, et aujourd'hui leur maintenance est devenue extrêmement problématique :

- Il n'y a pas de test pour sécuriser les évolutions,
- Pour introduire ces tests, il faut rendre le code testable,
- Pour rendre le code testable, nous devons le refactorer et donc potentiellement introduire des bugs car nous n'avons pas de tests pour sécuriser les évolutions.

C'est donc le serpent qui se mord la queue et nous arrivons à une situation où nous devons prendre des risques (introduire des bugs) si nous souhaitons améliorer la qualité de nos applications... Ce qui, dans beaucoup de cas, n'est pas acceptable.

Dans cette situation, l'utilisation de framework de type "profiler" peut nous aider à caser ce cycle de manière sécurisée : nous pourrons introduire des tests unitaires sans avoir à toucher le code, même si celui-ci n'est pas testable, et ainsi nous constituer un filet de sécurité qui nous permettra de faire évoluer le code.

Il en existe plusieurs sur le marché et nous avons déjà présenté le "Microsoft Fakes Framework" dans un article précédent. Aujourd'hui, nous vous proposons de découvrir JustMock à travers les anti-patterns les plus courants et nous vous montrerons comment l'utiliser pour isoler le SUT sans avoir à modifier le code.

JustMock est un Framework de "mocking" développé par Telerik, assez facile à prendre en main, qui vous permet d'écrire des tests unitaires assez simplement et rapidement. Ce Framework est conçu spécialement pour respecter le pattern AAA : Arrange (organise), Act (exécute), Assert (vérifie) afin de vous aider à rendre vos tests unitaires lisibles, structurés et propres.

Il existe deux versions du Framework :

- JustMock Lite : qui est une version de base et gratuite. Elle contient toutes les fonctionnalités que proposent les principaux Frameworks de mock gratuits du marché (tels que NSubstitute et Moq).
- JustMock Full Edition : celle-ci est payante et vous propose des fonctionnalités avancées vous permettant de résoudre quasiment toutes les problématiques liées à l'utilisation des mocks.

Exemples d'utilisation

Nous vous proposons ici les cas de couplage les plus répandus, cette liste n'est pas exhaustive concernant les possibilités de l'outil et nous vous invitons à vous rendre sur le site de Telerik si vous souhaitez avoir plus d'informations.

Classes du Framework .Net

Votre code fait souvent des appels à des objets/méthodes du Framework .Net ou à des classes de composants sur lesquelles vous n'avez pas la main. Par exemple, les appels aux classes DateTime, File, ConfigurationManager ne vous permettent pas d'isoler proprement votre SUT sans passer par des wrappers.

Dans l'exemple suivant, nous allons avoir une méthode qui, en fonction de la date, renvoie un format différent pour l'identifiant d'un utilisateur. Cette méthode peut être utilisée pour la désactivation ou le changement de comportement de notre application en fonction de la date (appelé aussi « Feature Toggle »).

```
public class User
{
  private readonly DateTime _refDate = new DateTime(2015, 1, 1);
```

```
  // SUT
  public string GetUserId()
  {

    if (DateTime.Today < _refDate)
      return "1";
    else
      return "00001";
  }
}

[TestClass]
public class UserTest
{
  [TestMethod]
  public void Should_Return_New_UserId()
  {
    var expected = "00001";

    // ARRANGE
    // Datetime.Today should return a date after the 01/01/2015
    var todayDate = new DateTime(2015,2,1);
    Mock.Arrange(() => DateTime.Today).Returns(todayDate);

    // ACT
    var user = new User();
    var actual = user.GetUserId();

    // ASSERT
    Assert.AreEqual(expected, actual);
  }
}
```

Appel statique

L'appel à un objet statique (et par extension l'utilisation du pattern singleton) est l'anti-pattern par excellence. Nous faisons appel à un objet qui peut avoir un état et qu'il n'est pas possible d'injecter.

```
  [TestClass]
public class UserTest
{
  public class User
  {
    // SUT
    public int GetUserId(string userName)
    {
      return UserStaticRepository.GetUserId(userName);
    }
  }

  //Static Dependency
  public static class UserStaticRepository
  {
    static UserStaticRepository()
    {
      throw new NotImplementedException();
    }
    public static int GetUserId(string userName)
    {
      throw new NotImplementedException();
    }
  }

  [TestMethod]
  public void GetUserIdTest()
  {
    var expected = 10;

    //ARRANGE
    Mock.SetupStatic(typeof (UserStaticRepository), StaticConstructor.Mocked);
    Mock.Arrange(() => UserStaticRepository.GetUserId(Arg.AnyString)).Returns(expected);

    //ACT
    var user = new User();
    var actual = user.GetUserId("Fathi");

    //ASSERT
    Assert.AreEqual(expected, actual);
  }
}
```

Future

Ici nous avons un cas un peu plus sournois, le couplage se matérialise par la création et l'utilisation d'un objet à l'intérieur même de notre méthode sans possibilité de l'atteindre depuis l'extérieur.

```
public class User
{
  // SUT
  public int GetUserId(string userName)
  {
    // dependency created and used inside the method
    var repository = new UserRepository();
    var id = repository.GetUserId(userName);

    return id;
  }
}

// Dependency
public class UserRepository
{
  public int GetUserId(string userName)
  {
    throw new NotImplementedException();
  }
}

[TestClass]
public class UserTest
{
  [TestMethod]
  public void GetUserIdTest()
  {
    var expected = 10;

    // ARRANGE
    //object used inside the SUT
```

```
    var futureMock = Mock.Create<UserRepository>();
     Mock.Arrange(() => futureMock.GetUserId(Arg.AnyString))
.IgnoreInstance().Returns(10);

    //ACT
    var user = new User();
    var actual = user.GetUserId("Fathi");
    // ASSERT
    Assert.AreEqual(expected, actual);
  }

 }
```

Constructeur

Le code du constructeur est toujours appelé... mais n'est pas souvent utile lorsque l'on souhaite tester une méthode ou une propriété, nous verrons ici comment le mocker proprement.

```
public class User
  {
    // Dependency
    public User()
    {
      //call db to fill User properties
      throw new NotImplementedException();
    }

    // SUT
    public int GetUserId()
    {
      return 1;
    }
  }
  [TestClass]
  public class UserTest
  {
    [TestMethod]
    public void GetUserIdTest()
    {
      var expected = 1;

      // ARRANGE
      var user = Mock.Create<User>(Constructor.Mocked);
      Mock.Arrange(() => user.GetUserId()).CallOriginal();


      //ACT
      var actual = user.GetUserId();

      // ASSERT
      Assert.AreEqual(expected, actual);
    }
  }
```

Appels à des méthodes/membres privés

Notre code peut avoir besoin d'appeler d'autres méthodes privées. Lorsque l'on fait du TDD, cela ne pose pas de problème car ces méthodes font partie du SUT. Cela permet, entre autres, de valider que notre encapsulation est pertinente et que notre code "privé" est bien appelé (si on n'arrive pas à l'atteindre depuis les méthodes publiques, c'est que ce code n'est jamais utilisé). Mais pour du code legacy, nous avons parfois besoin de le mocker.

```
public class User
 {
   // Dependency
   private int GetUserIdFromDb()
   {
     throw new NotImplementedException();
   }

   // SUT
   public int GetUserId()
   {
     var id = GetUserIdFromDb();
     return id;
   }
 }

 [TestClass]
 public class UserTest
 {
   [TestMethod]
   public void GetUserIdTest()
   {
     var expected = 10;

     var user = new User();

     // ARRANGE - When the non-public function GetUserIdFromDb()
 is called from the user instance,
     //  it should return expected integer.
     Mock.NonPublic.Arrange<int>(user, "GetUserIdFromDb").
Returns(expected);

     // ACT
     int actual = user.GetUserId();

     // ASSERT
     Assert.AreEqual(expected, actual);
   }
 }
```

Conclusion

Cette liste d'exemples n'est pas exhaustive, il existe en effet beaucoup d'autres cas de couplage que vous pouvez mocker avec JustMock. Vous trouverez plus de détails sur leur site web (http://www.telerik.com/products/mocking.aspx).

En conclusion, nous pouvons dire que c'est un outil très puissant qui va vous permettre d'isoler et de tester quasiment n'importe quel code. Mais nous devons garder à l'esprit que notre objectif n'est pas de tester notre code mais de nous assurer que celui-ci est de bonne qualité. C'est pourquoi, l'utilisation de ce genre d'outils doit être réservée à des cas où toutes les techniques classiques d'amélioration du code legacy ont échoué, en effet celles-ci imposent d'améliorer le code avant de pouvoir les tester.

Jason **De Oliveira** - *CTO | MVP C# chez Cellenza*
Son Blog : http://www.jasondeoliveira.com

Fathi **Bellahcene** - *Manager | MVP C# chez Cellenza*
Son Blog : http://blogs.codes-sources.com/fathi

Les lambdas en Java...script :

Programmation fonctionnelle en JS

1ère partie

Je vais profiter de l'actualité Java 8 et Javascript pour aborder un sujet que je pousse à chaque fois que l'occasion se présente : la programmation fonctionnelle - PF en abrégé.

On parle beaucoup de 'lambda' en ce moment chez les javaistes et principalement comme d'un moyen d'*alléger le code*... mais c'est l'arbre qui cache la forêt !

Qui parle de lambda parle de 'fonction' en tant que concept autonome, entité manipulable, réutilisable, composable. Ce n'est pas qu'une astuce, ou un sucre syntaxique pour réduire le nombre de lignes de nos programmes.

Disposer de la 'fonction' comme d'un élément structurant **de base** pour nos programmes (en Java, langage orienté objet, c'est la classe qui joue ce rôle) nous ouvre un champ de possibilités très vaste ; cela va beaucoup plus loin que le simple allègement du code, et on ne peut décemment pas l'ignorer lorsque l'on cherche l'élégance, la concision et l'expressivité dans les programmes.

Javascript, langage très souvent décrié et mal considéré, qui fait un come-back plutôt remarqué ces derniers temps avec le développement front-end, nous propose **justement de structurer nos programmes autour des fonctions**.

Alors, affutez vos fondamentaux en JS, oubliez tous vos réflexes de programmeur objet/impératif et chaussez vos escarpins de randonnée : **revisitons le JS à la sauce fonctionnelle**, en commençant **dans ce premier article par débroussailler quelques fondamentaux**.

Vous prendrez bien un peu de lambda calcul ?

Impossible donc d'y couper en ce moment avec la sortie de java 8 : *Lambda* par-ci, Lambda par-là...

```
Lambda => λ => 'fonction anonyme'.
```

Ce terme fait référence au lambda-calcul (-calcul), un système formel qui fonde les concepts de *fonction* et d'*application* de fonction à des *arguments*. Tout un univers gravitant autour du concept de **fonction comme centre des préoccupations** (approche plutôt étrange pour des OOP'istes, adeptes de la terre plate, pour qui une 'fonction' est avant tout une méthode, encapsulant un comportement d'instance/de classe).

Un univers très mathématiques, certes, mais qui énonce des règles et axiomes très puissants et intéressants pour qui souhaite en tirer parti dans le développement informatique.

Un argument, ça va ...

Un petit exemple de 'déclaration de fonction' en Lambda-calcul ?

```
λx.x
```

Comprendre : **je décris une fonction anonyme** qui **lie** un paramètre, **x**, à une **formule** qui une fois **appliquée** vaudra x (vous l'aurez peut-être reconnue, cette fonction : c'est la *fonction identité*).

Notez que le nom du paramètre importe peu (le lambda calcul parle d'alpha équivalence) :

```
λx.x ≡ λy.y ≡ λ@.@
```

Appliquez à cette expression une valeur revient à donner **une valeur à x** et **substituer la valeur dans la formule** :

[x.x] avec x = 2 nous donne : [2]

Traduisons rapidement ceci en langage de programmation afin de retenir ceux qui pensent déjà à s'enfuir :

```
// Lambda calcul
λx.x

// Javascript
function (x) {
  return x;
}

// Java 8
(x : int) -> x

// Scala
(x : Int) => x
```

Rien de bien compliqué ! 'Appliquer' des valeurs revient à 'exécuter' ces fonctions en leur passant une valeur en paramètre :

```
// Javascript : 'Application' de la fonction avec le paramètre '2'
(function (x) {
  return x;
}) (2)
// -> 2
```

Notez deux choses :

1 - Les notations Java 8 et Scala sont plus proches du concept d'application (j'associe un paramètre nommé à une expression, et je remplacerai la valeur dans l'expression par substitution au runtime) que la version javascript, qui souligne plus la notion d''appel' avec 'retour' (échange de messages).

2 - Le nombre d'arguments d'une fonction est aussi appellé **arité**. Dans notre exemple précédent : arité = 1.

Un autre petit exemple ?

```
// Lambda calcul : fonction 'élever au carré'
λx.x*x

// Javascript
(function (x) {
  return x*x;
}) (2)
// -> 2*2 -> 4
```

Deux arguments : ça va mieux !

Voyons maintenant la syntaxe d'une fonction à **deux** arguments (arité = 2) :

```
λx.λy.x+y
```

Je parie que vous vous attendiez à xy.x+y ? (si ce n'est pas le cas, vous avez sûrement triché).

Dans le lambda-calcul, les deux possibilités sont en fait correctes et équivalentes (je ne m'amuserai pas à ressortir la démonstration...). Elles aboutissent au même résultat une fois appliquées. Mais elles ne correspondent pas *tout à fait* à la même chose si on les transpose en code :

```
// Lambda calcul
λxy.x+y

// Javascript
function (x,y) {
  return x + y;
}

// Lambda calcul
λx.λy.x+y

// Javascript
function (x) {
  return function (y) {
    return x + y;
  }
}
```

(Ressortez vite "Javascript, the definitive guide" si vous sentez votre JS vaciller)
Essayez d'appliquer x et y à nos deux variantes de "lambda expression" : même résultat au final...
Sauf que cette application ne se fera pas de la même façon dans notre langage de programmation en fonction de la forme utilisée : soit x **et** y en même temps dans le premier cas, soit x **puis** y dans le deuxième.

```
// Javascript

(function (x,y) {
  return x + y;
}) (2,5)
// -> 7

(function (x) {
  return function (y) {
    return x + y;
  }
}) (2)(5)
// -> 7
```

Voilà déjà un apport intéressant de la théorie ! Le lambda calcul me dit que je peux considérer une fonction d'arité *n* comme autant de fonctions d'arité *1*, chacune étant **une étape d'une application / exécution**, plus complexe, partiellement résolue à chaque étape !
Quel intérêt me direz-vous ? J'étais moi aussi plutôt sceptique, au début. Mais nous verrons que dans certaines situations, cette propriété va nous être extrêmement utile pour arranger certaines tournures de style poussives en pirouettes fonctionnelles particulièrement élégantes.
Le terme officiel pour désigner ce passage (arité n) -> (n * arité 1) est **'Curryfication'** (ou 'Schönfinkelisation', mais c'est plus difficile à prononcer).

Mais où veut-il en venir ?

Ces exemples simples, en plus de glisser un peu de lambda calcul pour la culture, visent à amener la concentration du lecteur sur le concept même de **fonction**, de rappeler qu'une fonction peut **retourner une autre fonction**, et à présenter un premier outil fonctionnel de base, la curryfication, permettant de jouer sur la façon dont on va les invoquer (en une fois ou en plusieurs fois).

HERE BE DRAGONS

La programmation fonctionnelle ne se résume cependant pas à cela (la fonction comme brique de base) : c'est aussi un certain nombre de règles, conventions et axiomes, hérités des mathématiques et du lambda calcul où il n'y a pas de place à l'improvisation, aux 'hacks' et au débogage.
Certains langages (lisp, haskell...) sont génétiquement fonctionnels, d'autres non.
C'est le cas des langages modernes mainstream (Javascript y compris), qui laissent beaucoup de liberté aux développeurs : prenez des boucles *for*, des variables, des classes pour encapsuler un peu...
Cette liberté a malheureusement comme effet secondaire de rendre ceux-ci (les programmes, pas les développeurs) souvent durs à comprendre, maintenir, débugger.
Les plats de spaghettis ne sont pas rares ; et même avec la meilleure volonté du monde, un programme objet multi-couches multithreadé transbahutant des objets de droite et gauche finit toujours par atteindre un niveau de complexité trop élevé pour rester longtemps fréquentable.
Les développeurs le fuient, les utilisateurs craignent ses foudres...
Il y a donc deux catégories de langages : les langages purement fonctionnels, rigoristes, élégants, et les autres...

Les principes de base de la PF

En Javascript (deuxième catégorie), **il va donc falloir nous astreindre à appliquer autant que faire se peut les principes fonctionnels à force de discipline et de méthode**, et ne pas nous laisser (du moins, pas trop) tenter par les facilités laxistes qui nous sont offertes.
Quelle discipline, quels principes mettre en œuvre et respecter?
N'utiliser que des fonctions 'pures' : **sans effets de bords**

- **Ne pas utiliser de variables, uniquement des constantes**
- **Ecrire ses programmes** à partir de fonctions et, si possible, **uniquement de fonctions**
- Les fonctions peuvent **prendre en paramètres et renvoyer d'autres fonctions**.
- 'Fonction' est un type de donnée comme les autres (int, string...) et doit être utilisé comme tel.

Effets de bords

Le principe...
Par *effet de bords*, comprendre *dommages collatéraux* : on appelle une fonction en lui passant un paramètre, on récupère une valeur de retour... et le paramètre, ou quelque chose d'autre, a été modifié ailleurs, dans le programme. Vous, développeur, devez examiner le contenu de la fonction pour savoir exactement quels impacts son appel pourra avoir. *Encapsulation*, quelqu'un ?
Je passerais sous silence les fonctions de type *getSomeValue()* qui modifient des variables lorsqu'elles sont appelées - si possible sans rapport avec le *SomeValue*, pour bien faire. Pour illustrer, un peu de Javascript pas très fonctionnel (ni utile, d'ailleurs) :

```
var a = 1;
var b = 41;
function compute(){
  a += b;
}
compute();
alert('final value is ' + a);
```

Défauts :

- la fonction *compute* agit sur un élément extérieur => effet de bord observable, pas bien !
- la fonction *compute* utilise une information de son environnement pour fonctionner (valeur de b) => une fonction ne devrait se baser que sur les arguments qui lui sont passés (sauf dans un cas particulier dont nous parlerons plus tard) !
- la variable 'a' est modifiée après avoir été initialisée => pas bien (cf. utiliser uniquement des constantes) !

Version fonctionnelle, sans ces défauts :

```
function compute(a, b){
  return a + b;
}
alert('final value is ' + nice(1, 41));
```

L'exemple est trivial (et assez artificiel) et peut sembler anodin (juste un 'rewrite' de mauvaises pratiques). Mais imaginez cet embryon d'algorithme se diluer dans une fonction plus vaste... les variables s'empiler... il est tard, vous devez résoudre ce petit bug pour pouvoir livrer... Vous savez bien de quoi je parle !

Les fameuses fonctions de 1200 lignes bourrées de *if - else* et de variables, modifiées au fil de l'eau, sont faites de ces tout petits raccourcis qu'on se permet de prendre et qui au final coûtent très (trop) cher.

Donc : pas d'effets de bords, la fonction ne travaille et n'agit que sur et dans son corps et n'en sort jamais, et renvoie une valeur résultante de son application.

Une discipline, certes, mais dont les avantages à la longue n'ont pas de prix.

En pratique

Techniquement donc, **toute fonction se doit de 'retourner' quelque chose**. Autrement, elle ne sert à rien, puisque les effets de bords sont interdits.

Bien évidemment, pour qu'un programme soit utile, *certains* effets de bords doivent pouvoir avoir lieu : sortie console, enregistrement en base de données... autant d'effets observables en dehors des fonctions dont on ne peut se passer complètement.

On tache donc d'isoler le plus possible dans nos programmes les fonctions 'pures' des fonctions 'impures' à effets de bords, en les isolant les unes des autres et en les composant intelligemment.

En Javascript, une convention est de retourner '*undefined*' lorsqu'une fonction génère un effet de bord *(return void 0)*. C'est ce que 'vaut' toute fonction n'effectuant pas de return ; *console.info('hello')* par exemple vaut '*undefined*'. C'est l'équivalent du '*void*' qu'on trouve dans d'autres langages.

Les variables, pour une stabilité des programmes... variable.

Une variable est un identifiant auquel est associée une valeur qui peut changer dans le temps. C'est l'une des bases de l'objet : état et mutation d'état. Les problèmes apparaissent lorsque l'état en question est modifié à plusieurs endroits différents dans un programme, parfois par plusieurs threads simultanés. Avec un certain nombre de précautions, et en se limitant à des scopes très localisés (au hasard, le corps d'une fonction), la mutation d'état n'est pas un problème. Mais propager un objet entre plusieurs fonctions, chacune étant autorisée a en altérer les informations, est une source potentielle de bugs, pouvant parfois être très difficiles à comprendre, reproduire et éliminer sans l'aide d'un débuggeur et d'un déroulement pas à pas fastidieux.

Aussi, en programmation fonctionnelle, on préfèrera éviter les mutations d'états et ne travailler qu'avec des constantes - soit en s'appuyant sur une mécanique du langage (les val en Scala), soit en s'interdisant de le faire explicitement, à défaut. L'ultime but étant de ne plus avoir de variable du tout et de ne jongler exclusivement qu'avec des fonctions et des combinaisons de fonctions... et nous verrons que l'on peut aller très loin dans cette direction, en mettant en oeuvre des concepts de PF aux noms exotiques (Combinateurs, Monades, Kleisli...) mais terriblement concrets et efficaces.

Des programmes uniquement composés de fonctions

Ecrire ses programmes avec uniquement des fonctions ? Mais où sont les boucles ?

Parlons-en, des boucles. De quoi est constituée une boucle ? D'un accumulateur, d'une condition d'itération et d'un corps de l'itération.

```
var cahier = [ ],
    pageInitiale = 0,
    pageCourante = 0;

for(; pageCourante < cahier.length; page++){
  effectueCollageSur(cahier[pageCourante]);
}
```

Pas fonctionnel. Améliorons ça : commençons par supprimer la boucle et isoler le corps de celle-ci dans une fonction.

```
var cahier = [ ],
    pageInitiale = 0,
    pageCourante = 0;

function itererSurCahier(cahier) {

  effectueCollageSur(cahier[pageCourante]);

}
```

Problème : notre code ne boucle pas, puisque l'on a bien évidemment supprimé le *for*.

Introduisons une fonction intermédiaire *iterationSuivante* passant la page courante en paramètre :

```
var cahier = [],
    pageInitiale = 0,
    pageCourante = 0;

function itererSurCahier(cahier) {

  function iterationSuivante(page){
    effectueCollageSur(cahier[page]);
  }
}
```

Appelons maintenant naïvement *iterationSuivante* avec *pageInitiale*, juste pour voir :

```
var cahier = [],
    pageInitiale = 0,
    pageCourante = 0;
```

```
function itererSurCahier(cahier) {

  iterationSuivante(pageInitiale);

  function iterationSuivante(page){
    effectueCollageSur(cahier[page]);
  }

}
```

L'appel initial *d'itererSurCahier* se fait en allant 'chercher' la valeur de *pageInitiale* en dehors de la fonction : c'est bien sûr à proscrire, nous allons donc passer cette valeur en paramètre *d'itererSurCahier* :

```
var cahier = [],
    pageCourante = 0;

function itererSurCahier(cahier, pageInitiale) {

  iterationSuivante(pageInitiale);

  function iterationSuivante(page){
    effectueCollageSur(cahier[page]);
  }

}
```

Il ne manque alors plus qu'à réintroduire l'incrémentation et la condition d'itération, puis à itérer :

```
var cahier = [];

itererSurCahier(cahier, 0);

function itererSurCahier(cahier, pageInitiale) {

  iterationSuivante(pageInitiale);

  function iterationSuivante(page){
    effectueCollageSur(cahier[page]);

    if( (page+1) < cahier.length){
      iterationSuivante(page +1);
    }
  }
}
```

Et voilà ! Plus de boucle au sens 'programmation procédurale' de base. Plus de 'variable' de boucle qui change d'état à chaque itération. *La boucle 'for' ne serait donc qu'une sorte de fonction capable de s'auto appeler automatiquement en se passant les paramètres adéquats *?
Vous l'aurez sûrement reconnu : il s'agit de la **récursivité**.

Cette pratique, courante en PF, peut dérouter un oeil néophyte habitué aux structures de contrôle de boucles impératives, mais n'a absolument rien de compliqué à mettre en oeuvre une fois le concept assimilé.
Attention à la condition d'itération pour éviter les boucles infinies !
Pour répondre aux éventuelles interrogations :

- Les appels récursifs présentent un désavantage que nous aborderons dans un futur article (*débordement de pile*)
- Oui, cette forme est plus verbeuse que la boucle *for* ; mais nous découvrirons, là aussi dans un futur article, un outil fonctionnel nous permettant de réduire ce code à... une ligne.
- Oui, j'ai modifié l'objet original (cahier) dans ma fonction créant un effet de bord indésirable en PF ; nous verrons comment résoudre ce problème...

First Class Citizen, Higher order

Un langage de programmation fonctionnel doit me permettre de faire ceci :

```
// Je stocke ma fonction dans une variable => first class citizen
var f = function(){
  ...
};

// Higher Order => ordre supérieur => reçoit des fonctions en paramètres
function callIt(aFunction) {
  aFunction();
}

// Higher Order => ordre supérieur => renvoie des fonctions
function create() {
  return function(param){
    ...
  }
}

// Examples d'utilisation
callIt(f); // fonction stockée précédemment dans f
callIt(function(){
  ...
}); // Fonction anonyme crée à la volée
create()("Nice param");
```

Ceux qui suivent auront reconnu dans l'exemple de fonction *create* le concept vu au début en lambda calcul, dissertant sur l'arité des fonctions à plusieurs paramètres. L'un des grands jeux du programmeur fonctionnel est justement de repérer les structures algorithmiques similaires pour en extraire des abstractions - parfois très très abstraites... - réutilisables.
Le simple fait de pouvoir générer, renvoyer et passer des fonctions à la volée ouvre énormément de possibilités. Ce concept permet à lui seul de remplacer / simuler plusieurs designs-patterns objets impératifs : *strategy, factory, template method, command...*

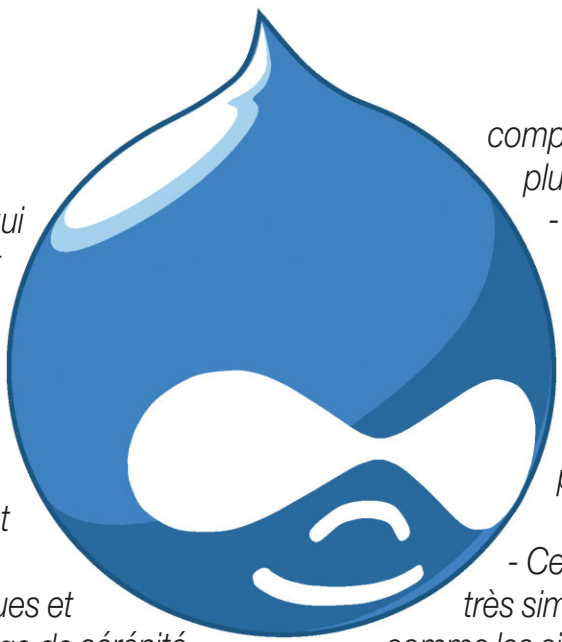
CONCLUSION

A travers quelques exemples simples et un peu de théorie, vous voilà maintenant introduit aux rudiments de la programmation fonctionnelle en Javascript.
La fonction est notre outil principal, et un bon algorithme se doit d'être structuré non en termes de classes s'échangeant des messages et autres changements d'état, mais en termes de combinaisons de fonctions. Ces boîtes noires se combinent, s'appellent en se passant et renvoyant des informations, et en ne travaillant que localement, sans jamais altérer une valeur.
A ce stade, des questions subsistent : comment se passer complètement de variables ? Comment alléger une syntaxe qui peut sembler parfois plus verbeuse que la programmation objet ? Autant de sujets que nous aborderons parmi d'autres, dans un prochain article !

Sébastien **NICHELE**, *Expert technique Java - SQLi*

Drupal : programmation avancée

Qui n'a jamais rêvé d'un terrain de jeu quasiment sans limite pour aller travailler avec entrain et bonne humeur ?



D'autant plus si vous êtes développeur et confronté régulièrement à des projets qui paraissent triviaux et cachent pourtant de nombreux défis techniques.

Savoir que moyennant un effort de réflexion, vous avez sous la main un outil solide, et largement supporté, pour répondre à vos défis techniques et besoins mouvants, est un gage de sérénité dans vos projets.
Drupal est un Système de Gestion de Contenus ou CMS en anglais, open source et entièrement gratuit. Il est réputé pour être extraordinairement polyvalent. En respectant les Bonnes Pratiques de développement Drupal, vous obtenez des sites sécurisés, faciles à faire évoluer, à administrer et à mettre à jour.

Vous pouvez vous lancer facilement en prenant soin de comprendre la logique inhabituelle de cet outil, et réaliser vos premiers sites grâce aux milliers de modules disponibles.

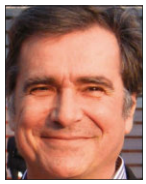
Puis viendra le temps de vous frotter à des besoins avancés, sujets de ce cahier spécial. La complexité en Drupal peut venir de plusieurs sources :

- Du fait des milliers de modules disponibles, il y a plusieurs manières possibles de résoudre un même problème, voire une dizaine de façons de procéder.

- Certains challenges ne sont pas très simples à traiter avec Drupal 7, comme les sites multilingues, les boutiques ou les réseaux sociaux. Dans certains cas des Distributions (Commerce Kickstart, Commons, ...), préconfigurées existent pour faciliter votre travail.

- D'autres besoins sont de niveau avancé par essence, au delà de la solution Drupal, comme les services web, les hautes performances, le responsive web design, les communautés ou boutiques multi-niveaux.

Si vous ne l'avez déjà fait, lancez-vous.
Bonne lecture.



Léon Cros
Président de l'Association Drupal France et Francophonie.
Directeur Technique et Formateur chez Chipway (@chipway)

Migration de données avec le module Migrate

Voilà un sujet qui peut être évoqué durant le cycle de vie d'un projet Drupal. Feeds et Migrate répondent parfaitement à cette problématique, ils ont chacun leurs points forts et leurs limites. Le but de cet article n'est pas d'en préférer un mais on choisit de traiter le sujet du module Migrate. Feeds fera l'objet d'un autre article.

Migrate est une référence dans le domaine d'import de données. Cet article va tenter d'expliquer les concepts généraux du module avec un tutoriel permettant d'importer des articles d'une base de données externe à Drupal et d'en générer des Node de type page.
Les raisons d'une Migration peuvent être diverses :

- Un changement majeur de version donc, récupérer le contenu initial
- Un import de fichier (JSON, XML, CSV, etc.) pour afficher du contenu d'une source externe
- Un import de données depuis un SI qui se base sur une technologie différente de Drupal
- Etc.

Il est évident que cette courbe d'apprentissage joue un rôle très important lorsqu'on jette un coup d'oeil sur la documentation de Migrate. En effet, ce choix de module offre une solution pour les profils développeur, contrairement à Feeds qui est plus "user friendly" avec son interface d'administration.

Architecture de Migrate

Le module Migrate est un Framework qui fournit un cadre flexible et puissant pour la migration de données. Il met à disposition une librairie d'API basé sur une architecture *ETL* Fig.1. Le composant central de sa structure réside dans la classe *Migration*. Cette classe couvre toute la partie métier d'un import de données en implémentant 4 autres classes :

- *MigrateSource* : représente l'origine des sources de données à importer
- *MigrateDestination* : représente quel est l'objet de destination (Node, Taxonomy, User, etc.)
- *MigrateMap* : représente la relation entre l'objet source et l'objet destination, permet aussi de garder toute la trace de cette relation dans le cas d'un rollback
- *MigrateFieldMapping* : représente le mapping entre les champs source et destination

Elles possèdent également des méthodes, principalement pour le formatage avancé des données, telles que *prepareRow()*, *prepare()*, *complete()* ou encore *createStub()* pour des références circulaires par exemple. Il en existe d'autres un peu moins utilisées comme *preImport()*, *postImport()*, *preRollback()* ou *postRollback()* qui agissent comme des callbacks dans les différents cycles de migration. De plus le module fournit aussi ses « Handler Classes », surtout destinées au support des modules de la contribution qui utilisent des types de champs personnalisés.

Passons maintenant à la pratique

Une fois installé, le module fournit une interface qui indique en détail quel est l'état des migrations de vos imports. De plus, il permet d'exécuter le processus d'import, de faire un Rollback, d'interrompre un processus d'exécution, etc. Toutefois, il n'est pas recommandé de lancer un import à travers l'interface et ce, pour des raisons de performances. Pour cela, je vous invite à lire cet article https://www.drupal.org/node/1806824 qui explique en détail les raisons de passer en ligne de commandes *Drush*.

Préparer l'environnement d'exécution

Dans notre exemple, nous allons créer notre module custom pour implémenter notre propre classe *ArticlesMigration* dérivée de la classe principale *Migration*. Pour cela, rien de plus normal que de mettre en place notre structure d'un module custom (.info, .module, etc.). Créons le fichier *articles.info*, contenant le code suivant :

```
name = Articles
description = Custom module d'importation d'articles
core = 7.x

dependencies[] =  migrate

files[] = includes/articles.inc
```

Puis nous mettons en fin de ce fichier le chemin de notre fichier (articles.inc) qui contient le code de notre classe *ArticlesMigration*.

hook_migrate_api()

A la racine de notre module nous créons un fichier nommé *articles.migrate.inc* pour implémenter le hook_migrate_api(). Ce hook fournit une configuration par défaut de notre processus de migration, la version d'API que nous utilisons, et il permet de fournir le nom machine de la classe utilisée (*ArticlesMigration*).

```
// Implements hook_migrate_api().

function articles_migrate_api() {
  return array(
    'api' => 2,
    'migrations' => array(
      'Articles' => array(
        'class_name' => 'ArticlesMigration',
        'group_name' => 'articles',
      ),
    ),
  );
}
```

A ce niveau de l'article, il est légitime de se demander à quel moment le fichier *.module* sera évoqué. Habituellement, ce fichier sera vide, sauf si vous souhaitez définir une route, par exemple, avec le hook_menu() afin de créer une entrée dans le menu Drupal. Créons donc notre fichier *articles.module* vide.

Définition de la classe de migration

La mise en oeuvre de cette classe commence par l'implémentation du constructeur permettant de récupérer toutes les étapes du processus d'une migration de la classe mère *Migration*. Sans oublier de fournir une description de l'import. Cette description apparaît sur l'interface du module Migrate, ou la console si on passe par *Drush*.

```
class ArticlesMigration extends Migration {
  public function __construct($arguments) {
    $this->arguments = $arguments;
    parent::__construct();
```

```
    $this->description = t('Import Articles from ...');
    ...
  }
}
```

settings.php

La source de données dans notre tutoriel est une simple table MySQL contenant 3 champs. Elle contient tous les articles que nous souhaitons importer comme Node de type page dans Drupal.
Il existe deux méthodes :

- Importer toute la table dans notre base de données Drupal,
- Configurer le settings.php pour dire à Drupal où se trouve cette base de données qu'on souhaite attaquer.

Dans notre cas, on utilise la deuxième méthode.

```
$databases['articles'] = array(
 'default' => array(
   'database' => 'migrate',
   'username' => 'root',
   'password' => 'pwd',
   'host' => 'localhost',
   'driver' => 'mysql',
 ),
);
```

Se connecter à la source

Une fois le fichier *settings.php* configuré, il suffit de se connecter à la base de données et de faire un SELECT sur les champs qu'on souhaite importer Fig.2.

```
$query = Database::getConnection('default', 'articles')
       ->select('articles', 'a')
       ->fields('a', array('id_article','titre', 'description'));
```

L'objet source

La classe *MigrateSource* a pour métier d'encapsuler les données qui doivent être importées. Dans notre exemple, il s'agit d'importer des données à partir d'une table MySQL, et par conséquent on utilisera la classe *MigrateSourceSQL*. Le constructeur de cette classe prend une *$query* comme paramètre pour la provenance des données.

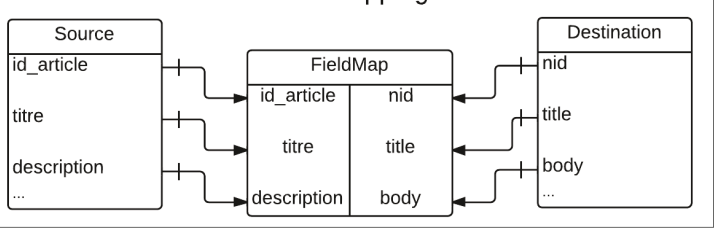
```
$this->source = new MigrateSourceSQL($query);
```

Toutes les sources acceptent un éventail d'options en paramètres pour modifier le comportement par défaut. Ces options sont implémentées dans la classe mère *MigrateSource* (c'est à dire qu'ils sont disponibles pour toutes les catégories de sources).

```
$this->source = new MigrateSourceSQL($query, array(), NULL,
array('cache_counts' => TRUE));
```

Enfin, il existe plusieurs types de classes source (CSV, JSON, HTML, Oracle, XML, etc.), il est même possible d'écrire soi-même une classe qui détermine la typologie de notre source (voir la documentation sur cette URL https://www.drupal.org/node/1006986).

Fig.3



L'objet de destination

Chaque mise en oeuvre d'une migration doit être créée avec une instance destination bien spécifique. En effet, il existe plusieurs classes qui permettent d'instancier l'objet destination... L'Entity qu'on souhaite importer dans notre exemple est de type Node, c'est pourquoi on instancie notre destination avec la classe *MigrateDestinationNode* qui prend en paramètre un *Bundle* (type de contenu).

```
$this->destination = new MigrateDestinationNode('page');
```

Field mapping

Dans cette dernière section du tutorial, nous examinons les classes sources de Migrate et comment le mapping se fait avec destination. *MigrateSQLMap* est une classe qui permet de créer les tables dans la base de données qui se chargent de lier les clés de source et de destination. Le premier paramètre est le suffixe des tables qui vont être créées. Le deuxième paramètre définit la clé de la source de données. D'un autre côté nous utilisons la classe intégrée *MigrateDestinationNode*. Drupal sait déjà quel est le type de données. Ainsi, la clé est fournie au travers de la méthode statique *getKeySchema()*.

```
$this->map = new MigrateSQLMap(
 $this->machineName,
 array(
   'id_article' => array(
     'type' => 'int',
     'unsigned' => TRUE,
     'not null' => TRUE,
   ),
 ),

   MigrateDestinationNode::getKeySchema()
 );
```

Pour finir, nous devons définir le mapping des champs de source et destination à travers la méthode *addFieldMapping* qui prend le champ destination comme premier paramètre et le champ source comme second Fig.3.

```
$this->addFieldMapping('title', 'titre');
$this->addFieldMapping('body', 'description');
```

Il ne nous reste plus qu'à lancer notre procédure de migration, et pour cela nous ne pouvons que recommander d'utiliser la commande drush « migrate-import » (mi) : ex. drush mi Articles
pour importer les articles Fig.4. Une liste des commandes est disponible ici : https://www.drupal.org/node/1561820.

Conclusion

L'objectif de ce tutoriel est de comprendre les principaux concepts du développement avec Migrate et aborder le sujet. En effet, choisir ce module comme solution de vos imports de données doit être mûrement réfléchi car il nécessite des compétences en développement orienté objet. La meilleure solution est toujours la plus simple. Si la configuration de la migration n'est pas trop complexe, alors Feeds remplit très bien son rôle. Mais si la manœuvre est un peu moins orthodoxe, avec des relations entre les entités, des traitements à effectuer sur les champs ou leur format, etc., alors l'arsenal que propose Migrate est surement plus adapté.

Tarik **Larbi**
Chef de projet technique chez Trained People

Créer son entité avec l'API de Drupal

Vous souhaitez utiliser Drupal comme framework, et aller plus loin que ses nœuds, commentaires, taxonomie et utilisateurs (tous les 4 sont des entités). Cet article se veut pratique puisque l'objectif est de créer une entité disponible pour n'importe quel site Drupal 7. L'exemple utilisé sera celui d'une entité post-it et de deux bundles : tache et liste de course.

Le concept d'entité

Une entité est une sorte de classe abstraite de laquelle vont dériver des bundles (on pourrait parler de classe). L'idée est que l'on va déclarer une entité avec un titre, et au moins un champ obligatoire et commun à nos bundles et certaines propriétés. Pour cela il faut déclarer cette entité comme un module Drupal avec certaines spécificités qui sont détaillées dans cet article.

LA DÉCLARATION DE L'ENTITÉ

Pour commencer, créer un dossier de module qui contiendra notre entité dans le répertoire sites/all/modules nommé postit.
Celui-ci contiendra pour commencer, deux fichiers : un postit.info et un postit.module puisque ces deux fichiers sont obligatoires pour déclarer un module Drupal. Voici le postit.info :

```
name = Postit
description = "Entité Postit et ses bundle (tache et course)"
version = 7.0.1
core = 7.x
```

Le fichier est donc une simple déclaration de module comme tous les autres modules de Drupal. Ensuite on va déclarer notre entité en utilisant la fonction entity_info dans le fichier postit.module

```
<?php

<?php

function postit_entity_info() {
 $entity = array();
 $entity['postit'] = array(
 // On définit l'étiquette du type d'entité
 'label' => t('Post-it'),
 // La table définie dans le especes.install
 'base table' => 'postit',
 'uri callback' => 'postit_uri',
 // On définit les clés liées à l'entité
 'entity keys' => array(
   // L'identifiant utilsée par entity_load
   'id' => 'pid',
 ),
     // La clé pour les bundles
 'bundle keys' => array('bundle' => 'type'),
 'bundles' => array()
 );
 return $entity ;
}
```

L'entité post-it est donc créée au niveau de Drupal, par contre dans l'état Drupal reconnaît une entité mais il reste du travail puisque rien ne peut être fait avec celle-ci.
Pour résumer, ce qui a été créé : une entité avec un nom, un identifiant unique, une table à utiliser (qui doit encore être créée) et la fonction que Drupal va utiliser pour afficher une entité Post-It.
Le fichier postit.install permet de créer la table nécessaire :

```
<?php
function postit_schema() {
 $schema['postit'] = array(
   'fields' => array(
     'pid' => array(
       'type' => 'serial',
       'unsigned' => TRUE,
       'not null' => TRUE,
       ),
     'type' => array(
       'description' => 'Le type de postit',
       'type' => 'text',
       'size' => 'medium',
       'not null' => TRUE
       ),
     'title' => array(
       'type' => 'text',
       'size' => 'medium'
       ),
     ),
   /*On définit la clé primaire de la table */
   'primary key' => array('pid')
   );
   return $schema;
};
```

Pour cela le hook schema est utilisé et sera appelé lors de l'installation du module Post-It. Cette table contient donc un identifiant unique (le pid), un titre et le type de bundle. L'entité Post-It a bien été créée, il reste à déclarer les deux bundles. Pour cela retourner dans la foncton postit_entity_info pour ajouter les lignes suivantes :

```
 // Bundle Tache
 $entity['postit']['bundle']['tache'] = array(
 'label' => t('Tache'),
 // On définit les éléments pour l'administration de ce type de postit
 'admin' => array(
   // Le chemin interne d'accès aux pages d'administration
   'path' => 'admin/postit/%postit_type',
```

actency

A CREATIVE AGENCY
POWERED BY DRUPAL

Audit
- applicatif
- performance
- sécurité

Maintenance
- préventive
- corrective
- évolutivet

Architecture technique

Développement

Théming

Support

Formation

Hébergement

PROPULSEZ VOTRE PROJET
AVEC L'AGENCE **100% DRUPAL**

ACTENCY PARIS
82 rue d'Hauteville / 75010 PARIS

ACTENCY STRASBOURG
45 avenue de Colmar / 67000 STRASBOURG

01 47 70 47 70

www.actency.fr

contact@actency.fr

```
    // Le chemin d'administration affiché aux utilisateurs
    'real path' => 'admin/postit/tache',
    // L'argument %postit_types doit être passé
    'bundle argument' => 2,
    // On ajoute la permission, ici on utilisera Administrer les nœuds
    'access arguments' => array('administer nodes')
  )
  );

    // Bundle Course
  $entity['postit']['bundle']['course'] = array(
    'label' => t('Course'),
    'admin' => array(
      'path' => 'admin/course/%course_type',
      'real path' => 'admin/postit/course',
      'bundle argument' => 2,
      'access arguments' => array('administer nodes')
    )
  );
```

Les deux bundles Tache et Course sont ainsi ajoutés à l'entité Post-It. Un seul bundle aurait pu être ajouté. Une fois cela fait, voyons en détail quelles sont les fonctions nécessaires pour que l'entité Post-It fonctionne.

L'administration de notre entité et les fonctions de base

Pour commencer, en restant dans le postit.module, la première fonction à déclarer est la fonction appelée par postit_entity_info, il s'agit de postit_uri qui permet de définir le modèle de routage de notre entité. Celle-ci s'insère dans le fichier postit.module :

```
function postit_uri($postit) {
  return array('path' => 'postit/' .$postit->pid);
}
```

L'entité nécessite également la fonction postit_menu qui permettra à notre entité Post-It de savoir les fonctions qu'elle peut utiliser. Cette fonction est donc d'ajouter au fichier postit.module :

```
function postit_menu() {
  $items = array();

  // La page d'administration générale
  $items['admin/postit'] = array(
    'title' => 'admin postit',
    'access arguments' => array('access content'),
    'page callback' => 'postit_admin',
    'file' => 'postit.pages.inc',
    'type' => MENU_NORMAL_ITEM
  );
  //La page d'administration par bundle
  $items['admin/postit/%postit_type'] = array(
    'title callback' => 'postit_type_title',
    'title arguments' => array(2),
    'access arguments' => array('access content'),
    'page arguments' => array(2)
  );
  //La page d'ajout d'un postit
  $items['admin/postit/%postit_type/add'] = array(
    'title' => 'add',
    'access arguments' => array('access content'),
    'type' => MENU_DEFAULT_LOCAL_TASK
  );
  //La page d'affichage d'un postit
  $items['postit/%postit'] = array(
    'title callback' => 'postit_title',
    'title arguments' => array(1),
    'page callback' => 'postit_display_one',
    'page arguments' => array(1),
    'access arguments' => array('access content'),
    'file' => 'postit.pages.inc',
    'type' => MENU_CALLBACK
    );
  //L'onglet voir de l'affichage
  $items['postit/%postit/view'] = array(
    'title' => 'view',
    'access arguments' => array('access content'),
    'weight' => -3,
    'type' => MENU_DEFAULT_LOCAL_TASK
  );
  //L'onglet modifier de l'affichage
  $items['postit/%postit/edit'] = array(
    'title' => 'edit',
    'page callback' => 'drupal_get_form',
    'page arguments' => array('postit_edit', 1),
    'access arguments' => array ('access content'),
    'file' => 'postit.pages.inc',
    'type' => MENU_LOCAL_TASK
  );
  return $items;
}
```

Cette fonction permet de déclarer l'ensemble des actions de création, recherche/affichage et modification d'une entité. Pour chaque chemin déclaré correspond une fonction; il reste à déclarer ces fonctions. Mais certaines fonctions sont aussi nécessaires et à ajouter dans le fichier postit.module :

```
function postit_type_load($postit_type){
  switch($postit_type) {
    case 'tache':
    case 'course':
      return $postit_type;
    default :
      return FALSE;
  }
}

function postit_type_title($postit_type) {
  return $postit_type;
}

function postit_title($postit) {
  return $postit->title;
}

function postit_load($pid) {
  $postit = entity_load('postit', array($pid));
  return $postit[$pid];
}
```

Les trois premières fonctions sont appelées dans la fonction postit_menu et la dernière fonction (postit_load) est obligatoire pour toute entité, puisque c'est ce qui permet de récupérer les informations d'une entité avec son identifiant. Notre entité est créée, il reste à créer le fichier postit.page.inc appelé dans les différentes fonctions et à créer les fonctions correspondantes de postit_menu.
Pour commencer, créer le fichier postit.pages.inc dans le dossier sites/all/modules/postit/.
La fonction appelée par l'admin sera la première à écrire :

```
function postit_admin($type = NULL){
 if ($type) {
   return drupal_get_form('postit_addpostit', $type);
 }
 else {
    $rows = array();
    $rows[] = array(l(t('Tache'), 'admin/postit/tache'));
    $rows[] = array(l(t('Course'), 'admin/postit/course'));
    $header = array('Type');
    $content = array(
      '#theme' => 'table',
      '#header' => $header,
      '#rows' => $rows
    );
    return $content;
 }
}
```

Fig.1

Lors de l'affichage de la page admin/postit voici le résultat obtenu :
Lors du clic sur un des types de bundle, puisque la fonction d'ajout n'est pas définie, une erreur s'affiche, celle-ci sera corrigée dès que les formulaires d'ajout de nos bundle seront écrits.

L'affichage d'une entité

Pour l'affichage d'une entité, une entité a été saisie via phpmyadmin sur le site pour tester, celle-ci a pour pid 1. Au niveau du code, créer le fichier postit.pages.inc dans le dossier sites/all/modules/postit/. Dans ce fichier, les fonctions suivantes sont déclarées :

```
<?php

function postit_display_one($postit) {
 $content[] = array(
       '#markup' => array( $postit->title, 'postit/' . $postit
->pid ),
       );
 return $content;
}
```

A l'url monsite/postit/1 voici ce qui s'affiche : Fig.2.

L'erreur vient du fait que notre post-it n'a pas de contenu, puisqu'aucun champ n'a été défini lors de la création de l'entité ou des bundles. Les champs seront ajoutés juste après la création du formulaire d'ajout de notre entité.

Le formulaire de création / modification / suppression d'une entité

Pour l'ajout d'entité, 3 fonctions sont appelées : une première qui définit le formulaire, une seconde qui valide son contenu et une dernière qui enregistre (soumet) son contenu.

```
//Fonction d'ajout d'un postit
function postit_addpostit($form, &$form_state, $type) {
 $form['title'] = array(
   '#type' => 'textfield',
   '#title' => 'title'
 );
 $postit = new stdClass();
 $postit->type = $type;

 $form['postit'] = array(
   '#type' => 'value',
   '#value' => $postit
 );

 $form['type'] = array(
   '#type' => 'value',
   '#value' => $type
 );

 $form['actions'] = array('#type' => 'actions');
 $form['actions']['add'] = array(
   '#type' => 'submit',
   '#value' => t('add')
 );
 return $form;
}
function postit_addpostit_validate($form, &$form_state) {
```

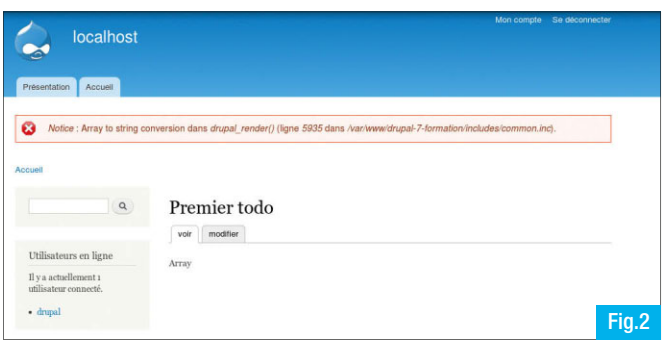


Fig.2

Affichage de notre premier postit

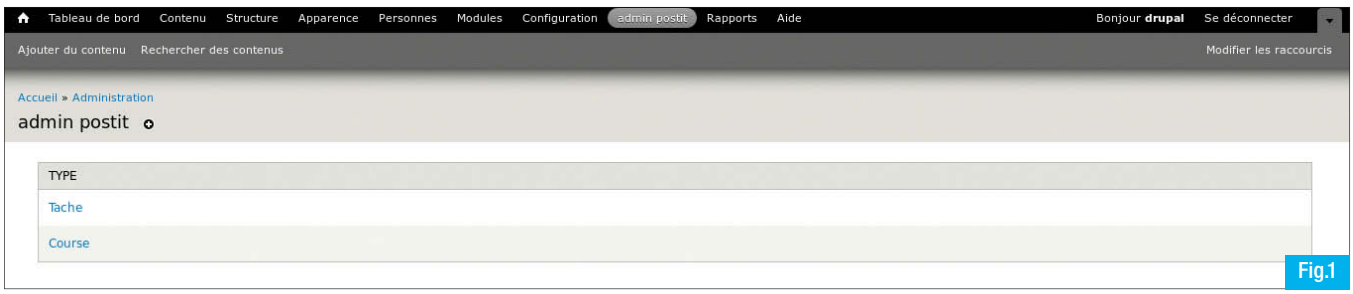


Fig.1

Page d'administration des post-it

```
  entity_form_field_validate('postit', $form, $form_state);
}
function postit_addpostit_submit($form, &$form_state) {
  $postit = $form_state['values']['postit'];
  $postit->title = $form_state['values']['title'];
  drupal_write_record('postit', $postit);
  entity_form_submit_build_entity('postit', $postit, $form,
 $form_state);
  drupal_set_message(
      t('Nouveau @type a été ajouté',
      array('@type' => $postit->type))
      );
}
```

Ce qui donne lors de l'appel à l'url monsite/admin/postit/course : Fig.3. Il est donc dès à présent possible de saisir des entités postit de type course ou de type tâche (ou tout autre bundle qui aurait été défini dans l'entité postit). Pour le formulaire d'édition (l'onglet Modifier lors de l'affichage du postit), le principe est le même avec 3 fonctions similaires :

```
function postit_edit($form, &$form_state, $postit) {
  $form['title'] = array(
    '#type' => 'textfield',
    '#title' => 'title',
    '#default_value' => $postit->title
  );
  $form['postit'] = array(
    '#type' => 'value',
    '#value' => $postit
  );

  $form['type'] = array(
    '#type' => 'value',
    '#value' => $postit->type
  );

  $form['actions'] = array('#type' => 'actions');

  $form['actions']['save'] = array(
    '#type' => 'submit',
    '#value' => t('save')
  );

  $form['actions']['delete'] = array(
    '#type' => 'submit',
    '#value' => t('delete'),
    '#submit' => array('postit_edit_delete')
  );
  return $form;
}
```

```
function postit_edit_validate($form, &$form_state) {
  entity_form_field_validate('postit', $form, $form_state);
}

function postit_edit_submit($form, &$form_state) {
  $postit = $form_state['values']['postit'];
  $postit->title = check_plain($form_state['values']['title']);
  drupal_write_record('postit', $postit, array('pid'));
  entity_form_submit_build_entity('postit', $postit, $form,
 $form_state);

  drupal_set_message(
      t('Le @type a été sauvegardé',
      array('@type' => $postit->type))
      );

  $form_state['redirect'] = 'postit/' . $postit->pid;
}
```

Les fonctions sont quasiment identiques. La différence est que lors de l'enregistrement on passe le pid comme argument, pour que Drupal fasse une mise à jour de la ligne correspondante en base et qu'à la fin du formulaire on redirige l'affichage vers la page "voir" du pid correspondant. Voici le formulaire : Fig.4. Une autre différence, est qu'un deuxième bouton d'action a été ajouté dans le formulaire il s'agit du bouton de suppression de l'entité. Celui-ci fait appel à la fonction postit_edit_delete qui s'ajoute également dans le postit.pages.inc et que voici :

```
// Fonction de suppression de l'entité
function postit_edit_delete($form, &$form_state) {
  $postit = $form_state['values']['postit'];

  db_delete('postit')
    ->condition('pid', $postit->pid)
    ->execute();
  $form_state['redirect'] = 'postit';
}
```

Les formulaires d'ajout / création / suppression de l'entité postit sont créés et fonctionnent. L'entité est donc pleinement opérationnelle. Pour aller plus loin avec les entités, il est possible d'associer la puissance de l'API de field à l'API des entités découvertes ici. Le potentiel de Drupal est donc quasiment infini avec ces API et permet de faire des choses très variées bien plus que d'autres CMS.

Vanessa **Kovalsky David,** *Lead développeuse Drupal / PHP chez Webnet* - http://vanessakovalsky.net
Membre actif de DrupalFr, et de la promotion du logiciel libre avec l'ALDIL
Lead dev sur le projet Esecouristes : https://github.com/vanessakovalsky/esecouristes

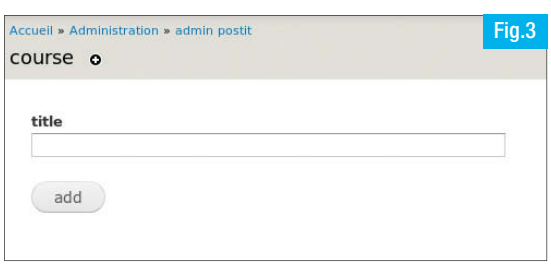


Fig.3

Formulaire d'ajout d'un postit de type course

Fig.4

Formulaire de modification d'un postit

TRAINEDPEOPLE

7 ans d'experience

180 jours de formation par an

600 sociétés nous font confiance

+DE 900 stagiaires ont suivi nos formations

9 formations personnalisables

FORMATION DRUPAL

DÉVELOPPEZ VOS COMPÉTENCES DRUPAL

- CONTRIBUTEUR / PUBLISHER
- PERFORMANCE & SÉCURITÉ
- WEBMASTER
- DÉPLOIEMENT
- THEMEUR / DESIGNER
- RESPONSIVE DESIGN
- DÉVELOPPEUR
- COMMERCE

8 NOUVEAU

INTRODUCTION À DRUPAL 8

Découvrez les principales nouveautés de Drupal 8, et apprenez les bases pour vous lancer !

ACCOMPAGNEMENT DE PROJET DRUPAL

UNE ÉQUIPE D'EXPERTS À VOTRE SERVICE

- CONSEIL
- ASSISTANCE TECHNIQUE
- AUDIT
- SUPPORT
- MAINTENANCE

Plus d'infos sur le site

www.drupalfrance.com

T / +33 1 46 94 69 32
F / +33 1 48 24 95 80

8 rue du Faubourg Poissonnière / 75010 Paris / contact@trainedpeople.com / www.trainedpeople.com

Un album photo en Drupal

Que faire des photos de vacances ? Vous voulez les publier et les partager avec la famille et les amis ? Il existe de nombreux projets PHP pour gérer une galerie de photos. Cependant si vous possédez déjà un blog ou un site web, vous pouvez avoir envie de posséder votre propre galerie de photos.

Sous Drupal 7, on dispose de différents modules de Galerie d'images ou de fichiers qui répondent à certains types de besoins qui sont peut-être malheureusement différents des vôtres; cet article va vous expliquer une méthode pour classer vos photos dans des dossiers et sous-dossiers.

Préambule

Pour réaliser vos différents albums, vous devez avoir installé au préalable Drupal 7, disponible sur le site de l'association Drupal France et Francophonie (www.drupalfr.org). Après l'installation du CMS, nous utilisons certains modules de la partie Core de Drupal, mais aussi certains modules de la communauté. Notre choix va se porter sur ceci :

- References : https://drupal.org/project/references

Il améliore les options de 'node_reference',

- Node Reference URL Widget : https://drupal.org/project/nodereference_url

Il ajoute un nouveau widget dans le champ 'node reference' qui se trouve dans le type de contenu,

- Views : https://drupal.org/project/views

Il permet de créer des vues à travers l'interface d'administration,

- EVA (Entity Views Attachment) : https://drupal.org/project/eva

Il fournit un plugin d'affichage pour le module Views. Il permet à la sortie de la vue d'être fixé sur le contenu de toute entité Drupal,

- Colorbox : https://drupal.org/project/colorbox

Il permet de customiser le plugin 'lightbox' de jQuery,

- path_auto : https://www.drupal.org/project/pathauto

Il génère automatiquement des alias.

Créer les types de contenu

Pour réaliser notre album, il est nécessaire de préparer 3 types de contenus différents qui seront reliés ensemble. Tout d'abord, il nous faut un type « album ». Chaque album se composera de catégories (appelées "famille") et dans chacun d'eux, nous retrouvons les photos. Fig.1.

Etape 1: création d'un album

Nous créons un type de contenu «album », qui servira de porte d'entrée. Ce type de contenu ne possède aucune spécification précise. Cependant, nous désactivons les options des commentaires et décochons la ligne 'Promu en page d'accueil'

Etape 2 : les dossiers

Cette étape permettra de retrouver les différents dossiers liés à un album précis. Pour ce faire, nous créons un type de contenu "dossier". Tout d'abord, nous désactivons les options des commentaires et décochons la ligne 'Promu en page d'accueil'. Ensuite, nous ajoutons un nouveau champ avec les critères suivants :

Etiquette : album référence
Nom Système : field_album_ref
Type de champ : Node reference
Widget : Reference from URL

Fig.2.

Bien entendu, lorsque ce champ est créé, vous le paramétrez avec le type de contenu qui a été créé précédemment, c'est à dire Album Fig.3.

Etape 3 : les photos

Ce dernier type de contenu 'photos', mémorisera l'ensemble des images et photos que vous aurez uploadées. C'est pourquoi nous créons un type de contenu 'photos.', associé à un type 'dossier'. Tout d'abord, nous désactivons les options des commentaires et décochons la ligne 'Promu en page d'accueil'. Ensuite, nous ajoutons deux nouveaux champs.

Fig.1

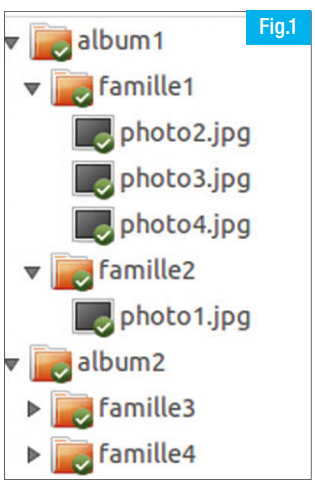


Fig.3

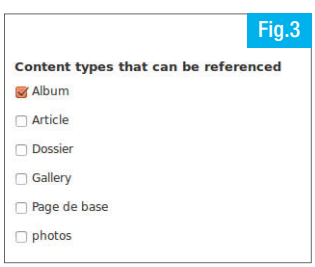


Fig.2

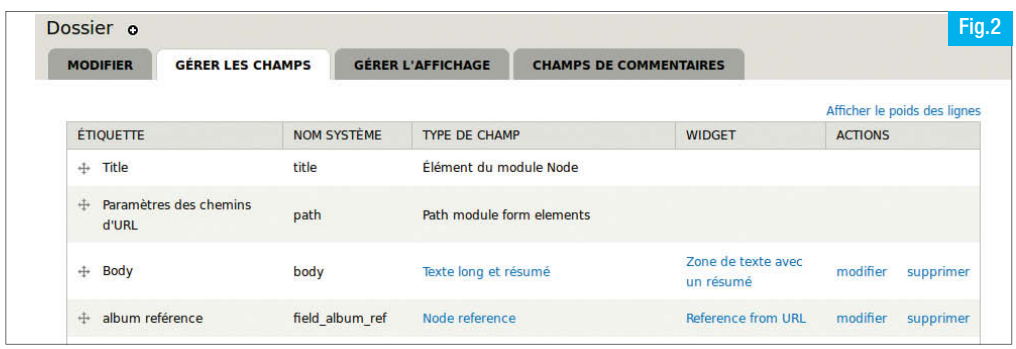
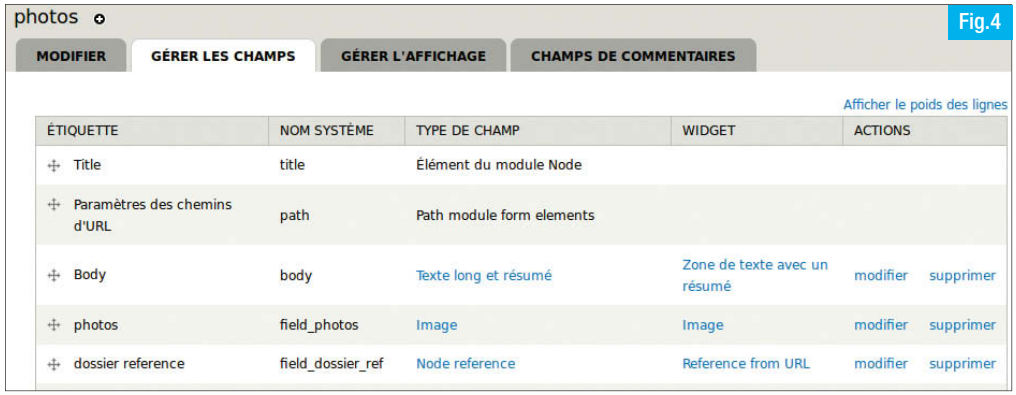


Fig.4



Le premier champ se découpe de la manière suivante :

Etiquette : photos
Nom Système : field_photos
Type de champ : image
Widget : image

Il est possible de configurer le type image comme vous le souhaitez :

- Cocher la case obligatoire
- Définir les dimensions d'affichage, Etc.

Le deuxième champ (Fig.4.) à créer, est :

Etiquette : dossier référence
Nom Système : field_dossier_ref
Type de champ : Node reference
Widget : Reference from URL

Bien entendu, lorsque ce champ est créé, vous paramétrez le type de contenu créé précédemment, c'est à dire dossier (Fig.5.).

Naviguer dans les albums

La navigation dans les albums peut s'effectuer de différentes manières. Ici nous choisirons une méthode parmi les autres et à partir du module Views. Pour cela, vous devez créer une vue avec les différentes entités. Le but est de simplifier la navigation à l'intérieur dans notre album photo.

> **Attention :** *Lorsque vous créez différentes entités dans une vue, il faut bien choisir Override (supplanter)*

Etape 1 : point d'entrée

Lorsque vous souhaitez voir des photos, vous devez choisir un album. L'affichage des albums se traduit par une vue. Ainsi, vous effectuez les opérations suivantes :

Créez une nouvelle vue (Structure > Vues > Ajouter une nouvelle vue)

Remplir

- Nom de la vue : Albums photos
- Saisir le nom d'une page et d'une URL qui sera utilisé par le menu (Fig.7.)
- Cliquez sur Continuer et modifier

Fig.6.

Ensuite vous arrivez à l'intérieur de la vue comme le montre l'image 7, la vue est déjà complètement renseignée.

Nous en profitons pour modifier certains paramètres d'affichages :

Format : Grid
Afficher : Fields
Type = Album

Etape 2 : contenu d'un album

Un album photo peut contenir différentes catégories ou dossiers de rangement pour vous faciliter la navigation dans l'album, ainsi que la recherche de photos. Nous devons ajouter une nouvelle entité dans notre vue, que vous effectuez de la manière suivante :

Cliquez sur le bouton Ajouter et sélectionnez le contenu entité 'Eva entité' (voir Fig.8.) Lors de l'affichage de cette nouvelle entité, vous devez renseigner différentes informations comme le montre l'image 9.

La colonne 1 :

- Saisir le display name (nom de l'entité) : Les dossiers de l'album
- Titre : Choisissez un dossier de l'album

Saisir dans la colonne 2 :

- Entity Type (Type entité) : Noeud
- Bundle (paquet) : Album
- Arguments : id
- Show titre (afficher le titre) : oui

Saisie dans la colonne 3 :

Contextual filters (filtre contextuel) : Contenu Album référence

- Choisir Provide default value (valeur par défaut)
- Content ID from URL (ID du contenu depuis l'URL)

Etape 3 : contenu d'un dossier

Le dossier d'un album photo affiche une ou plusieurs photos. Chacune d'elle a été uploadée à partir du contenu (voir la partie concernée). Nous devons ajouter une nouvelle entité dans notre vue, ce que vous effectuez de la manière suivante :

- Cliquez sur le bouton Ajouter et sélectionnez le contenu entité 'Eva entité' (Fig.9)
- La configuration de cette nouvelle entité, s'effectue de la manière suivante (voir Fig.10)

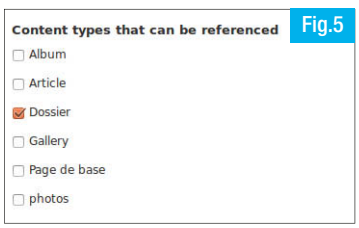


Fig.5

Fig.8

Fig.6

Fig.7

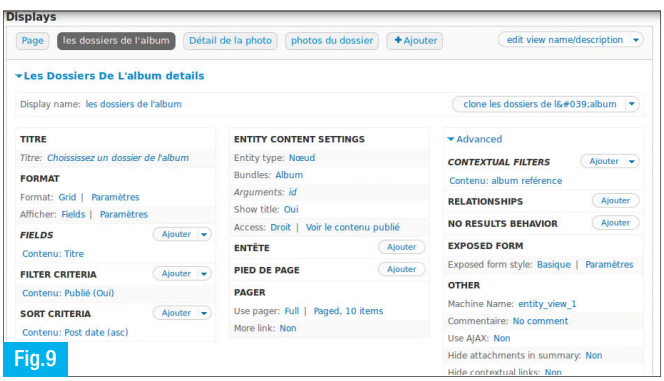
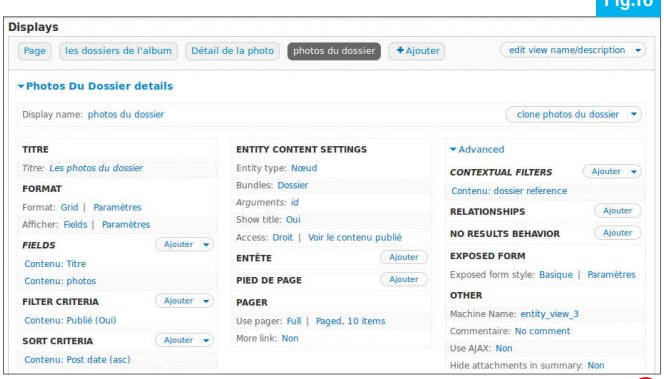


Fig.9

Fig.10



La colonne 1 :
- Saisir le display name (nom de l'entité) : photos du dossier
- Titre : Les photos du dossier
- Format : Grid
- ajouter un Field (champ) : photos
  - Décocher le label (étiquette)
  - Choisir le format (mise en forme) : image ou colorbox
  - Style de l'image : thumbnail (vignette)
  - Lier à l'image : contenu

Saisir dans la colonne 2 :
- Entity Type (Type entité) : Noeud
- Bundle (paquet) : Dossier
- Arguments : id
- Show titre (afficher le titre) : oui

Saisie dans la colonne 3 :
Contextual filters (filtre contextuel) : Contenu : Dossier référence
- Choisir Provide default value (valeur par défaut)
- Content ID from URL (ID du contenu depuis l'URL)

Etape 4 : détail de la photo

La dernière étape est optionnelle, mais utile surtout si vous souhaitez zoomer une photo et voir des informations supplémentaires. Il sera visible pour l'utilisateur à partir du contenu de dossier (étape 3). Nous ajoutons une nouvelle entité dans notre vue, que vous effectuez de la manière suivante :
- Cliquez sur le bouton Ajouter et sélectionnez le contenu entité 'Eva entité' (image 8)
- La configuration de cette dernière entité, s'effectue de la manière suivante (voir Fig.11)

La colonne 1 :
- Saisir le display name (nom de l'entité) : Détail de la photo
- Titre : Votre photo sélectionnée
- Format : Grid
- Ajouter un Field (champ) : photos
  - Décocher le label (étiquette)
  - Choisir le format (mise en forme) : image ou colorbox
  - Style de l'image : Medium (moyen)
  - Lier à l'image : contenu

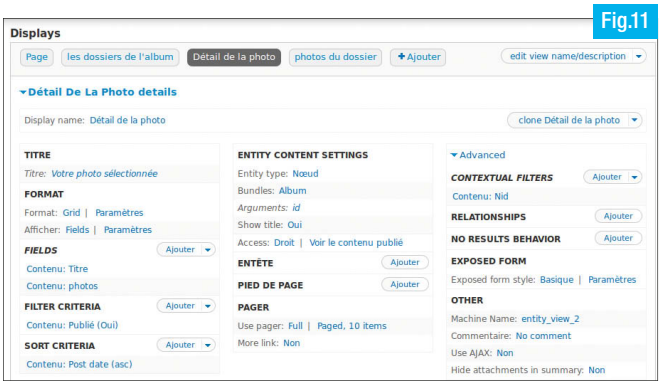


Fig.11

Fig.12

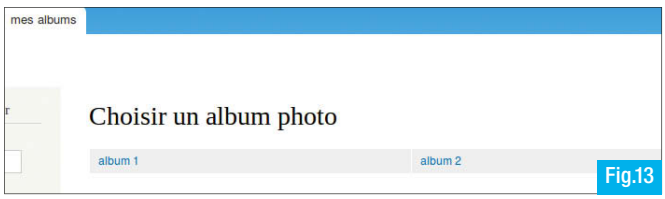


Fig.13

Saisir dans la colonne 2 :
- Entity Type (Type entité) : Noeud
- Bundle (paquet) : Album
- Arguments : id
- Show titre (afficher le titre) : oui

Saisie dans la colonne 3 :
Contextual filters (filtre contextuel) : Contenu : Dossier Nid
- Choisir Provide default value (valeur par défaut)
- Content ID from URL (ID du contenu depuis l'URL)

Créer du contenu

Cette partie s'effectue souvent en parallèle de la création des vues. Elle permet d'obtenir les résultats en même temps que la préparation de la vue. Pour ajouter du contenu et illustrer notre album, une certaine procédure doit être respectée c'est à dire,
- Vous créez un ou plusieurs albums (Fig.12)
- Vous créez des dossiers associés aux albums (Fig.13)
- Vous uploadez les photos avec le dossier parent associé (Fig.14)

Affiner l'affichage

Maintenant que le contenu et la navigation sont prêts, vous devez affiner l'affichage, c'est à dire que certains champs seront cachés car ils ne sont pas utiles et gâchent la présentation souhaitée. Pour cela vous vous rendez dans chacun des types de contenus que vous avez créés plus haut (Albums, dossier, photos) de la manière suivante :

structure > type de contenu > dossier
onglet : Gérer l'affichage
Pour cet onglet, vous cachez le champ : 'Album référence'
Cliquez sur Enregistrer

En résumé

L'album photo présenté dans cet article va vous servir de base ; vous pouvez le faire évoluer avec d'autres modules tout en customisant les vues d'affichage. Les sources de l'article sont disponibles sur le site du magazine sous la forme d'une feature.

Christophe Villeneuve
Consultant IT pour Neuros, auteur du livre "Drupal7 en mode avancé" aux éditions Eyrolles et auteur Editions ENI, Rédacteur pour WebRIVER, membre des Teams DrupalFR, AFUP, LeMug.fr, Drupagora, PHPTV..

Fig.14

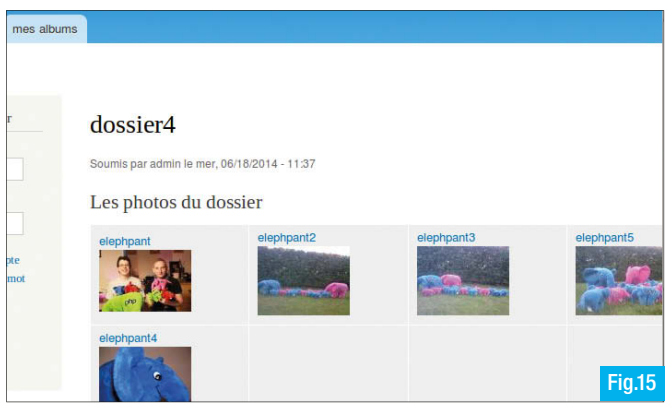
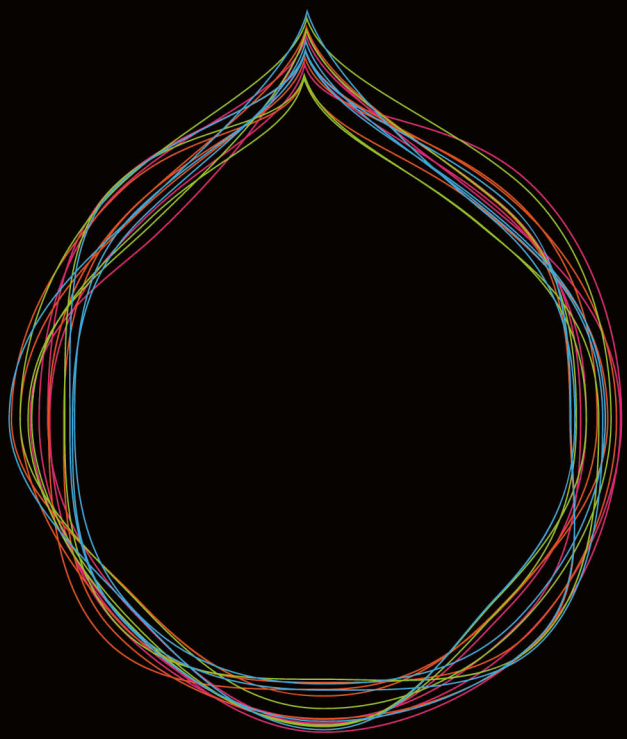


Fig.15

acquia.com



Une plate-forme digitale unique qui intègre Commerce, Contenu et Communautés

Optimisez et Personnalisez l'expérience utilisateur sur votre site avec Acquia Lift

Réduisez votre Time to Market avec Acquia Cloud Site Factory

Créez des experiences e-Commerce incomparables avec Acquia Commerce Cloud

Acquia | THE DIGITAL BUSINESS COMPANY

Drupal 7 : Thème Basic

basic

Lorsque l'on débute un nouveau site, on se demande souvent quel thème de base utiliser. Il existe une multitude de thèmes qui offrent différentes fonctionnalités, parfois fantastiques, parfois inutiles. Sur Drupal, il est un thème que je vous propose de découvrir : Basic1.

Quels sont les besoins initiaux ?

Dans la plupart des cas, ce que l'on souhaite d'un thème, c'est qu'il nous instaure de bonnes bases sans nous encombrer avec de lourdes fonctions desquelles nous ne nous servirons peu, voire pas du tout. Le plus important est de choisir un thème qui gère le responsive, ayant une architecture de fichiers organisée et qu'il soit facile d'utilisation en utilisant les dernières technologies/fonctionnalités que l'on peut voir ailleurs. Basic est un thème qui répond à ces prérequis, ni plus ni moins.

Basic, késako ?

Comme son nom l'indique clairement, Basic est un thème Drupal simple. Il est disponible en version de production sur Drupal 6 et 7, ainsi qu'en version bêta sur le futur Drupal 8.

Ce thème est construit autour des principes modernes du web. Ainsi, la structure des pages est composée de HTML5. Le thème est également basé sur le préprocesseur SASS qui procure une meilleure organisation des feuilles de style et permet l'intégration de variables, fonctions, etc(2). De plus, afin d'optimiser d'avantage le travail d'intégration, Basic intègre nativement les librairies Bourbon et Neat. Il est possible cependant d'utiliser Compass si vous êtes plus familier avec celui-ci.

Et ensuite ?

Je vois votre hâte, vous allez me dire : "Oui et alors ? Il est possible de faire ça avec tous les thèmes déjà !". Oui, mais non. Basic intègre également un fichier que l'on peut appeler fichier de configuration ayant pour but d'instaurer les bases du site. Ce fichier est parfaitement modifiable et il vous permettra d'altérer rapidement l'aspect primaire de votre site. Voici quelques exemples d'éléments qui le composent : **Fig.1.**

$visual-grid: true
$visual-grid-color: #EEEEEE
$responsive: true
$column: 60px
$gutter: 20px
$grid-columns: 15
$max-width-px: 1088
$max-width: em($max-width-px, 16)
$max-width-fluid: new-breakpoint(max-width $max-width-px + px $grid-columns)
$tablet: new-breakpoint(max-width 768px 8)
$mobile: new-breakpoint(max-width 480px 4)

Les fichiers SASS sont organisés selon le type d'élément à designer. Cela permet une compréhension plus rapide du contenu des fichiers. **Fig.2.**

- sass
  - blocks
  - components
  - elements
  - forms
  - nodes
  - pages
  - regions

Il y a plusieurs options spécifiques au thème qui permettent de personnaliser le fil d'ariane, d'activer ou non les feuilles de style pour Internet Explorer, ou encore rafraîchir les données du registre de thème. **Fig.4.**

Autre point important, Basic peut être utilisé avec Grunt. Les modules Grunt inclus permettent de compiler les fichiers SASS dès qu'il y a des modifications, de convertir les images en sprites, de minifier les fichiers de code. **Fig.3.**

Conclusion

Basic est l'outil idéal pour le développeur/intégrateur de sites sous Drupal. C'est un thème simple à prendre en main utilisant des processus complexe à mettre en place de façon séparée. Il permet d'être performant et organisé pour une possible future réutilisation. Il est possible de télécharger le thème à cette adresse : https://www.drupal.org/project/basic

La version Drupal 8 est en cours de développement.

Jérémy **Allèbe** – @feelandclic

Bourbon et Neat, késako ?

Bourbon est une librairie basée sur SASS permettant de décupler les fonctionnalités de celui-ci en un minimum de temps. Cette librairie comprend une multitude de fonctions facilitant la vie de l'intégrateur. Cela peut aller de la création d'animations/transitions CSS3 pour tous les préfixes de moteur de rendu, à l'extension des valeurs prédéfinies pour le type de mouvement de ces animations/transitions, en passant par la fonction HiDPI Media Query qui permet de définir le style d'éléments seulement pour les périphériques à haute densité de pixels. Cette librairie est très puissante et à la pointe des spécifications.

Par opposition, on peut comparer Bourbon à Compass qui est similaire en terme de fonctionnalités (3).

Neat est un Framework conçu aux côtés de Bourbon qui permet de gérer une page web sous forme d'une grille contenant un certain nombre de colonnes. Il est donc très facile de gérer le nombre de colonnes, ainsi que leurs tailles et leur emplacement.

Bourbon

Fig.2

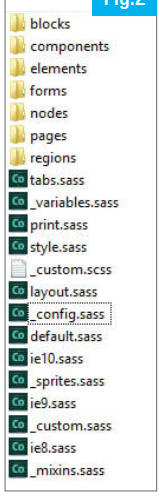


Fig.3

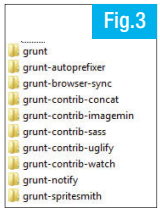


Fig.1

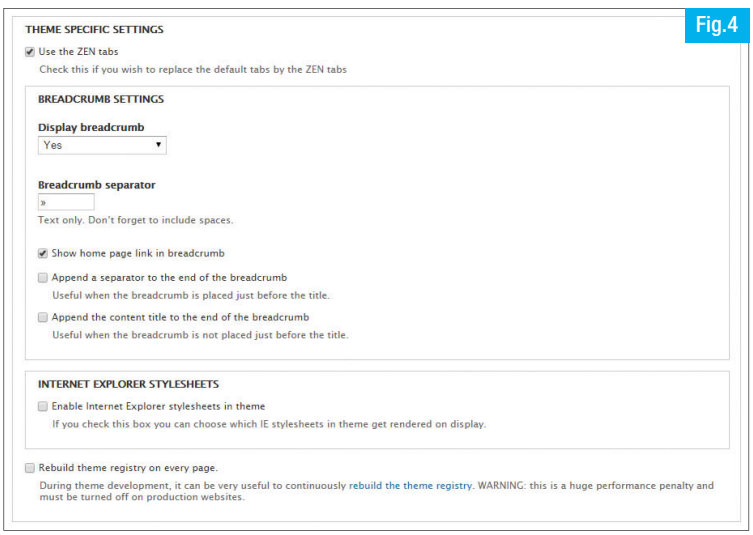
```
// Bourbon grid display. Comment out to remove display.
$visual-grid: true
$visual-grid-color: #eaeaea

// Set to false if you'd like to remove the responsiveness.
$responsive: true

// Setup the column, grid, and gutter amounts.
$column: 90px
$gutter: 30px
$grid-columns: 12
// We set the max width of the page using the px to em function in Bourbon.
// the first value is the pixel value of the width and the second is the base font size of your theme.
// In this instance we want the max width to be 1088px and on Line 6 of style.sass we have reset the base
// font size to 13px so the first value would be 1088 and the second 13
// ex. em(1088, 13)
$max-width: em(1200, 13)

// Define your breakpoints.
$mobile: new-breakpoint(max-width 480px 4)
$tablet: new-breakpoint(max-width 768px 8)
```

Fig.4



(1) https://www.drupal.org/project/basic
(2) Pour plus d'informations concernant SASS, consultez la documentation sur le site officiel : http://goo.gl/9O1KCl
(3) Pour plus d'informations sur utiliser Basic avec Compass : http://goo.gl/AAd1tO

Usines à sites

Une usine à sites est un catalogue de fonctionnalités, de configurations, de contenus ou de layouts. Ce catalogue présente une facilité de réplication et de personnalisation très appréciables pour de multiples projets. En offrant la possibilité de recycler un catalogue de développements déjà testés, validés et éprouvés, l'usine à sites permet de :

- s'appuyer sur un socle de développements qui a déjà fait ses preuves
- concentrer son effort sur les tâches de personnalisation.

Gain de temps dans la réalisation et la maintenance du projet, coût général et qualité optimisés : le sujet mérite donc un petit détour.

Quelles architectures possibles ?

On peut distinguer trois types d'architectures :

Sites totalement indépendants

Chaque instance dispose de son propre code et de sa propre base de données.
Ici l'objectif est de retrouver son catalogue de fonctionnalités lors de la création de chaque nouvelle instance (slideshows, headers et footers particuliers, utilisateurs spécifiques et permissions associées, etc.)

Sites à dépendance partielle

Chaque instance dispose d'une base de données propre mais le code est partagé par toutes les instances. Dans ce cas particulier il est également possible que certaines tables soient partagées entre instances. Le code étant partagé, l'ensemble des fonctionnalités standardisées est disponible partout mais les bases de données séparées permettent de conserver une gestion distincte en termes de contenus, d'utilisateurs et de configurations.

Sites totalement dépendants

Le code et la base de données sont intégralement partagés entre toutes les instances.
Une instance unique pour tout gérer.
Les distinctions graphiques, de contenus ou de configurations sont faites en définissant des contextes spécifiques dans la configuration globale.

Comment choisir la bonne approche ?

Il n'y a pas de « recette miracle » pour répliquer facilement un ensemble de fonctionnalités ou de configurations récurrentes. Le tableau ci-dessous récapitule les points importants permettant d'orienter un projet plutôt vers l'une ou l'autre des architectures.
En outre, il existe différentes solutions, en fonction du type d'architecture du projet.

FONCTIONNALITÉS	INDÉPENDANCE DU SITE : INDÉPENDANT	PARTIELLEMENT INDÉPENDANT	DÉPENDANT
Partage du code	✗	✓	✓
Partage de la base de données	✗	✗	✓
Facilité de création d'un nouveau « site »	✗	✗	✓
Facilité de maintenance	✗	✓	✓
SSO « par défaut »	✗	✗	✓
Partage des utilisateurs « par défaut »	✗	✗	✓
Partage des contenus « par défaut »	✗	✗	✓
Partage des configurations « par défaut »	✗	✗	✓
Backoffice partagé	✗	✗	✓
Indépendance du code et des déploiements	✓	✗	✗
Indépendance de l'hébergement	✓	✗	✗
Pas de « single point of failure »	✓	✗	✗

Quelles solutions Drupal ?

L'une des particularités de Drupal est certainement d'être le CMS le plus polyvalent sur le marché. Les solutions décrites ici ne sont pas les seules existantes pour réaliser des usines à sites sous Drupal, pour autant, elles offrent l'avantage d'être les plus populaires... et de fait, sont assez bien documentées.

Sites totalement indépendants

Drush make file

La création d'un fichier .make et son exécution via Drush (http://drush.ws/) permettent de créer une instance Drupal personnalisée. Grâce à ce fichier on peut spécifier la version du core Drupal à installer, la liste et la révision des modules contribs ou custom à installer, ainsi que les patchs à appliquer ou des éléments de configurations divers. A l'exécution d'un fichier .make Drush téléchargera l'ensemble des éléments nécessaires et en fera l'installation. Un exemple de fichier est disponible ici :
https://www.drupal.org/node/1432374

Installation profiles

Drupal offre une possibilité similaire directement intégrée au core. Un profil d'installation Drupal requiert de créer un fichier de profil et d'inclure l'ensemble des modules, thèmes et autres éléments auxquels il fait référence (un profil ne permet pas de télécharger automatiquement ces éléments). Il permet également de réaliser des configurations, des créations de contenus ou d'utilisateurs directement après l'activation de l'ensemble des composants embarqués. Pour plus d'informations :
https://www.drupal.org/node/1022020

Sites à dépendance partielle

Installation multisites

L'un des atouts de Drupal réside dans la possibilité de réaliser des instances dites multisites. Ce type d'instances propose d'utiliser un code unique pour plusieurs sites disposant chacun de leur propre base de données. Le mode opératoire est alors très simple : il suffit de mettre en place un fichier settings spécifique par domaine (voir https://groups.drupal.org/node/121989 pour plus de détails).
Toutefois, à défaut d'une parfaite maîtrise de l'approche choisie, ce type d'installation peut vite devenir très délicat. Certaines pratiques sont à proscrire ou du moins à limiter fortement. Le partage de tables, par exemple, ou la connexion à une autre base de données (via db_set_active()) peuvent introduire des effets de bord avec le cache ou avec certains hooks customs. C'est un outil puissant mais la prudence est de mise lors de son utilisation.

Sites totalement dépendants

Organic Groups

Organic Groups est un module qui permet de définir des groupes de permissions limitant l'accès à certains contenus aux membres de ces groupes. De nombreux modules viennent enrichir cette fonctionnalité première en permettant de définir des configurations spécifiques à certains groupes. Il devient alors possible de réaliser plusieurs sites au sein d'une même instance Drupal d'une façon, certes différente de celle que propose Domain Access, mais également très efficace. Cette approche permet de créer de nouveaux « sites » plus rapidement qu'avec Domain Access mais la différenciation des « sites » par domaine n'est pas possible. Plus de détails ici : https://www.drupal.org/node/861418

Domain Access

Domain Access est un module qui permet de définir des contextes basés sur les domaines d'un site sur une unique instance Drupal. Chaque domaine pointant sur cette instance peut être défini comme un site à part entière ayant ses propres contenus, son propre thème, des configurations différenciées, etc.
Pour plus d'informations :
https://www.drupal.org/documentation/modules/domain

Nicolas **LOYE** - *Chef de projet technique chez Actency. Développeur drupal depuis 2006. Nicoloye sur drupal.org.*

Le responsive et Drupal

Depuis l'avènement des Smartphones et tablettes, notre façon de développer des sites Web a changé. En effet, il y a cinq ans, on créait un site pour une seule résolution d'écran; il nous faut maintenant penser notre site Web pour les nouvelles résolutions qu'intègrent tous ces nouveaux périphériques.

C'est en partant de ce constat qu'Ethan Marcotte a introduit le terme de "Responsive Web Design" ou "Site Web Adaptatif" en Français. On simplifiera par RWD pour cet article.
Le RWD est une technique qui permet la modification de la structure graphique du site en fonction du média qui consulte ce même site. Ceci est possible grâce à la propriété @media de CSS3, la mise en place d'une grille fluide ainsi que d'autres procédés il est possible de changer l'aspect du site, un site sur trois colonnes passe sur deux sur une tablette et enfin une sur un smartphone.
Le principe du RWD est d'offrir une expérience utilisateur (UX) optimale quel que soit le support sur lequel le site est consulté.
Il existe deux façons d'aborder le RWD; la première, où l'on commence par la mise en place de la version Desktop du site Web que l'on dégrade, ceci pour terminer avec la version smartphone simplifiée. Le problème de cette pratique réside dans le fait de charger sur des téléphones les images et autres fichiers parfois très lourds constituant le site. L'arrivée de la 4G dans certaines zones soulage néanmoins ce problème. La seconde pratique appelé "Mobile First" et abordée dans le livre du même nom par Luke Wroblewski; il part de la mise en place du style de notre page pour un affichage sur Smartphone pour enfin terminer par la version Desktop. Les personnes travaillant sur la version 8 de Drupal se sont appuyées sur cette philosophie pour le style graphique du système d'administration.

Les thèmes "Responsive"

Plusieurs thèmes de base sont téléchargeables sur Drupal.org. Parmi eux, deux ont retenu mon attention lors de la mise en place de projets responsive. Ces deux thèmes sont *zen* et *AdaptiveTheme*.

Le thème Zen

Le thème Zen est un thème très léger où les bases d'une intégration responsive sont déjà mises en place (système de colonnes adaptables, image en max-width: 100% et height: auto etc...)
Le thème dans sa dernière version (7.x-5.x) est écrit en HTML5. Le HTML5 shiv (script permettant aux versions d'Internet Explorer inférieures ou égales à 8 de reconnaître et d'interpréter les balises HTML5) est inclus si besoin. Le layout par défaut est mobile first, avec un système de grille fluide. Le thème intègre respond.js, un script qui permet de prendre en charge les médias queries simples (min/max-width) sur Internet Explorer 8. Un normalize.css est aussi inclus, système de reset des styles par défaut du navigateur plus performant que le reset.css. Le normalize harmonise les styles de la page tandis que le reset remet tout à zéro. Le système intègre les classes conditionnelles pour Internet Explorer (mise en place d'une classe sur la balise <html> en fonction de la version d'IE qui consulte le site).
Zen est écrit en Sass, un préprocesseur CSS très performant. L'équipe de développeur de zen utilise les meilleures pratiques Sass/Drupal acquises au fur et à mesure du développement du thème. Une simple variable $legacy-support-for-ie7 passée à true permet le support d'IE7 et IE8.
Entrons un moment dans le cœur du thème. Pour installer le thème, décompresser le zip disponible à l'adresse : https://www.Drupal.org/project/zen dans le dossier sites/all/themes de votre site Drupal. Dans le dossier zen se trouve un dossier STARTERKIT qu'il faut que vous copiiez à la racine de vos thèmes sites/all/themes. Renommez ce dossier par le nom que vous allez donner au thème du site que vous développez. Dans le dossier de votre nouveau thème se trouve un fichier STARTERKIT.info.txt, supprimez l'extension .txt de ce fichier et renommez-le par le nom que vous avez donné au dossier du thème, ces noms doivent être les mêmes, attention à la casse. Editez ce fichier nom-de-votre-theme.info, c'est ici que vous mettrez toutes les informations concernant votre thème, nom, description, inclusion des feuilles de style et des fichiers javascript ainsi que définition des régions du thème.
Nous venons de créer un sous thème de Zen. Si vous avez besoin d'éditer les fichiers "tpl" du thème, copiez les fichiers appartenant au thème d'origine, dans le dossier templates et collez-les dans le dossier templates de votre nouveau thème, vous pourrez maintenant les modifier à votre guise.
Rendez vous maintenant à l'URL http://nom-de-votre-site.com/admin/appearance et activez votre nouveau thème. Par défaut le layout responsive est actif sur votre sous-thème. Si vous souhaitez utiliser une taille fixe pour votre site, il suffit de remplacer la ligne 13 du fichier css/style.css @import "layouts/responsive.css"; par @import "layouts/fixed.css";
Le thème utilise aussi zen grid, un ensemble de mixins Sass permettant la création d'un système de grilles fluides complexes très simplement. Attardons-nous un peu sur le fichier Sass/layouts/_responsive.scss, c'est ce fichier qui est appelé à la ligne 13 du fichier style.css. Utilisant la méthode mobile first le fichier est découpé en trois parties, le style de base est sur une colonne, remarquez les variables $zen-column-count: 1; et $zen-gutter-width: 20px; aux lignes 13 et 14, en fait ici on déclare le nombre total de colonnes de notre layout à 1 et l'espace entre nos colonnes (gouttière) à 20px. Ce style est appliqué pour les appareils avec une largeur d'écran inférieure à 480px, on le voit ligne 59 avec la règle @media all and (min-width: 480px), toutes les déclarations css comprises dans ce bloc s'appliqueront pour des largeurs d'écran supérieures à 480px. On est donc ici sur une logique de créer un layout pour les appareils ayant la plus petite largeur d'écran et de changer la mise en page grâce aux médias queries pour des appareils avec des largeurs supérieures, et grâce à respond.js nous n'aurons pas de soucis sur Internet Explorer version 8 et inférieur. À la ligne 48 la mixin @include zen-grid-item-base(); permet d'appliquer les paramètres par défaut à tous les conteneurs principaux de notre site. Ligne 55 @include zen-grid-container(); permet de stipuler que ces wrappeurs (conteneurs) utiliseront le système de grille de zen. Ligne 78 nous entrons dans la média query concernant les largeurs d'écrans comprises entre 480 et 959px, on constate que la variable $zen-column-count est redéfinie à 3, ce qui signifie que pour les appareils ayant une largeur d'écran correspondante, le nombre de colonnes de notre layout sera de trois. À partir d'ici la mixin @include zen-grid-item(); est beaucoup employée, cette mixin prend 2 paramètres obligatoires. Le premier correspond à la largeur qu'occupera l'élément concerné par la déclaration css; cette valeur est exprimé en nombre de colonne. 2 signifie que l'élément occupera l'espace de deux colonnes plus une taille de gouttière. Le second paramètre correspond à la colonne de départ de l'élément, @include zen-grid-item(2, 2); avec $zen-column-count: 3; déclaré pour toute la media query, correspond à un élément positionné à partir de la seconde colonne et qui occupe

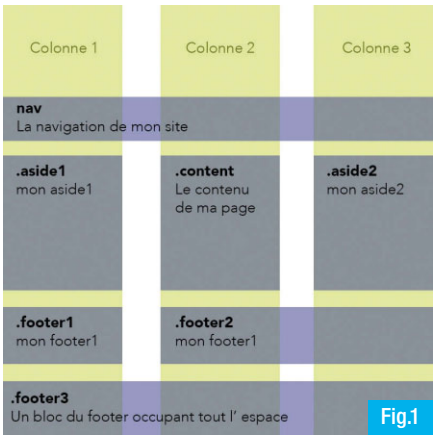


Fig.1

l'espace de deux. La mixin @include zen-clear(); permet d'arrêter le système de colonne et de repasser à la ligne.
Voici un petit schéma qui devrait rendre tout ça plus clair :

- Considérons la structure html suivante : Fig.1.
- Et le fichier Sass lié : Fig.2.
- Le navigateur affichera le résultat suivant : Fig.3.

Grâce au système de grille de Zen plus besoin de perdre du temps à calculer les tailles de nos éléments en pourcentage par rapport à leurs parents, on laisse faire Sass combiné à zen grid. Passons maintenant au second thème.

Le thème AdaptiveTheme

Ici nous avons affaire à un thème plus lourd que Zen mais néanmoins tout aussi appréciable. Pour installer le thème nous procéderons de la même manière que pour Zen. Dans ce thème la variable $is_mobile est accessible dans tous les fichiers de templates, elle permet d'afficher ou de cacher des éléments du site si le média consultant le site est un smartphone. AdaptiveTheme est configurable depuis l'interface d'admin de Drupal, rendez-vous à l'adresse http://nom-de-votre-site.com/admin/appearance/settings/nom-de-votre-theme.
On remarque que AdaptiveTheme propose beaucoup plus d'outils de configuration que Zen. Sur cette page se trouvent plusieurs onglets sur la gauche. Parcourons-les un à un. Le premier,

Fig.2
```
    <nav>
        La navigation de mon site
    </nav>
</header>

<main>
    <article class="content">
        Le contenu de ma page
    </article>
    <aside class="aside1">
        mon aside 1
    </aside>
    <aside class="aside2">
        mon aside 2
    </aside>
</main>

<footer>
    <div class="footer1">
        Le footer 1
    </div>
    <div class="footer2">
        Le footer 2
    </div>
    <div class="footer3">
        Un bloc du footer occupant tout l' espace
    </div>
</footer>
```

Standard Layout, concerne la version desktop de votre site. Ici vous pouvez choisir la taille et la position de vos différentes sidebars, la largeur de votre page, sa largeur maximale (max-width) et établir la media query associée à ce layout. Le second et le troisième onglet, Tablet Layout et SmallTouch Layout permettent de faire les mêmes réglages pour Tablette et Smartphone. Le quatrième onglet, Panels et Gpanels permet aux utilisateurs des modules Panels, Display Suite ou le module natif d'AdaptiveTheme de faire des mises en page complexes en quelques clics. Il suffit de choisir le layout, le nombre de colonnes et de cliquer sur la disposition souhaitée. L'onglet Global Settings quant à lui permet de choisir si l'on souhaite intégrer la méthode Mobile First ou Desktop First, on peut aussi y désactiver le Layout Responsive de son thème. L'onglet file management permet de changer le dossier vers lequel sont uploadés les fichiers de votre site; il met aussi à disposition un outil qui permet de concaténer tous les fichiers css et js de votre site pour n'en former qu'un. L'outil est en version beta et pose problème avec les version d'IE inférieures ou égales à la 8. L'onglet css permet d'utiliser le fichier responsive.custom.css pour y mettre ses propres medias queries. L'onglet Polyfills permet d'activer des polyfills principalement pour les problèmes liés à Internet Explorer, il inclut respond.js, le HTML5shiv et d'autres choses plus spécifiques. L'onglet Metatags permet de définir certaines balises meta de votre site. L'onglet Debuggers permet de mettre en valeur les régions de son thème, d'afficher la taille de la fenêtre et de gérer le debug lors de l'utilisation des layouts Panels et Gpanel. Cet onglet permet aussi de visualiser toutes les médias queries mises en place avec le thème.
Il est aussi possible avec AdaptiveTheme de charger des scripts ou des feuilles de style conditionnelles pour Internet Explorer, il suffit dans le fichier nom-de-votre-theme.info d'insérer par exemple la ligne ie_stylesheets[screen][IE 8][] = css/ie8.css pour charger une feuille de style seulement pour Internet Explorer 8.
AdaptiveTheme est un thème qui plaira à toutes les personnes qui ne désirent pas trop mettre les mains dans le code. En effet, il est possible de créer des mises en page très compliquées avec Gpanels. Nous allons maintenant voir quelques techniques RWD.

Fig.3
```
@media all and (min-width: 480px) and (max-width: 959px) {

  $zen-column-count: 3; // déclaration de 3 colonnes
  $zen-gutter-width:  20px; // 20 pixels de largeur pour les goutières

  header,
  main,
  footer { // ces wrappeurs contiendrons un système de grille zen
    @include zen-grid-item-base();
    @include zen-grid-container();
  }

  header {
    @include zen-grid-item(3, 1); // occupe tous l' espace ( 3 colonnes )
  }

  .aside1,
  .footer1 {
    @include zen-grid-item(1, 1); //occupe la place de la première colonne
  }
  .aside3 {
    @include zen-grid-item(1, 3); //occupe la place de la dernière colonne
  }
  .content {
    @include zen-grid-item(1, 2); //occupe la place de la colonne du milieu
  }
  .footer2 {
    @include zen-grid-item(2, 2); // occupe la place de deux colonnes en partant de la seconde
  }
  .footer3 {
    @include zen-grid-item(3, 1); // occupe tous l' espace ( 3 colonnes )
    @include zen-clear(); // Reviens à la ligne
  }
}
```

En bref...

Nous venons de faire un petit tour d'horizon de deux thèmes particulièrement intéressants pour le RWD sur Drupal. J'aimerais pour clore cet article partager avec vous quelques articles très intéressant concernant le RWD. Le premier “Responsive Webdesign – présent et futur de l’adaptation mobile” a été écrit sur alsacreation.com par Stéphanie Walter, on peut y comprendre toutes les problématiques liées au RWD. Le second est un article très intéressant sur les bienfaits qu'apporte une intégration entièrement en em, toutes les valeurs chiffrées de sa feuille de style en em ... L'article a été écrit par Marie Guillaumet et son titre est : "Refonte de mon portfolio : du responsive tout en em". L'article est en deux partie, un peu long mais grandement enrichissant. Il devrait vous convaincre d'intégrer vos sites de cette façon...

Bon responsive à tous.

Jérémie **de Cuyper** @feelandclic

Domain Access – How to

Un projet multisite peut être envisagé selon deux écoles :

- Partir d'une solution compartimentée avec "partage du code source", chaque site a sa propre base de données,
- Opter pour une solution centralisée « un seul code source et une seule base de données ». Chaque site est alors traité comme une entité particulière.

Le module « Domain Access » relève de cette seconde école. Même si la procédure d'installation est plus délicate, il s'installe et se configure de la même manière que tous les autres modules, sur une installation de Drupal déjà existante.

Installation et configuration du module

Domain Access est disponible sur drupal.org avec ses instructions d'installation : https://www.drupal.org/node/1068570

Il suffit donc de commencer par le télécharger dans le répertoire sites/all/modules/contrib/ de notre installation Drupal et de l'activer. On ne peut que recommander d'utiliser Drush qui s'en chargera pour nous en deux commandes :

```
> drush dl domain
   > drush en domain
```

Nos contenus (nodes) déjà existants, ainsi que nos utilisateurs, seront alors automatiquement assignés au domaine par défaut ainsi qu'à tous ses sous-domaines lors de cette première étape.

Vous pouvez néanmoins contourner ce comportement par défaut en assignant la valeur `FALSE` aux constantes `DOMAIN_INSTALL_RULE`, `DOMAIN_SITE_GRANT` et `DOMAIN_ASSIGN_USERS` issues du fichier sites/all/modules/contrib/domain.module.

Avant de pouvoir commencer à accéder à la configuration du module sans que Drupal ne retourne d'erreur, il est nécessaire d'ajouter la ligne suivante à la fin de notre fichier settings (sites/default/settings.php) :

```
include DRUPAL_ROOT . '/sites/all/modules/contrib/domain/
settings.inc';
```

Nous retrouvons l'accès à notre backoffice et Drupal qui nous propose alors de reconstruire les « permissions d'accès », ceci afin de recalculer l'accès aux contenus éventuellement déjà présents sur le site avant l'installation du module

Gestion des domaines et comportements par défaut

Nous pouvons maintenant créer de nouveaux domaines (c'est-à-dire de nouveaux « sites » à l'intérieur de notre configuration multisite) depuis l'interface : admin/structure/domain. Il suffit pour cela de cliquer sur le bouton « Create domain » et de renseigner.

L'url de notre site dans « Domain » (cela peut-être un sous-domaine ou un autre domaine **mais pas un sous-répertoire** du type dev.drupal7/site2)

- Un nom pour le domaine
- Le protocole (http ou https)

Pour accéder à chaque nouveau domaine, les DNS du serveur devront accepter des multiples entrées, et notre VirtualHost Apache doit être configuré en conséquence, comme dans l'exemple ci-dessous :

```
<VirtualHost * :80>
DocumentRoot /chemin/vers/installation/Drupal
ServerName dev.drupal7
ServerAlias *.dev.drupal7
ServerAlias site2.com
</VirtualHost>
```

Il faudra mettre à jour notre fichier hosts en local (/etc/hosts pour linux ou OS X et C:\Windows\System32\drivers\etc\host pour windows) :

```
127.0.0.1 dev.drupal7
127.0.0.1 site2.dev.drupal7
127.0.0.1  site2.com
```

A partir de là nous pouvons déjà créer de nouveaux contenus ou blocs, et les affecter à un ou plusieurs de nos domaines via les nouvelles options de configuration qui apparaissent désormais dans ces formulaires de création et d'édition. Le système de gestion d'accès aux contenus de Drupal est pensé pour surcharger toutes les autres fonctionnalités du site. De fait, les menus et les listings (avec Views) sont directement impactés et n'affichent que les contenus disponibles sur le domaine courant, sans pour autant nécessiter de configuration supplémentaire. Ainsi, un utilisateur ayant la permission d'« Outrepasser le contrôle d'accès au contenu » passe outre la restriction d'accès, et accède donc à tous les contenus sur tous les domaines.

Configurations spécifiques par domaine et extensions possibles

Bien souvent, il s'agit de pouvoir modifier certains éléments de configuration uniquement sur un domaine précis, comme tout simplement l'URL de la page d'accueil, l'email de contact, etc.

Pour cette raison, le module embarque tout un ensemble de sous-modules tels que Domaine Settings, Domain Theme et bien d'autres, qui vont nous permettre de **paramétrer des configurations différentes selon les besoins** pour chacun de nos domaines.

Et pour 99% des autres types de configuration, comme par exemple pour Organic Groups, Views ou encore Variable et bien d'autres, il est possible de choisir dans la liste des modules de la communauté qui étendent les fonctionnalités de Domain Access, disponible elle aussi sur drupal.org : https://www.drupal.org/node/1068570.

Bilan et limitations du module

Alors que les atouts du module sont indéniables, il reste quelques points d'amélioration, certes peu nombreux. Par exemple lorsque, par défaut, on accède successivement à l'un ou à l'autre de nos domaines, la session n'est pas persistante. Alors qu'il existe une solution simple pour les sous-domaines (qui consiste à modifier la valeur de `$cookie_domain` dans le fichier sites/default/settings.php en lui assignant le nom de domaine partagé, soit `$cookie_domain = '.dev.drupal7'` dans notre cas), pour les autres domaines (comme « site2.com ») il faut par contre envisager des solutions SSO plus avancées.

Pour conclure, le succès du module Domain Access ne se dément pas. Certes, il continue de faire ses preuves depuis sa sortie (qui, rappelons-le, remonte à l'époque de Drupal 5 !.) Mais surtout, la solide bibliothèque de modules additionnels reliés qui s'est progressivement constituée couvre aujourd'hui largement l'ensemble des problématiques qui peuvent se poser pour ces installations « multisite ».

Pour l'école du multi-site qui se base sur « un seul code source et une seule base de données », alors Domain Access tire son épingle du jeu. L'alternative solide se situe du côté d'Organic Group.

Ce sont alors les contraintes du projet qui vont dicter le choix entre ces deux solutions. Un sujet qui a déjà suscité d'intéressants échanges au sein de la communauté Drupal...

Leo **PRADA**
Chef de projet technique chez Actency

Manuel **GROSRENAUD**
Développeur Web depuis 2007 et chef de projet technique chez Actency.

ZF2 Discovery et le modèle MVC

1ère partie

Conçu pour résoudre la plupart du temps de façon standard des problématiques de complexité d'architecture et de dépendances, Zend Framework utilise intensivement les designs patterns au sein de ses classes. Parmi ses nombreux motifs, l'un d'entre eux est devenu populaire ces dernières années, celui du « Modèle - Vue - Contrôleur ». S'étant de facto imposé comme un standard de structure applicative, il permet de séparer les couches logiques d'abstraction, de présentation et de contrôle.

ZF2, dans sa deuxième version, aujourd'hui en 2.3.2, a vu sa copie complètement revue concernant son implémentation PHP du paradigme MVC. Pour cela, les core-developers se sont basés sur le principe de l'inversion de contrôle (IoC) : en premier lieu ils ont déstructuré la boucle de distribution bien connue sous ZF1 pour être complexe et se sont assuré de donner moins de responsabilité aux contrôleurs.
Ce patron IoC définit "que le flot d'exécution d'un logiciel n'est plus sous le contrôle direct de l'application elle-même mais du framework ou de la couche logicielle sous-jacente." (Wikipedia). Martin Fowler s'inspirant de ce principe a créé le terme "d'injection de dépendances" (Di) qui permet de résoudre les problématiques de couplage en créant dynamiquement les dépendances entre classes grâce à des fichiers de configuration ou déterminées à l'exécution du code.
Une implémentation de ce mécanisme existe dans ZF2 via le motif de conception "ServiceLocator" (Zend\ServiceManager\ServiceManager). Alors que Zend\Di\Di injecte les dépendances en dehors de la classe, le ServiceManager permet de récupérer les dépendances de cette classe depuis l'intérieur de celle-ci. C'est grâce à ce dernier que ZF2 permet une combinaison de souplesse, de robustesse et de scalabilité.

Travaux pratiques

Nous allons traverser et appréhender les différents aspects de ce que contient fréquemment une application web de nos jours, à savoir des pages basées sur des URLs, des listes d'entités qu'on peut créer et modifier par un formulaire, et une base de données pour la persistance de ces entités. Nous nous baserons sur l'application d'exemple proposée par l'équipe de Zend Framework, "Zend Skeleton Application" (ZSA). Nous réutiliserons le thème de Twitter Bootstrap, ce qui nous permettra de nous départir de toute complexité de présentation. Les requêtes seront toutes en HTTP. A savoir qu'un bon exercice à la suite de cet article serait de développer les mêmes fonctionnalités avec de l'AJAX... ;).
Comme beaucoup de projets open-source actuels, différentes options d'installation s'offrent à vous pour récupérer le projet ZSA qui nous servira de base de travail. Télécharger une archive ZIP, cloner le dépôt Git, ou bien utiliser Composer. Nous allons utiliser Composer, car il offre la possibilité d'installer automatiquement les dépendances du projet, que nous nommerons « zf2app » :

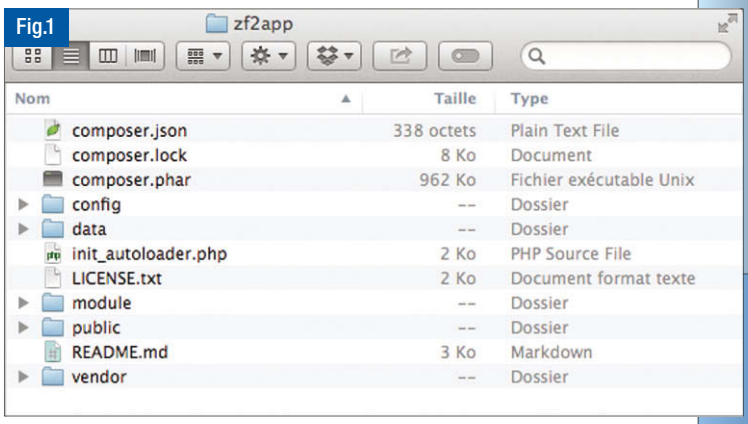
```
curl -s https://getcomposer.org/installer | php --
php composer.phar create-project -sdev --repository-url=
"https://packages.zendframework.com" zendframework/skeleton
-application zf2app
```

Si vous disposez de PHP 5.4 ou supérieur, placez-vous dans le dossier zf2app puis saisissez :

```
php -S 0.0.0.0:8080 -t public/ public/index.php
```

Cette commande lancera le serveur embarqué de PHP sur le port 8080 et y connectera toutes les interfaces réseaux. Vous avez également la possibilité de définir un hôte virtuel Apache comme suit : (pensez à faire pointer votre IP locale sur zf2app.local) Fig.1.

Fig.1




```
<VirtualHost *:80>
    ServerName zf2app.local
    DocumentRoot /path/to/zf2app/public
    <Directory /path/to/zf2app/public>
        DirectoryIndex index.php
        AllowOverride All
        Order allow,deny
        Allow from all
    </Directory>
</VirtualHost>
```

Faisons un rapide tour des dossiers :

- config : contient la configuration applicative dans sa globalité, ainsi que les fichiers de configuration dépendant de l'environnement.
- data : contient les données temporaires, caches fichiers, et autres persistances diverses relatifs au développement.
- module : contient le cœur métier de l'application. Chaque module est un sous-dossier de ce dossier.
- public : contient généralement le bootstrap de l'application, ainsi que les assets (images, feuilles de styles, javascript...)
- vendor : contient tous les modules tierce-partie. Ce dossier est généralement exclu du versioning.

Naviguer avec une vue claire sur la route

Pour commencer à développer notre application d'exemple de gestion de statuts, la première étape va consister à implémenter les pages dédiées. Pour cela, trois étapes sont nécessaires : créer le contrôleur qui sera en charge de gérer les requêtes, ajouter la configuration de routage pour les URLs et créer les vues pour afficher les pages.
Le contrôleur trouve sa place dans *module/Application/src/Application/Controller*. Créez une classe nommée StatusController. Celle-ci héritera de

Zend\Mvc\Controller\AbstractActionController, qui est la classe de base des contrôleurs dans ZF2. L'application possède deux pages, nous allons donc créer deux actions dédiées.
Ajoutez deux méthodes publiques à la classe *StatusController*, à savoir *indexAction* et *updateAction*. La première servira la liste des statuts ainsi que leur création, la deuxième la mise à jour d'un statut.
L'association de vues à ces actions se trouve dans le fichier de configuration du module (module.config.php) dans l'entrée du ViewManager.
Le rendu des scripts de vue est géré par *Zend\View\Renderer\PhpRenderer*. Celui-ci accepte tout type de langage de template et fonctionne avec des « resolvers », qui sont chargés de transcrire un nom de template en une ressource que le « renderer » devra charger. ZF2 fournit deux « resolvers » : le TemplateMap et le TemplatePathStack, chargés dans cet ordre (grâce à *Zend\View\Resolver\AggregateResolver*). Dans le TemplateMap, vous définissez les vues qui seront chargées le plus souvent, celles qui ne peuvent pas être chargées automatiquement par correspondance de noms/chemins, ou encore celles dont vous voulez changer cet automatisme.
Dans le TemplatePathStack, vous définissez dans quels chemins parcourir l'arborescence pour trouver une correspondance.
Si une correspondance est trouvée dans un des « resolvers », les suivants ne seront pas chargés.
En l'occurrence pour notre application, nous n'avons aucune modification à apporter à la configuration du ViewManager : l'entrée `template_path_stack` nous indique que les vues, pour être chargées, devront se trouver dans le dossier module/Application/view et, par défaut, suivre la hiérarchie view/:module/:controller/:action.phtml.
Créez un dossier *module/Application/view/application/status* ainsi que les fichiers *index.phtml* et *update.phtml* au sein de celui-ci. Insérez un texte en HTML dans chacun de ces fichiers.
Pour rendre les contrôleurs accessibles et donc afficher les vues via une requête HTTP dans un navigateur, nous devons faire correspondre ce couple contrôleur/action à une URL. C'est le rôle du routeur. Il existe différents types de routeur HTTP au sein de ZF2, dans le namespace *Zend\Mvc\Router\Http* : Hostname, Literal, Method, Part, Regex, Scheme, Segment, Query. Chacun d'entre eux offre une méthode spécifique pour capturer l'URL demandée et en extraire un contrôleur, une action, et des paramètres supplémentaires. Pour notre application, nous utiliserons le routeur HTTP « Segment » qui permet comme son nom l'indique de segmenter les différentes parties d'une URI avec un séparateur spécifique. Les informations minimales à renseigner sont la clé de la route dans le routeur, son type, ainsi que le pattern de la route dans les options. Dans le fichier module/Application/config/module.config.php, supprimez l'entrée de tableau [`'router'`][`'routes'`][`'application'`] et ajoutez [`'router'`][`'routes'`][`'status'`] comme suit :

```
'routes' => array(
 'status' => array(
   'type' => 'Zend\Mvc\Router\Http\Segment',
   'options' => array(
     'route' => '/status[/:action][/:id]',
   )
 ),
),
```

Les règles seront :
- La liste des statuts sera accessible via /status
- La modification d'un statut sera accessible via /status/update/<ID>.

L'action étant une variable, cette configuration permet d'en créer des supplémentaires comme "view", "download", etc... En ajoutant des actions/vues au contrôleur, sans modifier le fichier de configuration.
Nous pouvons "aliaser" le type par son nom de classe, ainsi que définir des valeurs par défaut (soit pour des valeurs non définies, soit pour des valeurs paramétrables) ainsi que des contraintes pour ces valeurs. A noter que la valeur du contrôleur, qui est un service, doit correspondre à la clé définie dans le ControllerManager ([`'controllers'`][`'invokables'`]).
Complétez la configuration du routeur comme suit :

```
'routes' => array(
 'status' => array(
   'type' => 'Segment',
   'options' => array(
     'route' => '/status[/:action][/:id]',
     'defaults' => array(
       'controller' => 'Application\Controller\Status,
       'action' => 'index',
     ),
     'constraints' => array(
       'action' => '[a-zA-Z][a-zA-Z0-9_-]*',
       'id' => '[0-9]*',
     ),
   ),
 ),
),
```

Naviguez sur http://localhost:8080/status. Votre message HTML du fichier index.phtml s'affiche ! :)
Pour pouvoir passer des valeurs à la vue depuis son action dans le contrôleur, plusieurs possibilités s'offrent à vous :
- Retourner un « simple » tableau associatif avec ces valeurs et leurs clés.
- Retourner une instance de Zend\View\Model\ViewModel pour une configuration objet plus avancée de cette vue.

Testez en ajoutant à StatusController::indexAction la ligne suivante :

```
public function indexAction()
{
 return array('message', 'Placeholder for status list');
}
```

Et dans view/application/status/index.phtml :

```
<?php echo $this->message; ?>
```

Ou bien :

```
<?php echo $message; ?>
```

Naviguez sur http://localhost:8080/status. Votre message envoyé depuis le contrôleur s'affiche ! :)
[Navigation (factory, container, template)]

Prendre exemple sur le modèle

Maintenant que nos pages sont accessibles, il nous faut leur faire afficher dans un premier temps la liste des statuts qui proviendront, pour notre exemple, d'une base de données locale SQLite. A noter que le code de cet exemple peut fonctionner avec tout type de moteur de base de données (MySQL, PgSQL, SQL Server, etc...). Pour créer la source, enregistrez ce script depuis un éditeur de texte sous zf2app/data/db/status.sql :

```
BEGIN TRANSACTION;
CREATE TABLE status (id INTEGER PRIMARY KEY, user TEXT, message
 TEXT, created_at TEXT, updated_at TEXT);
INSERT INTO status VALUES(1,'jguittard','My name is Julien',
```

```
'2014-04-02 11:30:22','2014-04-21 01:28:28');
INSERT INTO status VALUES(2,'johndoe','I am unknown','2014-04
-15 16:12:44',NULL);
COMMIT;
```

Puis dans un terminal, placez-vous dans zf2app/data/db et saisissez :

```
sqlite3 status.db < status.sql
```

Premier élément à configurer : l'adaptateur à la base de données, de type *Zend\Db\Adapter\Adapter*. Cela se fait en 2 éléments de configuration : les informations de la source, et le service qui gèrera l'accès aux données. Dans le fichier module.config.php, ajoutez les entrées suivantes :

```
'service_manager' => array(
  […]
  'factories' => array(
    'Zend\Db\Adapter\Adapter' => 'Zend\Db\Adapter\AdapterService
Factory',
  ),
),
[…]
'db' => array(
  'driver' => 'Pdo',
  'dsn' => 'sqlite:' . getcwd() . '/data/db/status.db',
),
```

Afin de pouvoir interagir avec cette source de données, ZF2 nous laisse le choix de l'implémentation de ce qu'on appelle la « couche modèle » (« data layer ») et fournit un composant relatif au motif de conception « Table Data Gateway Pattern ». Ce motif stipule « qu'un objet agit comme une passerelle à une table de base de données. Une instance gère toutes les lignes de cette table ». (« *Patterns of Enterprise Application Architecture* », Martin Fowler : je le recommande :))
Concrètement, chaque table de notre base sera représentée par une classe héritant de *Zend\Db\TableGateway\AbstractTableGateway*.
Créez une classe nommée StatusTable héritant de *Zend\Db\TableGateway\AbstractTableGateway* dans src/Application/Model. En regardant le code de cette classe abstraite, nous remarquons qu'elle nécessite une instance de l'adaptateur que nous avons précédemment créée pour exécuter les requêtes en base.
Pour satisfaire ce besoin, nous allons utiliser l'injection de dépendances au travers du motif de conception « Factory »
Créez une classe nommée StatusTableFactory implémentant *Zend\ServiceManager\FactoryInterface*.
La création du service se déroule comme suit :

```
public function createService(ServiceLocatorInterface $serviceLocator)
{
  $dbAdapter = $serviceLocator->get('Zend\Db\Adapter\Adapter');
  return new StatusTable($dbAdapter);
}
```

Il nous faut maintenant déclarer cette création de service dans la configuration du module :

```
'service_manager' => array(
  […]
  'factories' => array(
    'Zend\Db\Adapter\Adapter' => 'Zend\Db\Adapter\AdapterService
Factory',
     'Application\Model\StatusTable' => 'Application\Model\
StatusTableFactory'
  ),
),
```

[=> Explanations]
Enfin, notre classe *StatusTable* doit pouvoir accepter cet adaptateur injecté dans son constructeur. Le mode de fonctionnement est celui-ci : pour chaque entrée du tableau « factories », le service manager va tester si la valeur de chaque clé est soit une classe implémentant FactoryInterface et donc appeler la méthode « createService », soit un type « callable » et donc l'exécuter. Dans les deux cas, une instance du service manager est injectée dans ces deux méthodes et un objet est attendu en retour. Une exception Exception\ServiceNotCreatedException est générée en cas d'échec.

```
public function __construct(Adapter $adapter)
{
  $this->adapter = $adapter;
}
```

Ajoutez également un membre définissant le nom de la table en base, ainsi qu'une méthode permettant de requêter la liste de tous les statuts, avec des paramètres de sélection optionnels :

```
public function fetchAll($params = array())
{
  $resultSet = $this->select($params);
  return $resultSet;
}
```

Dès lors, nous pouvons depuis nôtre contrôleur, et ce grâce au ServiceManager, obtenir une instance de notre service StatusTable et appeler sa méthode fetchAll() :

```
public function indexAction()
{
  $service  =  $this->getServiceLocator()->get('Application\
Model\StatusTable');
  var_dump(get_class($service));
  $statuses = $service->fetchAll();
  var_dump(get_class($statuses));
  var_dump($statuses->toArray());
  exit;
}
```

Notre service est bien une instance de *Application\Model\StatusTable* et le retour de sa méthode fetchAll() une instance de *Zend\Db\ResultSet\ResultSet*. Cette classe représente le type de retour par défaut en utilisant *Zend\Db\TableGateway\AbstractTableGateway*.
Elle implémente une méthode toArray() qui permet d'obtenir un tableau associatif des résultats.
Afin d'avoir une meilleur gestion ainsi qu'un contrôle plus structuré et sécurisé de nos données, nous allons « modéliser » celles-ci. Chaque entrée sera une représentation objet d'une ligne à lire, insérer ou modifier.
Pour cela, créez un dossier src/Application/Entity et créez-y une classe Status, avec des propriétés protégées correspondant aux colonnes de la table « status » (id, user, message, created_at, updated_at) ainsi que les accesseurs correspondants (getters & setters).
Pour rappel, ces derniers permettent de conserver l'encapsulation de la classe en n'exposant pas les propriétés de façon publique. Une alternative serait d'exposer ces propriétés privées via ArraySerializableInterface.

Julien **Guittard**
Architecte / Formateur PHP/Zend - @julienguittard

Un site PHP Symfony 2 dans Azure Web Sites

Parmi les différentes façons d'héberger un site Web PHP dans Microsoft Azure, Azure Web Sites est une offre plateforme de type PaaS permettant de disposer très rapidement d'un environnement Cloud hautement disponible et élastique pour gérer son site Web (PHP, ASP.NET, Node.js, Java et Python pour le moment).

Parmi les avantages d'Azure Web Sites, nous citerons :

- Elasticité montante et descendante automatisée
- Une balance de charge déjà intégrée (Load Balancer/LB)
- Diverses méthodes de publication supportées: FTP/FTPS/WebDeploy
- Publication via des outils de contrôle de code source tel que Git et Github, CodePlex, Bitbucket, Mercurial, TFS, VSO etc...
- Une disponibilité quasi immédiate de nouvelles instances (quelques dizaines de secondes seulement pour associer une nouvelle instance au LB de votre site web)
- Support de certificats SSL
- Intégration simplifiée avec tous les autres services d'Azure (Blob Storage, Media Services ...)

Pour de plus amples informations sur Microsoft Azure Web Sites, rendez-vous sur l'espace dédié du portail Microsoft à l'adresse http://azure.microsoft.com/fr-fr/services/web-sites/.

Comment se lancer ?

La création d'un site Azure Web Site peut être faite très rapidement et de 3 manières différentes.

1 - Via le portail d'administration de Microsoft Azure (http://manage.windowsazure.com),
   - Une fois identifié dans le portail, vous pouvez créer votre site tel qu'affiché ci-dessous, Fig.1.

2 - En utilisant les outils pour ligne de commande pour Mac / Linux (Azure CLI),
   - Installation Azure CLI,
   - Téléchargement de vos paramètres de publication,
   - Appel de la commande suivante :

```
azure site create MySiteName
```

Votre site ainsi créé aura une URL qui sera http://MySiteName.azurewebsites.net (plus de détails sur cette méthode dans la suite de l'article).

3 - Via l'utilisation des Commandlets Azure PowerShell,
   - Installation des Commandlets Azure Powershell,
   - Téléchargement de vos paramètres de publication,
   - Appel de la commande suivante dans un invite de commande Powershell :

```
New-AzureWebSite MySiteName
```

On retrouve le même nom de site qu'avec l'Azure CLI ou le portail, à savoir http://MySiteName.azurewebsites.net

Fig.1

Une fois que vous avez créé votre site dans Azure Web Sites, vous pouvez dès lors bénéficier des nombreux avantages d'Azure Web Sites mentionnés en introduction (plus d'informations sur comment se lancer dans Azure Web Sites via le portail Microsoft : http://azure.microsoft.com/fr-fr/documentation/services/web-sites/)

Azure Web Sites pour quelles technos ?

.Net, Java, Python, NodeJS, PHP et d'autres à venir... Azure Web Sites s'adapte à votre projet et votre framework. Focalisons-nous sur l'écosystème PHP.

La majorité des Frameworks et CMS PHP est disponible nativement dans la plateforme via la galerie d'Azure Web Sites. Il n'y a d'ailleurs pas que des templates/CMS PHP, on y retrouvera également du Django, etc. Il est ainsi par exemple possible d'installer en seulement « 3 clics » sa propre version de Drupal, Wordpress, Joomla!, CakePHP, myWiki.

(NDOP redondance)Cependant, dans un cas un peu plus complexe, il convient parfois de pouvoir installer des extensions tierces, voire ses propres extensions, de personnaliser la configuration PHP (tant d'un point de vue version utilisée – PHP5.5 disponible à l'écriture de cet article – que concernant la configuration de chaque extension), voire même d'exécuter un ensemble de tâches pour initialiser votre projet (tâche Grunt par exemple). Voyons ceci au travers d'un exemple de déploiement d'une application Symfony2 sur Microsoft Azure Web Sites.

Symfony2 dans Azure Web Sites

Nous allons nous intéresser dans cette partie au déploiement d'une application Symfony2 standard dans Microsoft Azure Web Sites au travers du système de déploiement Git lié à un repository local (un branchement via un repository Github est également nativement supporté par la plateforme). Pour cela, nous reverrons très succinctement comment initialiser un projet Symfony2 et ses dépendances avant de nous intéresser à la création et la configuration d'un nouveau Web Site. Nous traiterons ensuite du déploiement de votre application sur Azure Web Sites et des conseils de personnalisation de votre application pour la rendre plus performante compte tenu de l'infrastructure de Microsoft Azure Web Sites.

Prérequis : nous supposons que vous disposez, sur votre environnement, de PHP et composer disponibles dans votre variable PATH ainsi que de NodeJS (facultatif pour les utilisateurs Windows).

Installation de l'édition Standard de Symfony2

Pour installer l'édition standard de Symfony2, on s'en tient à l'utilisation de la ligne de commande et de l'outil composer :

```
$ composer create-project symfony/framework-standard-edition
myWebSites "2.5.*"
```

Nous reviendrons sur sa configuration dans la suite de cet article.

Initialisation et configuration d'Azure Web Sites

Pour créer un environnement Web Sites nous allons ici utiliser l'outil cross-platform Microsoft azure-cli (outil NodeJS). Pour son installation et

sa configuration, nous vous invitons à vous intéresser au lien suivant : http://azure.microsoft.com/fr-fr/documentation/articles/xplat-cli/
Rendez-vous ensuite dans le répertoire de votre projet Symfony et exécutez la commande suivante :

```
azure site create myWebSites --location "West Europe" --git
 --gitusername sescandell
```

Le résultat de cette opération sera sensiblement identique à celui-ci :

```
[09:23 AM]-[vagrant@packer-virtualbox-iso]-[/tmp/programmez]
$ azure site create programmez --location "West Europe" --git --gitusername sescandell
info:    Executing command site create
+ Getting sites
+ Getting locations
info:    Creating a new web site at programmez.azurewebsites.net
info:    Created website at programmez.azurewebsites.net
+
info:    Executing `git init`
info:    Initializing remote Azure repository
+ Updating site information
info:    Remote azure repository initialized
+ Getting site information
info:    Executing `git remote add azure https://sescandell@programmez.scm.azurewebsites.net/programmez.git`
info:    A new remote, 'azure', has been added to your local git repository
info:    Use git locally to make changes to your site, commit, and then use 'git push azure master' to deploy to Azure
info:    site create command OK
```

On note notamment trois points importants :

1 - Le site programmez.azurewebsites.net vient d'être créé. L'URL du site est directement liée au nom donné à notre Web Site. Il doit évidemment être unique sur l'ensemble de la plateforme Microsoft Azure Web Sites,
2 - Nous venons là, via l'option `--git` et `--gitusername` d'initialiser un repository Git pour le déploiement de notre application,
3 - La commande a automatiquement ajoutée sous forme de remote, une URL de publication. Un `git remote show` en local vous montre que ce remote se nomme par défaut "`azure`".

Il nous reste alors maintenant à configurer notre Web Site pour pouvoir fonctionner correctement avec Symfony2. Pour son bon fonctionnement, ce framework requiert, entre autres, la présence de la librairie php_intl, conseille l'activation d'accélérateur de code (type APC) et nécessite la définition de la variable de configuration date.timezone.
Concernant la librairie php_intl, elle fait maintenant partie de la configuration de PHP par défaut. Aucune action n'est donc requise.
Pour la gestion d'accélération de code, nous allons utiliser le couple OpCode PHP5.5 et Wincache (les instances Web Sites sont des serveurs Windows avec serveur Web IIS. Il est donc préférable d'utiliser Wincache à APC). Pour activer PHP5.5, une ligne de commande est suffisante :

```
azure site set --php-version 5.5 programmez
```

Concernant Wincache, il est par défaut activé et configuré pour se limiter au cache de fichiers PHP et cache applicatif utilisateur. Aucune action complémentaire n'est donc requise.
Reste enfin le cas de la configuration de la variable date.timezone. Pour pouvoir personnaliser cette configuration, nous allons avoir recours aux fichiers .user.ini. En effet, sous environnement IIS, PHP est exécuté en mode FastCGI, il nous est donc possible de personnaliser son contexte d'exécution via un fichier de configuration utilisateur (plus d'informations sur le sujet sur le site php.net : http://php.net/manual/fr/configuration.file.per-user.php). Nous rajoutons donc dans le dossier web/ un fichier .user.ini avec le contenu suivant :

```
date.timezone=Europe/Paris
```

On ajoute ce fichier à notre VCS et on en a terminé avec la configuration de l'environnement d'exécution de PHP. Avant de déployer, cependant, il nous reste deux actions à effectuer. L'une d'entre elle est facultative, fortement conseillée tout de même.
La première consiste à définir le fichier web.config pour rediriger toutes les requêtes vers le fichier app.php de Symfony.
La seconde, facultative, est de configurer le point d'entrée de IIS de notre application sur le répertoire web/ et non à la racine de notre projet (pour des questions de sécurité). Cette opération nécessite une intervention sur le portail Microsoft Azure. Il suffit de se rendre dans la partie "Configuration" du Web Site que l'on vient de créer et de modifier, dans la zone "Applications et annuaires virtuels", l'entrée "/" pour pointer vers le dossier "`site\wwwroot\web`" (au lieu de "`site\wwwroot`" uniquement).
Reste alors à ajouter dans votre dossier web un fichier web.config (l'équivalent du fichier .htaccess pour IIS) avec pour contenu :

```
<!-- web.config -->
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <system.webServer>
    <rewrite>
      <rules>
        <clear />
        <rule name="Serve static files" stopProcessing
="true">
          <match url="^(.*)(\.css|\.js|\.jpg|\.png|
\.gif)$" />
          <conditions logicalGrouping="MatchAll" trac
kAllCaptures="false">
          </conditions>
          <action type="None" />
        </rule>
        <rule name="Symfony Frontcontroller" stop
Processing="true">
          <match url="^(.*)$" />
          <action type="Rewrite" url="app.php" />
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
</configuration>
```

Déploiement du projet

Pour déployer un projet sur Azure Web Sites un simple `git push azure master` est suffisant. Cependant, dans le cadre d'un projet Symfony2, nous avons besoin, lors du déploiement, d'installer/mettre à jour nos dépendances et éventuellement d'initialiser le cache. Microsoft Azure Web Sites intègre le projet Kudu pour pouvoir contrôler vos déploiements et notamment effectuer des actions personnalisées lors de la mise à jour de votre repository (base sur un hook git).
Commençons donc par Initialiser le projet avec le script de déploiement par défaut pour un projet PHP via la commande :

```
azure site deployment -php
```

Nous modifions alors le fichier généré pour lui ajouter, juste avant la partie "Deployment", le code suivant :

```
<<CodeBash>>
```

```
export SYMFONY_ENV=prod
cd "$DEPLOYMENT_SOURCE"
# Run composer install only if there is a difference between the
# new composer.lock and the previous one
 diff composer.lock "$DEPLOYMENT_TARGET/composer.lock" >> /dev/null
 if [ ! $? -eq 0 ]; then
   # Check composer is there
   if [ ! -f composer.phar ];
   then
     echo "Downloading Composer"
     curl -sS https://getcomposer.org/installer | php -- --quiet
     exitWithMessageOnError "Composer download failed"
   else
     echo "Updating Composer"
     php composer.phar self-update
     exitWithMessageOnError "Composer self-update failed"
   fi
   php composer.phar install --prefer-dist -v --no-interaction
 --optimize-autoloader
   exitWithMessageOnError "Composer install failed"
 else
   echo No need to run composer install
   php app/console cache:clear
   php app/console assets:install web
 fi
 php app/console assetic:dump
<</CodeBash>>
```

Nous ajoutons alors ce fichier à notre VCS, et nous sommes fin prêts à déployer l'édition Standard Symfony2 sur Microsoft Azure Web Sites. Pour cela, une seule ligne de commande à exécuter :

```
git push azure master
```

Et c'est terminé. Notre site est alors disponible à l'adresse http://programmez.azurewebsites.net.

Améliorations et Bundle Symfony

La méthode de déploiement que nous venons de voir reste très manuelle et très basique. Pour des performances accrues et une automatisation du processus de déploiement, avoir les bons outils devient une nécessité.

Par exemple, compte tenu de l'architecture d'un environnement Azure Web Sites (figure ci-dessous), déplacer le répertoire de cache devient une nécessité Fig.2.

Pour cela, la société Brainsonic a repris l'initiative sur un Bundle facilitant le déploiement sur Azure et l'utilisation des services Azure. Disponible en open source sur la plateforme Github (https://github.com/brainsonic/AzureDistributionBundle/) ce bundle, en développement actif, permet entre autres :

- L'accès, sous forme de Services au sens Symfony, aux services Azure (Service Bus, Service Storage Blob/Table/Queue, ...),
- La gestion de vos déploiements pour Web Sites (avec automatisation des tâches décrites précédemment) et aussi pour les environnements Cloud Services,
- La personnalisation de l'environnement d'exécution de votre projet pour correctement définir les chemins de cache (amélioration des performances), ou la centralisation des logs par exemple,
- La pré-configuration de vos environnements pour une communication avec SQL Server (nécessitant une manipulation particulière sur Web Sites),
- Un suivi des évolutions des deux environnements (Symfony et Azure, deux mondes en éternelle évolution et changement).

Conclusion

Comme nous venons de le voir, déployer une application sur Azure Web Sites est une tâche relativement aisée une fois l'environnement et les concepts Azure compris.

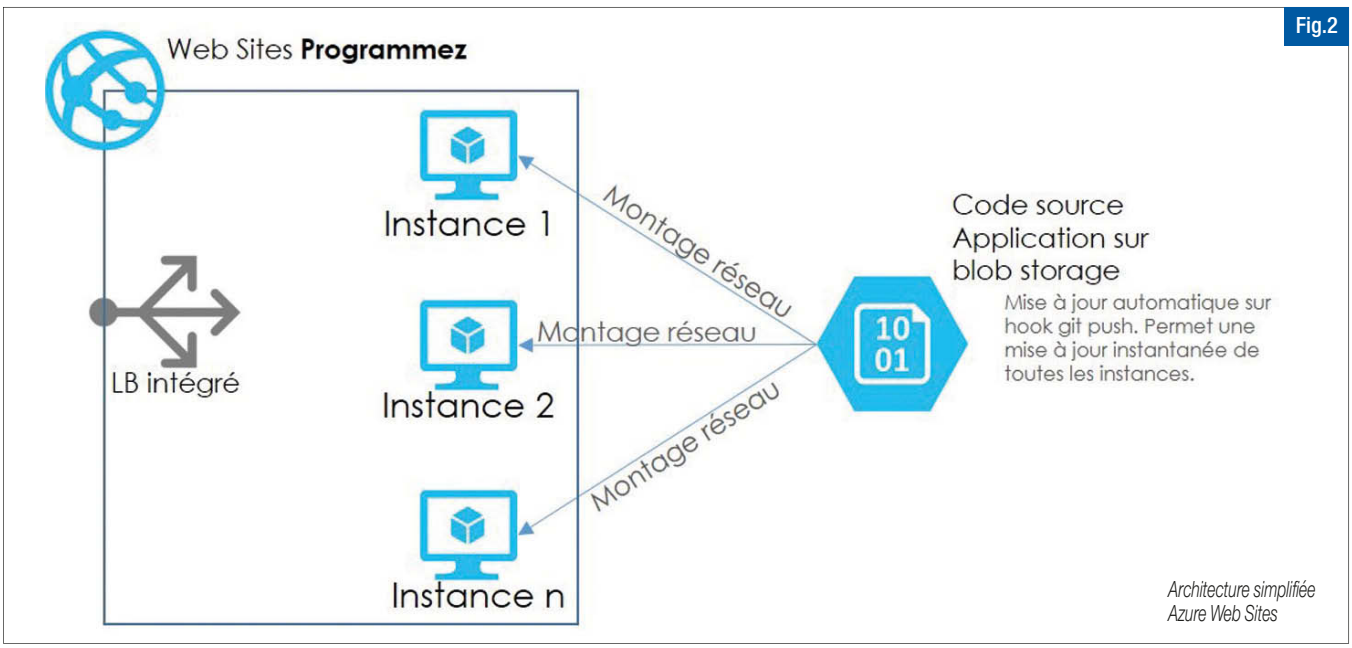
Dans le cadre d'une application Symfony, nous avons vu comment simplement déployer une application existante sur le Cloud et immédiatement tirer profit de sa scalabilité et disponibilité sans changement.

Pour pouvoir ensuite entièrement tirer parti des bénéfices du Cloud tels que l'auto-scalabilité d'instances, la mise à disposition des resources etc., des adaptations peuvent être nécessaires et l'utilisation d'outils tels que le bundle brainsonic/AzureDistributionBundle peuvent grandement aider à cela.

Stéphane **Escandell**
Chef de projet Technique @ Brainsonic

Benjamin **Moulès**
Technical Evangelist Microsoft Azure

Fig.2



Architecture simplifiée Azure Web Sites

Les conteneurs Docker

Docker par ci, Docker par là ... impossible de passer à côté. Docker fait un tel buzz qu'on ne peut pas l'ignorer, même si un certain scepticisme reste de rigueur : nous avons été habitués à ces outils révolutionnaires qui devaient changer le visage du développement logiciel, et qui sont venus s'ajouter à notre boîte à outils, voire ont tout simplement disparu dans l'anonymat.

Docker part d'un constat : développer du logiciel repose sur un flux complexe et hétérogène entre le développeur et la plateforme de production, jalonné par diverses étapes de validation : développement, packaging, tests, QA, performances, pré-production...
Avec toute la bonne volonté du monde, passer du poste du développeur à l'environnement de production ne se fait pas sans douleur, ni sans mauvaises surprises. Comment anticiper un problème de clustering quand on développe sur un poste de développement isolé ?
Comment éviter les difficultés de portage entre un environnement de développement sous OSX ou Windows et une machine Red Hat Enterprise Linux ?
L'idée de Docker, c'est de rendre ce chemin le plus linéaire possible pour le logiciel, à travers un format binaire parfaitement portable. D'autres s'y sont déjà essayés, avec des succès variables, à commencer par Java « write once, run anywhere » - sauf que les applications modernes ne sont plus monolithiques – mono-instance, et nécessitent pour être testées de déployer N services sur N environnements. Accessoirement, la portabilité de Java n'est pas parfaite. Qui n'a pas rencontré un problème d'encodage UTF-8 ? Par ailleurs elle ne sont pas réduites au seul serveur d'application Java : quid de la base de données, des middlewares divers et autres load-balancer ?
Certains ont recours à la virtualisation, utilisant Vagrant pour démarrer sur le poste de développement un environnement iso-production basé sur des machines virtuelles. Cette approche a un intérêt évident pour fiabiliser la chaîne de développement, mais n'offre qu'une réponse partielle, et surtout coûteuse. Si vous avez déjà lancé plus de trois machines virtuelles sur votre poste, vous comprenez très bien le problème. Le temps de démarrage et la consommation de ressources sont tels que la fluidité du développement en souffre, et que cette solution reste cantonnée à des cas spécifiques ou des problématiques de diagnostic.

Et voilà Docker

Que propose Docker pour dépasser ce semi-échec ? Si elles n'étaient pas aussi lourde à mettre en œuvre – consommation alarmante de ressources, taille des images disque – les machines virtuelles seraient une approche idéale pour avoir un packaging complet et transportable depuis le poste de développement jusqu'à la production.
Si on décortique l'approche machine virtuelle, on constate qu'on empile un nombre de couches alarmant : sur la machine, son OS, et ses diverses librairies, on fait tourner un hyperviseur (typiquement Virtualbox) qui va émuler une plateforme matérielle sur laquelle on fait tourner un OS, ses librairies, le runtime java, notre serveur d'application, notre application, tout ça pour afficher « hello world ».
De toute évidence, on paye au prix fort l'exécution de notre application dans un environnement sous contrôle.
Si on veut retenir une image de cette situation, on est en train de faire porter à notre machine une machine aussi lourde qu'elle, tout ça pour avoir une petite cabine de seconde classe avec hublot.
Le contrat que Docker va vous faire signer est très simple : votre environnement cible est un Linux – pas n'importe quelle distribution, mais un Linux. Si vous développez des applications pour Windows ou AIX passez votre chemin. Vous êtes encore là ? Très bien.

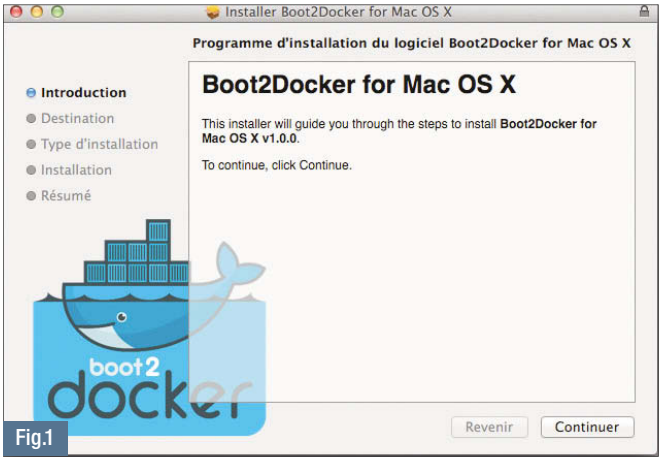


Fig.1

Hey mais attendez, vous venez de dire que les machines virtuelles c'est le mal absolu et que Docker leur tordait le coup !

Docker utilise une technologie existante, issue d'une décennie d'évolution du noyau linux, et qui permet de lancer ce qu'on appelle des « conteneurs ». Un hyperviseur traditionnel du type Virtualbox est capable d'émuler le matériel d'une machine complète, ce qui permet de faire tourner un émulateur Atari ST. Cependant peu d'entre nous on ce besoin. L'idée de Docker est de ne pas chercher à émuler toute une machine, mais à utiliser le noyau Linux de l'hôte, pour « juste » remplacer le système par un autre. Cela ressemble à un *chroot*, mais les conteneurs vont bien au delà.
Un conteneur est un groupe de processus Linux qui fonctionnent en complète isolation avec le reste du système. Ils ne voient pas les autres processus, ne peuvent communiquer avec eux (à moins qu'on fasse explicitement le nécessaire pour les y autoriser), ne peuvent dépasser la consommation de ressources qu'on leur a allouée. Certains utilisent déjà depuis des années la technologie associée (namespace, cgroups), mais elle reste très technique, et, surtout, non disponible sur le poste du développeur lambda. Au lieu de tout un OS virtuel, on ne fait donc tourner que quelques processus, et on ne peut donc faire tourner de très nombreux containers. Pour reprendre le parallèle maritime, notre bateau va pouvoir porter de nombreux containers de contenu varié, sans risque de surcharge.
Docker a décidé de prendre cette technologie qui arrive aujourd'hui à maturité, et de la simplifier au travers d'une interface utilisable même par un développeur Java Junior – c'est tout dire – pour qu'elle serve de passerelle entre tous les environnements.
L'autre secret de fabrique de Docker, c'est le système de fichiers qu'il utilise. Lorsqu'on va démarrer un conteneur, celui-ci va utiliser une image qui définit le système de fichiers qui sera vu par l'application. Ce ne sera pas celui de la machine hôte, mais un système de fichiers stocké sous forme binaire compacte. Il devient alors possible de lancer une application qui tournera sur un système CentOS depuis une Ubuntu. Par ailleurs, le système de fichiers de l'image est en lecture seule, et une couche *copy-on-write* permet d'enregistrer les modifications apportées. Seules ces différences devront être échangées avec un autre environne-

ment pour reproduire fidèlement le contexte de l'application. En résumé, tous les avantages de la machine virtuelle, mais sans le surcoût.

En pratique

Linux uniquement ? Mais moi je bosse sur Mac, c'est tellement plus confortable (#troll) – ou bien sous Windows, parce que c'est la règle dans ma société. Pas de panique.
Nous allons installer Docker, ici sur un Mac mais l'approche est la même sous Windows. Commencez par télécharger le paquet d'installation sur https://docs.docker.io/installation/mac. Docker fournit un installeur complet qui va se charger de tout configurer pour nous. Celui-ci va nous installer le client Docker, VirtualBox et une machine virtuelle Boot2Docker Fig.1.

Docker ne peut tourner que sur un Linux, aussi l'équipe Docker utilise une machine virtuelle Linux microscopique (25Mo) « boot2docker » avec le juste nécessaire pour faire tourner Docker. Cette VM ultra légère ne comporte aucun processus coûteux, et sera lancée une seule fois pour tous les conteneurs dont nous aurons besoin. Le client Docker lui est natif OSX/Windows et communique avec cette machine virtuelle, aussi tout est transparent pour le développeur.
Après l'installation, une application boot2docker est disponible.
Au premier lancement, elle va se charger de configurer notre machine virtuelle. Depuis la ligne de commande qu'elle a démarré, vous pouvez passer les commandes Docker et interagir avec vos conteneurs.
Pour vous simplifier la vie, je vous conseille aussi de récupérer l'IP de votre machine virtuelle via la commande `boot2docker ip` et de lui donner un petit nom dans votre `/etc/host`. Dans mon cas j'y accède sous le nom « docker ».

Jouons un peu

Le premier conteneur que nous allons lancer utilisera une Ubuntu – lancée depuis OSX. Pour disposer d'une image Ubuntu sur laquelle baser notre premier conteneur, lancez la commande :

```
docker pull ubuntu
```

Docker va télécharger l'image Ubuntu depuis DockerHub, le dépôt des images Docker, l'équivalent de Maven Central. Cette image sera conservée par la VM boot2docker et utilisée pour tous nos autres conteneurs Ubuntu. Pour cette première fois il faut donc prendre son mal en patience... Fig.2.
Vous noterez au passage le concept de *layers*, dont nous reparlerons plus loin. Une fois l'image Ubuntu disponible, nous allons lancer un premier conteneur et y exécuter un shell bash en more interactif

```
docker run -ti ubuntu /bin/bash
```

Et hop, nous voilà dans une Ubuntu. Vous pouvez vérifier, allez faire un tour sur le système de fichiers, c'est bien celui d'une Ubuntu. Certes, un `uname -a` montrera que le noyau Linux n'est pas exactement celui que vous attendiez - c'est celui de boot2docker ; n'oubliez pas, nous ne sommes qu'un processus, isolé dans un conteneur, qui a accès à un système de fichiers Ubuntu, mais est au final un processus du système hôte.
Vous pouvez quitter le shell et relancer un conteneur, la commande est quasi instantanée et la consommation de ressource de votre machine n'a pas bougé. Comparez-ça au lancement d'une machine virtuelle Ubuntu !
Vous pouvez même faire des choses plus étonnantes. Depuis notre shell ubuntu (vérifiez à deux fois que vous êtes bien dans votre conteneur !) vous pouvez lancer une commande généralement peu recommandée :

```
root@90feb5b33554:/# rm -rf /bin
root@90feb5b33554:/# ls
bash: ls: command not found
```

Oups. Sortons vite de ce conteneur tout cramoisi, pour en relancer un autre. Le système Ubuntu est comme neuf ! En fait il est strictement conforme à l'image Ubuntu que nous avons téléchargée, et c'est ce qui va faire de Docker un outil de choix pour notre chaîne de *continuous delivery*. L'image Docker que je construis sur mon poste (ou sur DockerHub), que je valide par la suite, et que je démarre sur mes machines de production est identique à l'octet prêt, et le sera à chaque redémarrage d'un conteneur ; un niveau de reproductibilité à faire pâlir votre responsable QA.
Comme indiqué précédemment, le système de fichiers Ubuntu est en lecture seule, et les modifications apportées sont enregistrées en mode copy-on-write dans un nouveau *layer*. Si je décide de conserver le layer (via la commande `docker commit`), je peux le partager avec un autre développeur via une nouvelle image. Si il dispose déjà de l'image Ubuntu il ne téléchargera que le layer manquant. D'où une rapidité de diffusion des images Docker qui fait vite oublier les giga-octets des images de machines virtuelles.

Construire ses images

Lancer un bash dans Ubuntu c'est bien, mais c'est un peu limité, il va falloir mettre dans nos images Docker diverses librairies, runtimes et notre code applicatif. Nous pouvons le faire avec nos petites mains, et *committer* le résultat pour que Docker conserve l'image modifiée, mais nous sommes en 2014 et ce genre de choses ne se fait plus – pensez infrastructure as code !
Docker sait construire des images à partir d'une représentation texte, une recette en quelque sorte, et si on s'y prend bien avec des informations précises, la recette produit toujours le même gâteau.

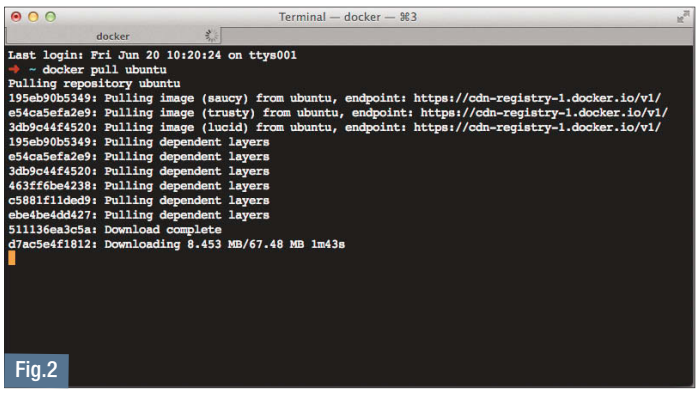


Fig.2

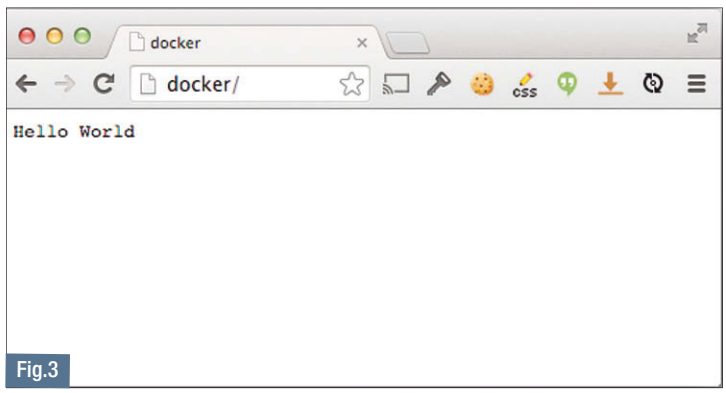


Fig.3

Pour nos premiers pas, nous allons créer une application NodeJS minimale. Créez un répertoire de travail, et créez dedans le fichier main.js :

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

var server = http.createServer(function (request, response) {
  response.writeHead(200, {"Content-Type": "text/plain"});
  response.end("Hello World\n");
});

server.listen(8000);
console.log("Server running on port 8000");
```

Créons ensuite le fichier Dockerfile qui va définir l'image Docker de notre projet. Un Dockerfile est un pur fichier texte, très facile à diffuser, dans lequel on va déclarer étape par étape la construction d'une image Docker, à partir d'une image de base.

```
FROM ubuntu:14.04

RUN apt-get update
RUN apt-get install -y nodejs

ADD . /src

EXPOSE  8000
CMD ["nodejs", "/src/main.js"]
```

Sans entrer dans le détail de la constitution d'un Dockerfile (je vous recommande de suivre l'excellent tutoriel en ligne), notre recette décrit la construction de l'image applicative étape par étape :
FROM définit l'image de base sur laquelle sera basée notre nouvelle image : une Ubuntu, plus précisément dans ce cas une 14.04 – il est recommandé de bien préciser vos versions si vous ne faites pas juste du prototypage.
RUN définit des commandes à exécuter pour compléter l'image. Ici nous installons le paquet système npm, nous pourrions passer n'importe quelle autre commande shell.
ADD permet d'inclure dans l'image Docker des fichiers issus de votre environnement de développement. Ce contenu sera copié dans le conteneur sous l'arborescence /src.
EXPOSE permet de spécifier à Docker que notre image désire communiquer avec le monde extérieur (le hublot de notre cabine), sur le port 8000.
Enfin, CMD définit le processus à démarrer dans notre conteneur si nous n'en précisons pas un lors du lancement du conteneur par docker run. Notre Dockerfile prêt, nous lançons la construction de l'image :

```
-> vi main.js
-> vi Dockerfile
-> docker build -t ndeloof/hellonode .
Sending build context to Docker daemon 3.584 kB
Sending build context to Docker daemon
Step 0 : FROM ubuntu:14.04
 ---> e54ca5efa2e9
Step 1 : RUN apt-get update
 ---> Running in 0a35730aa6f1
Ign http://archive.ubuntu.com trusty InRelease
Ign http://archive.ubuntu.com trusty-updates InRelease
(...)

---> 3e9979f31f92
Removing intermediate container 037d38f35d3a
Step 3 : ADD . /src
 ---> efe3125b818a
Removing intermediate container 45d0ee8c9173
Step 4 : EXPOSE  8000
 ---> Running in c707a0945345
 ---> 25b42abcbcbf
Removing intermediate container c707a0945345
Step 5 : CMD ["nodejs", "/src/main.js"]
 ---> Running in c57188a3458a
 ---> 405542e5b0b9
Removing intermediate container c57188a3458a
Successfully built 405542e5b0b9
```

Le dernier argument est le chemin vers le Dockerfile. Détail amusant, vous pouvez passer une URL, comme par exemple un dépôt Github, qui devient un standard de fait pour diffuser tous les hello-world de la terre. Votre image est prête.
Chaque commande définie par le Dockerfile est enregistrée comme une modification à partir de l'image Ubuntu, ces modifications s'empilant sous formes de layers successifs. Ce mécanisme (basé sur AUFS) permet à Docker de préserver l'image initiale, mais aussi de ne pas refaire ce qu'il a déjà fait. Si nous relançons la commande `docler build`, le résultat sera quasi instantané, car Docker a mémorisé dans un layer le résultat de l'exécution de la commande indiquée à partir de l'image précédente. De même, un second Dockerfile, utilisant les mêmes commandes apt-get dans le même ordre, va tout simplement reprendre l'image résultat dans son cache, jusqu'à ce qu'il ait quelque chose de nouveau à faire.
Avantage important : en tant que développeur, reconstruire votre image Docker est très rapide. Avec un framework adapté et des plugins de type livereload, le cycle de développement « je code, je sauve, je vois ce que ça donne » devient possible tout en fonctionnant dans l'environnement cible.

La ronde des containers

Une fois l'image construite, nous lançons un conteneur pour notre image Docker

```
docker run -p 80:8000 -d ndeloof/hellonode
```

Le paramètre –P nous permet de définir le port de l'hôte qui va correspondre au port 8000 réclamé par le conteneur. L'application web magnifique que nous avons construite est alors disponible sur http://docker:80/ (rappelez vous que j'ai enregistré l'IP de la VM boot2docker dans mon /etc/hosts sous ce nom) Fig.3.
Cette abstraction nous permet d'ailleurs de lancer 12 conteneurs sans conflit :

```
docker run -p 81:8000 -d ndeloof/hellonode
docker run -p 82:8000 -d ndeloof/hellonode
docker run -p 83:8000 -d ndeloof/hellonode
docker run -p 84:8000 -d ndeloof/hellonode
…
```

Du point de vue de l'hôte, il ne s'agit que de quelques processus, consommant assez peu de ressources, mais avec l'isolation des conte-

neurs ils se comportent comme si nous avions déployé notre application sur un cluster. Nous pouvons donc très facilement tester notre load-balancer – lancé lui-même dans un conteneur – l'implémentation de notre cache distribué, etc.
Docker permet de lier nos conteneurs entre eux, pour les faires collaborer ensemble et former ainsi une vraie application. Si par exemple mon application se connecte à une base de données :

```
-> docker run -d mysql
406b334d890a1f7d623109cbc8a18e7b5d3cc675b9fdae2695f9f1475f5e
-> docker run -p 80:8000 -link 406b334d890:db -d ndeloof/app
```

L'application va pouvoir accéder à une machine « db », Docker se chargeant de fournir la résolution DNS adaptée et de router le trafic réseau vers le conteneur qui héberge notre base MySQL.
Docker rend obsolète tous nos frameworks de configuration dynamique, car le runtime devient suffisamment flexible pour adresser les spécificités de l'exécution. Finis les profils, ports définis via des propriétés système, frameworks de log paramétrables dans tous les sens.
Codez tout en dur, et laissez Docker adapter le runtime à vos exigences de déploiement !

Pour le développement

Docker est donc un outil simple d'utilisation qui offre de nouvelles perspectives. En tant que développeur, comme vous l'avez vu sur ces exemples simples, déployer l'application selon une topologie en cluster typique de la production est trivial et peu coûteux en ressources. Il va alors être possible de tester systématiquement, de manière fiable mais sans surcout notre application dans son environnement cible. Plus d'excuse à la « *je comprend pas, ça marche sur mon poste* »
Une autre utilisation sur un poste de développement est une gestion largement simplifiée des middlewares. Plutôt que de perdre du temps à installer une base de données locale, lancez donc l'image Docker associée ? `docker run mongodb` et le tour est joué. Vous pouvez même intégrer à votre projet la construction d'image Docker de développement incluant la base de donnée et son initialisation avec des données de référence, pour permettre au développeur fraichement parachuté sur le projet d'être productif à t0.

Pour l'intégration continue

DockerHub se propose de construire pour vous les images à partir de vos Dockerfile, et de garantir ainsi leur origine. Ces « trusted builds » passent par un dépôt GitHub ou BitBucket et peuvent être privés (mais payants). De nombreux projets opensource proposent ainsi des images prêtes à l'emploi dont vous pouvez vérifier la recette de construction.
DockerHub propose même la notion de *certified repository*, permettant l'installation d'un middleware comme MongoDB; cela devient alors un simple Docker run mongo. Pour votre usage interne, vous pouvez confier la construction de vos images Docker à un serveur d'intégration continue comme Jenkins. Il existe déjà plusieurs plugins qui tirent partie de Docker dans ce contexte.
Le plugin *Docker build and publish* de mon collègue Michael permet par exemple de faire un équivalent du DockerHub pour vos images internes. Il propose, entre autres, de forcer l'éviction du cache Docker, si vous voulez garantir une production de l'image « propre » à partir de du Dockerfile – au prix d'un build plus lent.
Le plugin *Docker* (tout court) permet d'utiliser Docker pour définir vos esclaves de build. Il reste cependant d'un usage assez limité.
J'ai créé le plugin *Oki-Docki*, qui vise à figer votre environnement de build via un Dockerfile, plutôt que de reposer sur des outils pré-installés et/ou sur une armada de plugins Jenkins. Dans ce Dockerfile vous allez définir votre JDK, la version de Maven, bref tout ce qui est nécessaire pour construire le projet. Vous conservez ce Dockerfile avec votre code source. C'est d'ailleurs exactement le même Dockerfile que j'ai évoqué précédemment pour définir votre environnement de développement.
Le plugin détecte ce Dockerfile et lance un container pour exécuter le build, puis passe une commande (typiquement, l'exécution d'un script shell lui aussi dans votre repo). Si le Dockerfile évolue il reconstruit l'image à la volée. Le workspace Jenkins qui contient le code de votre projet est monté dans le conteneur sous `/var/workspace`. Cela permet à

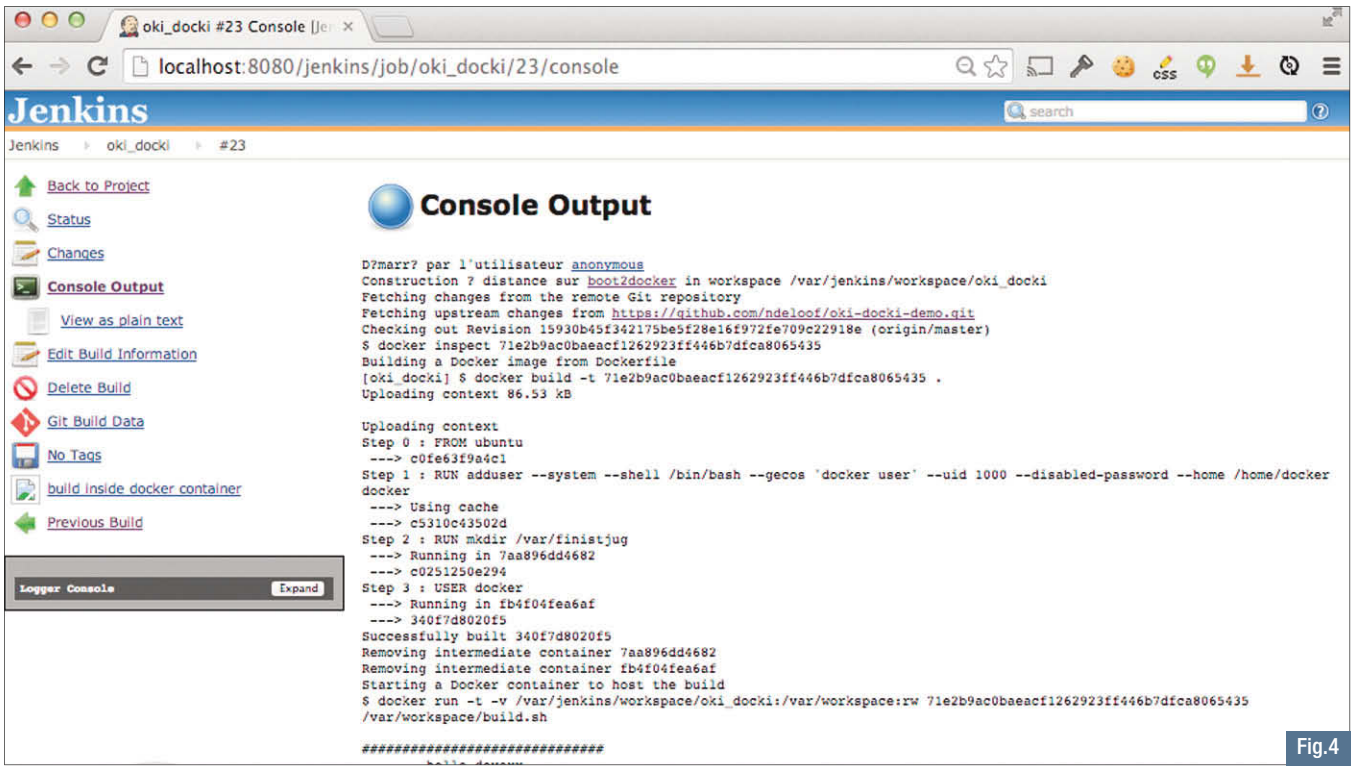


Fig.4

Jenkins d'exploiter le résultat du build pour archiver les binaires résultants ou analyser les résultats de tests Fig.4.
Groupon a développé le macro-plugin *DotCi* utilisant une approche équivalente, mais incluant l'intégration directe avec GitHub et plusieurs autres fonctionnalités spécifiques.
Bref, comme toujours, l'écosystème Jenkins voit fleurir plusieurs solutions que vous devrez tester pour voir celle qui convient le mieux à votre besoin.

Pour la QA

Exécuter des tests fonctionnels, des tests de performance, d'endurance, cela permet d'augmenter la confiance dans le livrable et est donc un élément qui doit faire partie de la chaîne de *continuous delivery*. Ici aussi, Docker va se montrer utile.
Comme nous l'avons vu, l'isolation des conteneurs permet de tester le clustering et la topologie d'une application complexe sans nécessairement disposer d'un nombre élevé de machines. Par ailleurs, la parfaite reproductibilité des images Docker va permettre de mettre en place une chaîne de validation basée sur des images Docker, prêtes à être utilisées en production.
DockerHub propose d'ailleurs un mécanisme de web hook pour ce même besoin. Une fois un changement détecté sur votre repo Github, il va certes reconstruire l'image Docker associée, mais peut aussi vous notifier pour tout process annexe (validation, redéploiements...). Le plugin Jenkins DockerHub que je développe en collaboration avec l'équipe Docker peut vous aider à mettre en place un processus complet de validation.

Pour la Production

La production, et c'est son rôle, est souvent frileuse à adopter des solutions techniques encore très jeunes. Pourtant, comme l'ont prouvé les témoignages utilisateur lors de la DockerCon en juin à San Francisco, nombreux sont ceux qui n'ont pas attendu une 1.0 officielle pour utiliser Docker. Evidemment, une production qui exploite déjà LXC sera mieux préparée à cette adoption.
Installer Docker sur les machines de production n'est pas compliqué en soit, par contre les Ops doivent disposer d'outils adaptés pour gérer cette nouvelle approche et lier les conteneurs entre eux.
L'approche conteneur encourage également à morceler les applications en composants interconnectés, et à adopter un rythme de livraison élevé. Clairement, l'outillage Ops doit évoluer en conséquence. Des briques existent déjà, et les équipes vont devoir les apprivoiser. Citons par exemple Fig (http://orchardup.github.io/fig/) qui permet de lancer un groupe de conteneurs Docker en une seule commande.
S'il ne gère pas la distribution sur un cluster, il offre une solution simple pour des applications qu'on a séparées en petit modules spécialisés. Il peut être utile également aux développeurs qui désirent lancer l'ensemble de l'appli facilement. Google a défini un format de descripteur yaml pour un groupe de conteneurs qui doivent être déployés en groupes. Utilisé par *Managed VM* sur Google Compute Engine, il est aussi exploité par Kubertenes.
Kubertenes est un projet qui, comme de nombreux autres, propose de distribuer des conteneurs Docker sur les nœuds d'un cluster – citons aussi CoreOS, et Apache dans cette catégorie. Ces outils proposent une gestion unifiée des ressources du cluster et de la distribution des conteneurs sur les noeuds. Ces solutions sont plus ou moins matures, et nécessitent un temps de prise en main pour en tirer le meilleur parti, mais apportent une souplesse de déploiement qui en vaut la peine.

La suite ?

Docker a bâti son succès non sur une révolution technologique, mais en ayant pris une technologie arrivée à maturité et en ayant défini une interface commune, adoptée largement. Les conteneurs existent depuis longtemps sur BSD, Solaris, OpenVz, mais n'ont pas su percer car ils étaient limités à ces environnements.
L'équipe Docker a donc défini une API, libContainer, sur laquelle se base Docker, qui utilise pour l'instant les capacités du noyau Linux. Cette base va permettre de contribuer à une version DBS, OpenVz ou autre, pour élargir le champ d'application de Docker. RedHat, Google, Parallels et Canonical sont très impliqués dans son développement et son portage vers d'autres technologies.
De la même façon, Docker a ouvert la possibilité de tester simplement des architectures distribuées. Cependant, ce type d'architecture commence en général en lançant plusieurs Threads dans le même processus, puis plusieurs processus, avant de considérer une distribution sur N machines. La communication entre ces acteurs change de nature à chaque étape. Docker a donc à nouveau défini une interface commune, libChan – encore en cours d'élaboration – qui permet aussi bien de faire communiquer des threads que des serveurs distants. L'abstraction du protocole déplace la complexité du côté de la gestion de l'infrastructure, et permet de venir greffer à la volée des composants d'infrastructure (monitoring, audit, routage, ...). Définir les briques d'une infrastructure distribuée est évidemment le premier problème auquel on est confronté. Si Kubernetes ou Mesos sont très sexy, que faire de l'infrastructure existante et des outils associés ? D'ailleurs pourquoi choisir CoreOS et pas l'une des nombreuses solutions comparables qui émergent aujourd'hui? A nouveau, une technologie arrive à maturité mais n'a pas de langage commun.
LibSwarm a pour objectif de fournir un socle pour permettre de définir la topologie d'une application modulaire et/ou distribuée, sans dépendre d'une implémentation, ainsi que de permettre de construire ses propres extensions pour tirer partie d'outils maison.
La visibilité sur ce projet est encore réduite, mais les premières démonstrations (http://www.youtube.com/watch?v=a_YbxWbHgQA) montrent par exemple un agrégateur; il permet sur une commande `docker ps` d'accéder à la fois au démon Docker local, à une instance hébergée sur le Cloud Digital Ocean et une autre sur Orchard. L'unification ainsi obtenue va à nouveau dans le sens d'une fluidification du processus de développement, effaçant les distinctions entre poste de développement, serveurs locaux et hébergement Cloud.
L'écosystème Docker est donc en ébullition permanente, avec toujours plus d'acteurs à venir rejoindre la partie : Google, RedHat. Pivotal, RackSpace, Canonical, Opscode, PuppetLabs ... Certains sont d'ailleurs membres du comité de pilotage du projet open-source Docker, une ouverture importante à noter sur un projet qui peut devenir rapidement stratégique, et qui ne vous liera donc pas à un unique fournisseur.
Si l'intérêt de Docker est évident dès son utilisation sur le poste de développement, son impact sur les pratiques de développement et les infrastructures n'en est encore qu'à ses débuts. A vous d'inventer les patterns qui vont émerger de cette petite révolution.

Nicolas **De Loof** - *Hacker chez CloudBees*

La création d'un projet de Voting Boards pour Devoxx France 2014

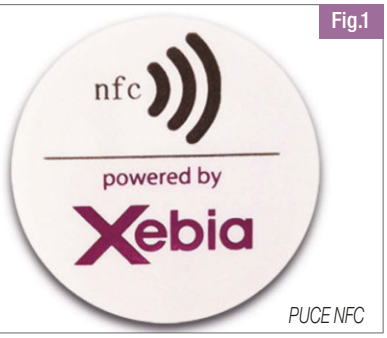
Arrêter avec l'ère du tout digital / dématérialisé pour revenir à un matériel tangible, profiter de nos expériences passées sur Raspberry Pi et réaliser un projet tous ensemble, voici les motivations qui nous ont poussé fin 2013 à essayer une autre façon de participer à Devoxx France !

L'idée qui nous est venue : réaliser des boîtiers physiques, pour donner du feedback immédiat aux speakers de Devoxx. A la manière des bornes de vote de Devoxx Anvers, nous voulions donner la possibilité aux participants de donner une note de 1 à 5 aux conférences auxquelles ils assistaient, et d'avoir un retour des résultats immédiat sur le tweetwall de la salle des congrès du Marriott Hôtel.
Cet article retrace la genèse de ce projet.

La naissance d'un projet

A l'approche de Devoxx France 2014, nous nous sommes penchés sur la frustration que pouvait avoir un speaker en n'ayant aucun retour concret concernant ses conférences. Il lui est impossible de jauger sa performance et donc de progresser. Il existe bien un formulaire postDevoxx, qui permet d'avoir un minimum de feedback mais nous voulions obtenir cette dimension du "saisi sur le vif".
De plus, deux évènements nous ont inspiré. Tout d'abord, nous avons mené un projet NodeJs sur Raspberry Pi pour lequel nous nous interconnections avec un lecteur NFC. Puis, lors de notre passage à Anvers en 2013, nous avons eu l'occasion de voir que l'architecture de ce projet était très similaire à celle de leurs boîtiers de vote NFC. L'idée était donc née : nous voulions nous occuper du système de vote pour Devoxx France !
Un restaurant avec l'équipe organisatrice plus tard, le partenariat entre Xebia et Devoxx France était sur les rails pour l'édition 2014.
Les bases fonctionnelles ont été rapidement posées :

Fig.1



PUCE NFC

- un système intuitif,
- un fonctionnement rapide, pour ne pas engorger la circulation des personnes à la sortie des salles,
- la préservation de l'anonymat des votants.

Autre contrainte à prendre en compte : le coût. Avec six salles de conférence à équiper, la dépense devait être contenue.
Enfin, nous souhaitions éviter l'effet "je joue avec tous les boutons" et préserver la notion d'unicité du vote. Nous avons donc choisi d'équiper les participants de Devoxx d'une puce NFC, contenant un identifiant unique. Ces puces ont été collées sur des badges blancs, pour garantir qu'une puce ne puisse pas être associée à une personne physique Fig.1.
Les conditions de réalisation mises en place, nous pouvions commencer le prototypage.

Le matériel

Première question, quel matériel pour nos bornes de vote ?
Notre choix s'est porté sur des RaspberryPi. En tant qu'utilisateurs, nous avions l'habitude de ce hardware. Il est facilement disponible et peu onéreux. De plus, il possède un Operating System issu de Debian, qui propose toutes les fonctionnalités que nous avons l'habitude de trouver sur un serveur classique. Il offre aussi un large panel de langages de développement non négligeable.
Enfin, les 16 pins GPIO (General Purpose Input Output) du PI permettent d'ajouter facilement des devices physiques (écran, bouton, led...).
Il existe une large communauté d'utilisateurs et de boutiques qui fournissent un nombre important de composants. L'une d'elles, Adafruit, propose une gamme étendue de produits, et offre un grand nombre de tutoriels sur son blog.
Nous avons lancé les commandes pour notre prototype : cinq boutons pour pouvoir voter et un écran LCD afin d'avoir un minimum de feedback visuel sur l'opération de vote. Seul le lecteur NFC, branché en USB afin de faciliter et d'alléger le système, a été acheté à part.

Le prototypage

Adafuit possède un repository Github, contenant du code Python, pour rapidement tester le matériel. Le système basique d'I/O a été rapidement opérationnel : allumer l'écran, afficher un message, recueillir une pression sur un bouton ... La première version créée s'est faite sur une breadboard. Ainsi, nous avons rapidement validé notre configuration physique.
Depuis le début, notre volonté a été de créer la totalité du projet en open source, des plans jusqu'au code. Nous avons donc mis à disposition notre schéma électrique, ainsi que la composition de la breadboard Fig.2 et 3.

Fig.2

La breadboard

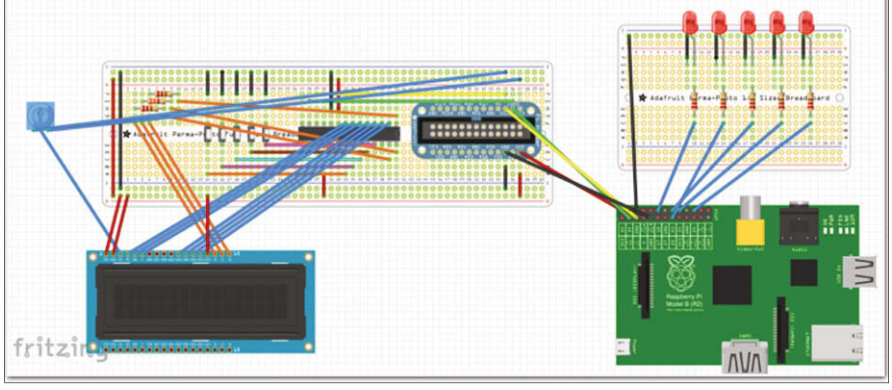


Fig.3

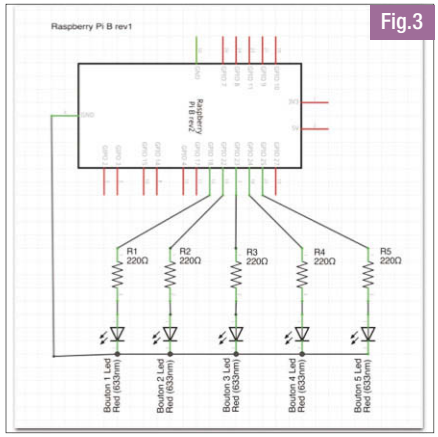


Schéma électrique des boîtiers de vote

Concernant la lecture du tag NFC, Raspbian propose un certain nombre de packages natifs qui permettent de détecter le lecteur USB, et d'interagir avec lui.
Tous les ingrédients unitaires sont là. Il est maintenant temps de passer au développement qui permettra de les intégrer.

Le Soft

Grâce à la complétude de la Raspbian, tous les choix étaient ouverts : NodeJs, Python, Java ... Nous nous sommes dirigés vers un de nos langages de prédilection : Groovy, porté par un framework événementiel, Vert.X.
Le séquencement des opérations de vote et les interactions avec le boitier se prêtaient particulièrement bien à une modélisation sur le bus d'évènements de Vert.x. Nous avons opté pour 3 verticles (l'unité de déploiement sur la plateforme Vert.x) communiquant entre eux via le bus interne :

- verticle électronique : dédié à la communication avec le matériel (boutons, LEDs, écran),
- verticle NFC : en écoute sur le lecteur NFC,
- verticle persistance : chargé de sauvegarder les votes en local et de cadencer la remontée des votes au central,

Pour le verticle électronique, il fallait migrer nos scripts Python d'interaction avec le matériel externe vers Groovy. Heureusement, une grande partie du travail est déjà réalisée dans la librairie open source pi4J. La gestion des LEDs et des boutons via le GPIO étant déjà packagés, nous avons seulement adapté les classes nécessaires à la gestion du LCD.
La gestion du NFC est fournie en natif dans Java, via l'API javax.smartcardio.
Pour la persistance, nous avons choisi de rendre nos boitiers indépendants de leur environnement. Chaque vote possède donc un timestamp, l'uid de la puce nfc associée, la note, et l'uid du boitier.
La communication au serveur central, qui possède l'intelligence (agenda et localisation géographique des boitiers) devaient transiter via le wifi. Dans une conférence comme Devoxx, la stabilité du wifi est toujours une inquiétude. Il était nécessaire de faire un stockage en local et d'avoir un mécanisme d'envoi vers un serveur central, avec une gestion des coupures éventuelles. Pour cela nous nous sommes plongés vers les standards émergeants de l'internet des objets. Nous y avons choisi un protocole : MQTT (Message Queuning Telemetry Transport), léger et pensé pour faire de la communication machine à machine de façon très épurée. Pour découpler au maximum le serveur Vert.x de l'envoi des données au central, nous nous sommes reposés sur la notion de bridge entre deux brokers. Nous avons choisi un broker open-source, Mosquitto, qui gère de manière native ce mode bridge et permettait donc de coder sans se soucier de l'envoi des données vers le central : Vert.x poste les messages des votes dans un Mosquitto local qui, via le mode bridge, garantit la livraison des données vers le Mosquitto central Fig.4.

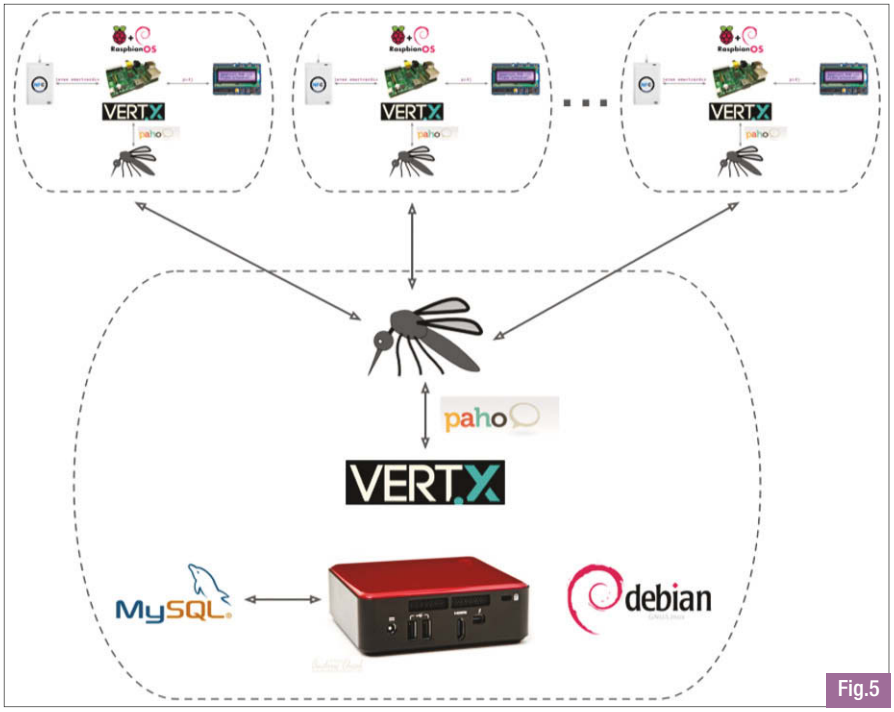


Fig.5

Agrégation des votes

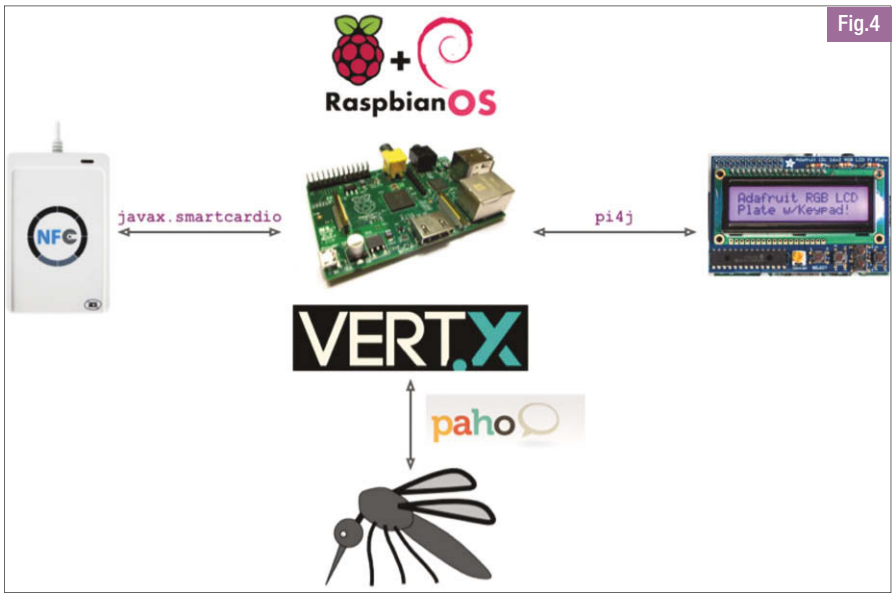


Fig.4

Vue logicielle des boitiers.

L'agrégation des votes

Une fois parvenus sur le Mosquitto du serveur central, les messages sont traités avec la même mécanique : un verticle dépile le message et va l'enregistrer dans une base de données MySQL.
Le choix d'une base relationnelle comme MySQL permet de gérer simplement la déduplication. En composant une clef primaire "intelligente", seul le dernier vote d'un utilisateur, pour un créneau horaire donné, est pris en compte. De plus, le modèle relationnel nous offre des possibilités de requêtage étendues, et nous permet de gérer facilement l'intelligence du central. L'organisation des conférences (créneau horaire, localisation par salle, gestion par track) se prête idéalement à une modélisation relationnelle Fig.5.

Cloud

Dernier étage de la fusée, pour que les données soient accessibles de partout, nous avons

Fig.6

La version finale

déployé une architecture similaire à l'architecture d'agrégation dans le Cloud. Notre serveur d'agrégation possédait un bridge vers un MQTT en SaaS, CloudMQTT. Les messages de ce dernier broker étaient consommés par un Vert.x déployé sur CloudBees.

La visualisation des résultats

Nous avons mis en place deux systèmes de visualisation. Le premier, personnel, permet à un utilisateur équipé d'un téléphone NFC (Nexus, Nokia...) de passer sa puce au dos de l'appareil et de retrouver ses votes.
Le second permettait de consulter, sur le tweet-wall géant du Marriott, les sessions les mieux notées. Cette fonctionnalité repose sur la facilité de créer des services REST avec Vert.x

Le boîtier

Nous aurions aimé avoir un superbe boîtier transparent en plexiglass. Malheureusement, pour des contraintes de temps et de coût, nous nous sommes rabattus vers un boîtier plastique / aluminium, qui possédait l'avantage de pouvoir être usiné facilement à partir des outils dont nous disposions (perceuse, Dremmel...) Fig.6.

En live

Heureux et fiers du travail accompli, la mise en place du projet débuta la veille de Devoxx France. Et comme lors de toute mise en prod, il y a eu un léger hic : l'épaisse moquette du Marriot créa des phénomènes d'électricité statique, qui faisaient "disjoncter" les boitiers. Tout bon électronicien y aurait pensé mais, en tant que bons informaticiens, nous avions totalement omis cette composante. Une nuit blanche et trois bidouilles électriques plus tard, le mal était réparé, mais nous nous rappellerons de cette fichue moquette !

Conclusion

Nous avons beaucoup appris en remettant les mains dans le physique. Et grâce à du matériel enfin abordable, à un mouvement open hardware de plus en plus développé, la créativité des développeurs est favorisée.
Nous avons vu les initiatives de ce type se multiplier : de manière industrielle, d'abord, avec des bornes de satisfaction qui fleurissent dans les supermarchés et les agences RATP. De manière plus artisanale ensuite, avec nos collègues du BreizhCamp qui ont eux aussi proposé leur boîtier de vote (la LikeBox).
Pour nos boitiers de votes, l'histoire n'est pas finie : depuis Devoxx France, ils ont déjà connu deux nouvelles conférences (Xebicon aux Pays Bas, et Devoxx UK). Et de nouvelles fonctionnalités sont apparues (iBeacon, réseau mesh indépendant) ou sont à l'étude (autoconfiguration...).
Nous espérons que cet article vous aura donné envie de réaliser de nouveaux objets intelligents, et que nous les verrons fleurir dans vos GitHub d'ici peu.

Merci à Aurélien **Maury**, Julien **Buret**, Thomas **Guerin**, Simone **Civetta**, Benjamin **Lacroix**, Antoine **Michaud**, Mathieu **Bigorne**, et Geoffroy **Warin** d'avoir permis à ce projet d'exister.

Pablo **LOPEZ**, *CTO chez Xebia France*

JACK : Jack Audio Connection Kit

Férus de nouvelles technologies et de musique, cet article devrait vous plaire. Etant moi-même un passionné de musique, je me suis souvent posé des questions existentielles de la sorte :
Comment faire pour rooter mes signaux audio d'un périphérique physique à un autre ?
Comment streamer un groupe complet de musiciens via internet ?
Comment faire dialoguer entre elles mes applications audio ?

J'ai alors commencé mes recherches sur le web pour me rendre compte qu'il existait une solution toute faite qui répondait exactement à toutes ces questions (une chance !).
Cette solution, c'est JACK, acronyme de Jack Audio Connection Kit.
Dans cet article, nous verrons ce qu'est JACK, comment l'utiliser sur OSX, et nous programmerons un petit client JACK en C.

JACK, qui es-tu ?

JACK est un logiciel permettant de rooter des flux audio et MIDI à l'intérieur de votre ordinateur. Il fonctionne sous Linux, OSX et Windows. On peut l'utiliser avec des périphériques externes branchés à notre ordinateur ou directement entre des applications installées sur nos machines. Il permet aussi la diffusion sur un réseau (mais nous n'aborderons pas ce point dans cet article). JACK est composé d'une API, de diverses implémentations de cette API, d'applications de contrôle sous forme d'interfaces utilisateur et d'instances au travers desquelles d'autres applications peuvent dialoguer avec JACK. JACK est un logiciel open source, dans sa version 0.124.1 depuis le 22 janvier 2014 et en perpétuel développement.

JACK, que fais-tu ?

Je vais aborder les fonctionnalités de JACK par l'exemple. En effet, qui n'a jamais rêvé de pouvoir faire un concert chez soi, avec ses amis musiciens, et le retransmettre en streaming sur Internet via les plateformes habituelles ? J'utilise une carte son Focusrite LIQUID Saffire 56 branchée en FireWire à mon iMac pour notre exemple mais il existe des cartes avec plusieurs entrées moins onéreuses. Cette carte possède 28 entrées et 28 sorties. Je n'utilise pour notre exemple que 4 entrées de la carte son (une pour un micro statique, une pour un micro dynamique et deux autres pour brancher la sortie d'un pédalier à simulateur d'amplificateurs en stéréo).
Le fonctionnement est le même que vous utilisiez une entrée de votre carte son ou plusieurs cartes son avec des dizaines d'entrées. Pour diffuser ce que nous produirons, nous allons nous pencher sur Ustream. Ustream est une plateforme qui permet de faire du streaming sur Internet depuis n'importe où. Je me connecte donc sur le site de Ustream, je m'authentifie, je créé ma chaîne Ustream, je commence la diffusion et là, surprise, seule la première entrée de ma carte son externe est reconnue par Ustream. C'est une mission pour JACK.
Je lance donc le serveur JACK sur mon iMac Fig.1.
Le logiciel que je lance sur0 OS X s'appelle JackPilot. Il sert à démarrer le serveur JACK et à configurer le routage des flux audio.
Comme l'interface de JackPilot est un peu rudimentaire (même si elle reste tout à faire utilisable), j'utilise un autre logiciel qui, lui, est fourni dans toutes les distributions de JACK. Il se nomme qjackctl. Son interface est beaucoup plus riche et permet de faire beaucoup de chose Fig.2.
Les différentes fonctions de qjackctl sont :

- Démarrer un serveur JACK.
- Arrêter un serveur JACK.
- Avoir une fenêtre de logs des messages du serveur JACK.
- Voir le statut du serveur JACK.
- Faire le routage des flux audio (ce qui nous intéresse le plus).

Fig.2

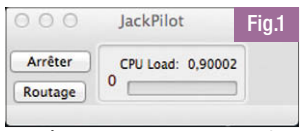


Fig.1

- Faire des configurations prédéfinies de flux audio (Patchbay).
- Paramétrer les différentes options possibles du serveur JACK.

Le bouton « Connect » nous permet donc de réaliser le routage de nos flux audio avec plus de facilité que JackPilot.
Voici à quoi ressemble la fenêtre de routage des flux dans qjackctl : Fig.3.
La partie gauche correspond aux différents émetteurs et la partie de droite aux différents récepteurs.
La partie centrale (les traits) représente les connexions en cours.
Ici, on voit que Google Chrome (qui héberge la version web du site Ustream) peut jouer le rôle de récepteur comme d'émetteur. On voit aussi toute une ribambelle de récepteurs et d'émetteurs sous « system ». Il s'agît en fait de ma carte son qui a 28 entrées et 28 sorties possibles. L'astuce pour Google Chrome (mais cela fonctionne aussi avec n'importe quel logiciel ayant moins de récepteurs que vous n'avez d'émetteurs, comme Skype) est de faire la connexion entre chacun de vos émetteurs vers le premier récepteur. C'est tout simple, mais ça fonctionne plutôt bien.

JACK, m'entends-tu ?

Si tout se passe bien, vous devriez pouvoir voir la barre de son de Ustream bouger (n'oubliez pas de monter le gain de vos préamplis sur votre carte son externe).
Dans un monde parfait, JACK fonctionnerait sur toutes les plateformes avec tous les logiciels. Néanmoins, j'ai pu constater que, sur un WINDOWS 8.1 en version 64 bits, il peut s'avérer difficile de reconnaître les applications réceptrices et émettrices. J'ai pour ainsi dire tout essayé

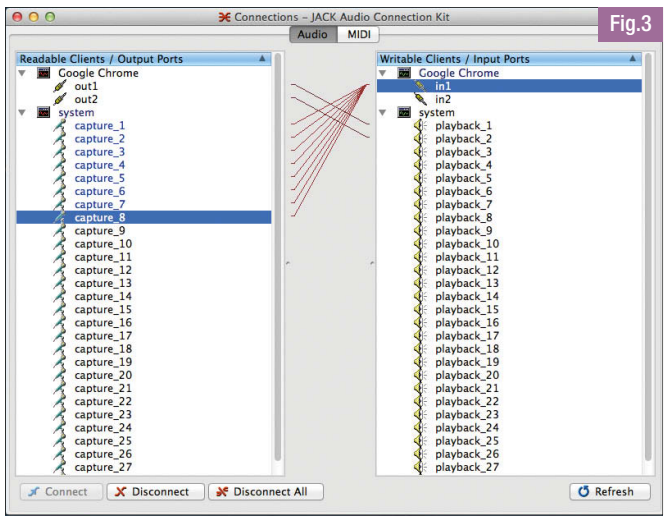


Fig.3

pour que le serveur JACK retrouve ses petits, mais rien n'y a fait, je suis resté sur ma faim de ce côté-là. Malgré tout, sur OSX je n'ai pas eu le moindre problème. D'après les forums spécialisés, il n'y a pas non plus de soucis sous LINUX mais je n'ai pas moi-même testé.

JACK, me codes-tu ?

Nous allons passer maintenant à la partie développement de cet article. Dans cette partie nous allons faire un client tout simple qui utilisera un serveur JACK déjà lancé pour réaliser une connexion entre un émetteur et un récepteur.

```
#include <stdio.h>
#include <errno.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

#include <jack/jack.h>

jack_port_t *port_recepteur;
jack_port_t *port_emetteur;
jack_client_t *client;
```

Cette première fonction permet de réaliser le transfert de nos trames audio de notre port d'entrée vers notre port de sortie. (C'est la fonction principale de notre programme qui sera répétée tout au long de ce dernier).

```
/**
*Petite fonction permettant de réaliser le process réellement
exécuté.
*/
int
process (jack_nframes_t trames, void *arg)
{
 jack_default_audio_sample_t *entree, *sortie;

 entree = jack_port_get_buffer (port_recepteur, trames);
 sortie = jack_port_get_buffer (port_emetteur, trames);
 memcpy (sortie, entree,
   sizeof (jack_default_audio_sample_t) * trames);

 return 0;
}
[/CODE]
```

Ce tout petit bout de code sert à arrêter le programme si le serveur JACK s'arrête prématurément ou s'l décide de déconnecter le client.

```
void
jack_shutdown (void *arg)
{
 exit (1);
}
```

Voici le corps de notre programme. Il est constitué de plusieurs parties :

- La déclaration des constantes,
- La déclaration des options,
- La déclaration des variables,
- L'ouverture de la connexion vers le serveur JACK (avec gestion des erreurs),
- L'utilisation de la fonction « process » définie plus haut,
- L'utilisation de la fonction « jack_shutdown » définie plus haut,
- L'affichage de certaines informations (Taux d'échantillonnage),
- La création de deux ports (pour l'entrée et la sortie de notre flux audio),
- L'activation de JACK et la connexion des ports (avec la gestion des éventuelles erreurs),
- Le processus continue tant qu'il n'y a pas de contrordre,
- La fermeture propre du client.

(Se référer aux commentaires pour voir dans le code où se situe telle ou telle partie).

```
int
main (int argc, char *argv[])
{
 const char **ports;
 const char *nom_client = "simple";
 const char *nom_serveur = NULL;
 jack_options_t options = JackNullOption;
 jack_status_t status;

 /* Ouvre une connexion vers le serveur JACK */

 client = jack_client_open (nom_client, options, &status,
nom_serveur);
 if (client == NULL) {
   fprintf (stderr, "jack_client_open() echec, "
      "status = 0x%2.0x\n", status);
   if (status & JackServerFailed) {
     fprintf (stderr, "Impossible de se connecter au serveur
JACK\n");
   }
   exit (1);
 }
 if (status & JackServerStarted) {
   fprintf (stderr, "Serveur JACK en cours d'execution\n");
 }
 if (status & JackNameNotUnique) {
   nom_client = jack_get_client_name(client);
   fprintf (stderr, "nom unique `%s' pris\n", nom_client);
 }

 /* On appelle le processus défini plus haut tant qu'il y a
 des choses a faire
 */

 jack_set_process_callback (client, process, 0);

 /* On dit au serveur JACK d'appeler la fonction jack_shutdown
 si on en a besoin
 */

 jack_on_shutdown (client, jack_shutdown, 0);

 /* On affiche le taux d échantillonnage courant.
  */

 printf ("taux d echantillonnage : %" PRIu32 "\n",
   jack_get_sample_rate (client));

 /* on fait deux ports en utilisant les valeurs par défaut */
```

```
port_recepteur = jack_port_register (client, "input",
          JACK_DEFAULT_AUDIO_TYPE,
          JackPortIsInput, 0);
port_emetteur = jack_port_register (client, "output",
           JACK_DEFAULT_AUDIO_TYPE,
           JackPortIsOutput, 0);

if ((port_recepteur == NULL) || (port_emetteur == NULL)) {
  fprintf(stderr, "plus de ports JACK disponibles\n");
  exit (1);
}

/* On est prêt pour activer JACK ! Tout en faisant attention
aux éventuelles erreurs */

if (jack_activate (client)) {
  fprintf (stderr, "impossible d activer JACK");
  exit (1);
}
/* on connecte les ports
 */

ports = jack_get_ports (client, NULL, NULL,
      JackPortIsPhysical|JackPortIsOutput);
if (ports == NULL) {
  fprintf(stderr, "pas de port disponible\n");
  exit (1);
}

if (jack_connect (client, ports[0], jack_port_name (port
_recepteur))) {
  fprintf (stderr, "pas de port disponible\n");
}
```

```
free (ports);

ports = jack_get_ports (client, NULL, NULL,
      JackPortIsPhysical|JackPortIsInput);
if (ports == NULL) {
  fprintf(stderr, "pas de port disponible\n");
  exit (1);
}

if (jack_connect (client, jack_port_name (port_emetteur),
ports[0])) {
  fprintf (stderr, "pas de port disponible\n");
}

free (ports);
/* on continue à tourner tant qu'il n'y a pas de contrordre */

sleep (-1);

/* Il est important de fermer le client.
*/

jack_client_close (client);
exit (0);
}
```

Pour aller plus loin

Pour aller plus loin je vous conseille d'essayer par vous-même de faire un petit client JACK. Si vous avez besoin de renseignements complémentaires, toutes les informations sont disponibles sur le site http://jackaudio.org/ (momentanément hors ligne au moment où j'écris ces lignes).

Patrick-André **Marendat**
Softfluent

Votre veille technologique au quotidien

- Les dernières **actus**
- La **newsletter hebdo** :
  la synthèse des informations indispensables.
- **Agenda** :
  Tous les salons, hackathon et conférences.

Abonnez-vous, c'est gratuit !

www.programmez.com

Partager des données entre ASP et ASP.NET en utilisant la mise en cache distribuée hautes performances

ASP reste toujours très utilisé alors qu'ASP.NET est arrivé sur le marché il y a déjà plus de 6 ans. Il y a encore de nombreux développements qui n'ont pas encore migré d'ASP à ASP.NET en raison du prétendu manque de bénéfices d'ASP.NET. Dans cet article, mon propos est de vous montrer comment prendre une application ASP pour la migrer en ASP.NET.

Vous avez la possibilité de migrer l'application en entier en une seule fois en ASP.NET en tant qu'un seul projet, mais cela augmente alors le risque de pannes et des délais fortement rallongés. Vous pouvez aussi prendre une approche itérative et incrémentale plus facile à gérer en augmentant vos chances de réussir plus vite.

Migrer l'application en entier en une seule fois peut prendre de 1 à 2 ans alors que la même migration avec l'approche incrémentale prendra 2 à 3 mois. La migration incrémentale vous donne aussi l'avantage d'analyser les problèmes identifiés dans chaque itération et vous permet d'améliorer les choses pour l'itération suivante. Enfin, le plus important bénéfice d'une approche itérative est que vous allez respecter vos deadlines plus facilement.

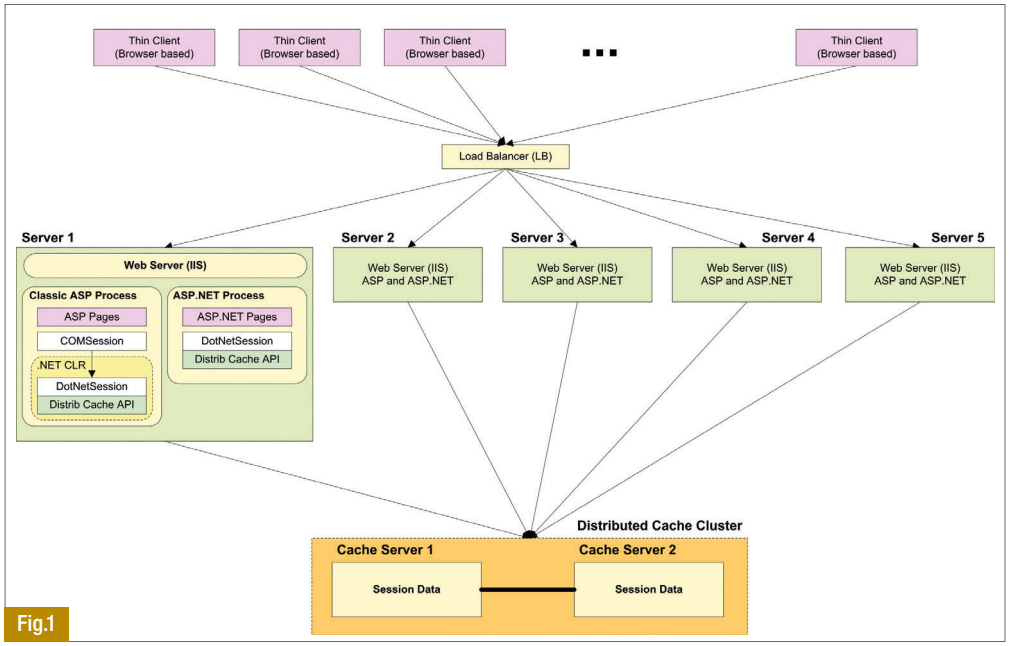


Fig.1

Si vous décidez de transcrire votre application ASP par petites itérations, alors vous vous retrouverez dans la situation où votre application fonctionne de concert avec les portions qui ont déjà été migrées en ASP.NET. Il va de soi que dans ce cas vous devez donner à l'utilisateur la sensation de passer de manière transparente d'une partie importante de l'application à une autre sans qu'il perde sa session. Dans les faits, vous devez partager l'état de la session entre les applications en ASP et ASP.NET à l'aide de COM+.

En plus des données de session, les applications ASP et ASP.NET doivent également partager des données d'application. Ce n'est toutefois pas un problème parce que ces données sont stockées dans une base de données commune. ASP comme ASP.NET peuvent tous deux accéder à ces données grâce à leur propre code d'accès de base de données, que ce soit avec ADO ou ADO.NET. Mais ces sessions ne sont pas automatiquement partagées. Vous devez au préalable fournir un mécanisme COM+ dans lequel les sessions peuvent être partagées au travers des applications. Dans cet article, je vous montre comment vous pouvez partager les sessions entre l'ASP classique et l'ASP.NET.

Partager l'état des sessions grâce au cache distribué

Billy Yuen (Microsoft) a écrit un article sur le partage d'état de sessions entre ASP et ASP.NET (http://msdn.microsoft.com/en-us/library/aa479313.aspx). Toutefois, l'approche prise ici prend une tournure différente. En suivant celle-ci, vous obtiendrez une solution plus évolutive, associée à une meilleure performance et une plus grande fiabilité.
M. Yuen favorise l'idée de stocker les sessions dans la base de données en SQL Server, puis de les partager entre ces deux applications. Cela signifie que l'ASP et l'ASP.NET vont tous deux chercher leurs sessions dans la base de données du serveur SQL. La raison en est que ces deux applications ne s'exécutent pas dans le même processus de travail et peut-être même pas sur la même machine. Par conséquent, ils ont besoin de stocker les sessions dans un emplacement commun afin de pouvoir les partager, d'où la nécessité d'une base de données SQL Server.
Toutefois, le stockage des sessions dans une base de données SQL Server fait se poser des questions en termes de performances et d'évolutivité; si votre application est largement utilisée, vous allez immédiatement avoir à faire face à des problèmes.

Au lieu de reposer sur une base de données SQL Server, notre approche utilise un cache en mémoire distribuée comme espace de stockage et y stocke les sessions à la place de le faire dans une base de données. Un cache distribué est un cache en mémoire qui peut aller déclarer un seul cache sur plusieurs serveurs et vous proposer une vue logique globale du cache en permanence synchronisée sur l'ensemble des serveurs et processus.
Dans la Figure 1, vous voyez un schéma montrant comment un cache distribué est utilisé pour partager des sessions, non seulement entre l'ASP et l'ASP.NET mais aussi entre de multiples instances de ces applications dans une ferme de serveurs web.

Comme vous pouvez le voir, l'ASP et l'ASP.NET utilisent tous deux un cache commun et distribué qui vit dans ses propres processus. Ce cache distribué synchronise en permanence toutes les mises à jour des caches des serveurs du cluster. Il y a plusieurs caches distribués gratuits ou

payants disponibles pour ASP.NET. Parmi les gratuits, il y a Memcached, Velocity et NCache Express. J'utilise NCache Express dans les exemples de code que je vous détaille mais vous pouvez le remplacer par votre cache distribué favori.

Implementation dans ASP.NET

Comme vous le montre la Figure 1, la session en cours est implémentée dans l'environnement .NET et les applications en ASP classique comme en ASP.NET peuvent y accéder. Voici ce que vous devez implémenter dans ASP.NET pour le réaliser.

Classe DotNetSession

Cette classe remplace la version standard de la classe d'état de session d'ASP.NET et fournit la même interface pour les opérations les plus courantes. Il s'agit vraiment d'une classe qui ne s'occupe que de maintenir les données; elle ne s'occupe ni de les charger, ni de les sauvegarder dans le cache distribué. Notez également que cette classe ne fournit pas toutes les méthodes publiques de la classe de gestion de l'état de session ASP.NET, étant donné que le sujet est ici de ne donner qu'un exemple simple. Voici le code source pour cette classe:

```
[Serializable]
public class DotNetSession
{
  private Hashtable data = new Hashtable();

  public DotNetSession() {}

  public string this[string name]
  {
    get { return (string)data[name.ToLower()]; }
    set { data[name.ToLower()] = value; }
  }
}
```

Interface IDotNetSessionFactory

Ce qui suit est l'interface qui représente tous les comportements relatifs au chargement et à la sauvegarde des DotNetSession dans le cache distribué. Il utilise à la fois du code ASP.NET et Visual Basic 6.0.

```
public interface IDotNetSessionFactory
{
  // Recupère et charge la session dans le cache distribué
  DotNetSession GetSession(string sessionId);
  void Save(string sessionId, DotNetSession Session, int
expire);

  // Génère une sessionId (GUI + appId). Elle est sauvegardée
  // dans le HttpCookie et est également utilisée comme clé
dans le cache distribué.
  // SessionCookieName est le nom du cookie.
  string GenerateSessionId(string appId);
  string SessionCookieName { get; }
}
```

Classe DotNetSessionFactory

Cette classe implémente l'interface IDotNetSessionFactory. Code complet sur www.programmez.com

GetSession() est responsable du chargement de la session depuis le cache distribué. La "clé" utilisée est la sessionId qui est une combinaison d'un GUID et d'une AppId fournie par l'utilisateur. L'AppId autorise l'utilisateur à ne pas partager des sessions au travers de plusieurs domaines d'applications ou au travers d'applications ASP.NET et ASP. En revanche, si appId est laissée vide ou si appId est la même pour toutes les applications, cela implique que la session est partagée.

Save() method est chargée de sauvegarder DotNetSession dans le cache distribué. Pendant la sauvegarde, elle spécifie également l'expiration décalée. L'expiration décalée est une caractéristique du cache distribué qui demande au cache d'automatiquement expirer la session au cas où personne n'effectue de "get" ou "save" dans l'intervalle de temps d'expiration spécifié. Dans ce sens, l'application cliente n'a pas à se préoccuper de savoir quand vider une information en cache.

Finalement, une DotNetSession doit être chargée à plusieurs reprises pendant une session utilisateur. Ceci est fait en créant une SessionID (opération réalisée avec une GUID et une AppId dans cet exemple) et en la stockant dans le HttpCookie des applications ASP et ASP.NET. Cela garantit que lorsque la prochaine requête d'un utilisateur arrive, l'en-tête Http contient le HttpCookie et que la SessionID est alors utilisée pour aller chercher la session dans le cache distribué.

Classe BaseSessionPage

C'est la classe de base pour toutes les pages ASP.NET et pour encapsuler l'objet DotNetSession object. Code complet sur www.programmez.com

La méthode **OnInit()** de cette classe crée d'abord une nouvelle session vide DotNetSession ou charge une session existante depuis le cache distribué si l'en-tête Http contient une SessionId dans le cookie. Puis cela inscrit la méthode SaveSession() afin qu'elle soit appelée lors d'une demande Unload. SaveSession() sauvegarde la DotNetSession dans le cache distribué de façon à ce que la requête web suivante soit capable de la trouver. Toute cette logique est masquée de l'ensemble des classes de Pages dérivées, par conséquent les sessions sont gérées de manière transparente sur les pages dérivées qui en connaissent tous les rouages.

Implémentation en ASP

Après avoir créé DotNetSession, IDotNetSessionFactory, et DotNetSessionFactory du point de vue .NET, vous êtes maintenant prêt à les appeler d'un point de vue ASP. Cependant, contrairement à ASP.NET avec lequel vous pouvez directement appeler IDotNetSessionFactory et DotNetSession, en ASP, vous devez au préalable créer une classe COM+ (appelée COMSession dans notre exemple) qui va encapsuler tout le comportement de la session. Cette classe COM+ appelée alors IDotNetSessionFactory et DotNetSession pour les pages en ASP classique. Ci-dessous l'exemple de code pour le faire :

Classe COMSession

Cette classe est implémentée en Visual Basic et est utilisée en tant qu'objet COM+ d'une page ASP. Code complet sur www.programmez.com

Dans cette classe identique à la classe ASP.NET BaseSessionPage, principalement trois choses sont réalisées. En premier, une nouvelle DotNetSession est créée dans le cache distribué où une déjà existante est chargée. En second, à chaque fois qu'une page ASP veut accéder à l'objet DotNetSession pour lui appliquer un “get” ou “put”, il lui en est retourné une depuis la méthode Contents.

Enfin, à la fin de la Page ASP, quand la classe est terminée la DotNetSession est sauvegardée dans le cache distribué.

Page SessionInclude.asp

Cette page est incluse dans toutes les pages ASP par une instruction Include. C'est une page très simple qui crée seulement un objet COM+ appelé COMSession.

```
<%
Dim Session
Set Session = Server.CreateObject("SharedSession.COMSession")
%>
```

Avec cette instruction, toutes les pages ASP classiques se terminent en utilisant cette classe COMSession avec la variable de Session plutôt que l'objet standard de session ASP.

Limitations et considérations spécifiques

A chaque fois que vous partagez des sessions entre de l'ASP et de l'ASP.NET, vous devez garder en tête les limitations suivantes :

- **ASP utilisant des objets COM personnalisés.** Si votre application ASP stocke des objets COM+ personnalisés dans la Session, vous devrez alors développer un code de sérialisation pour chacun de ces objets COM. Ce code de sérialisation transforme l'objet COM+ personnalisé dans un format portable, dont la forme se rapproche d'une chaîne de caractères ou d'un document XML; ce code est alors stocké dans DotNetSession. Ceci est dû au fait qu'un objet COM+ standard ne peut pas être transporté depuis COM à .NET.
- **Partager des données non-primitives ASP et ASP.NET.** Même si vous partagez l'objet de session entre de l'ASP et de l'ASP.NET, vous ne pouvez partager que des données de types primitifs. Cela inclut les chaînes, les nombres, etc. Si vous avez stocké des objets COM personnalisés dans un code sérialisé, cela peut ne pas être lisible côté .NET jusqu'à ce que vous écriviez un code personnalisé en ASP.NET qui transcrive cela dans une classe .NET ou quelque chose d'équivalent. En définitive, si vous avez un objet personnalisé du côté COM, vous avez à la fois à comprendre sa version sérialisée (peut-être en XML) ou à avoir le code pour retransformer sa version sérialisée en objet personnalisé du côté .NET.

Sélectionner le cache distribué le mieux adapté

Avant que vous sélectionniez un cache distribué, vous devriez garder en tête les caractéristiques suivantes pour être capable de le sélectionner au mieux :

- **Performance & Evolutivité:** un cache distribué doit avec les meilleures performances possibles. En outre, il doit pouvoir évoluer de manière totalement transparente lorsque votre application en a besoin. L'évolutivité signifie ici qu'il doit pouvoir fournir les mêmes performances même lorsque la charge en transactions s'accroît. De plus, un cache distribué optimisé doit pouvoir le faire en vous permettant d'ajouter plus de serveurs de cache dans un cluster afin d'évoluer de manière linéaire.
- **Fiabilité:** un cache distribué doit intrinsèquement être très fiable. La fiabilité ici signifie qu'il ne peut pas y avoir de perte de données même lorsqu'un serveur de cache tombe en panne. Pour les sessions en ASP et ASP.NET, vous ne pouvez pas vous permettre de perdre la moindre donnée car il n'y a pas de stockage maître autre que le cache qui stocke ces données.
- **Disponibilité:** la principale caractéristique est la haute disponibilité, ce qui signifie que le cache est un serveur de stockage de session pour votre application et devrait à ce titre être disponible 100% du temps. De ce fait, si vous avez besoin d'ajouter ou retirer un serveur de cache dans votre cluster, vous devriez être capable de le faire sans arrêter quoi que ce soit. Ceci est particulièrement important dans le Cloud où vous auriez régulièrement à faire évoluer à la hausse ou à la baisse votre configuration afin de subvenir aux différents niveaux de charge de vos applications. Et vous devriez être capable de le faire sans interruptions de service.
- **Expirations décalées:** les expirations décalées permettent au cache d'automatiquement supprimer une session après un certain temps d'inactivité. De cette manière, si un utilisateur a fini une session, l'objet de session ne peut plus être accédé et le cache l'efface automatiquement dès que le temps d'expiration s'est écoulé.
- **Evictions:** un cache distribué stocke les données dans la mémoire et par conséquent doit pouvoir gérer les situations dans lesquelles il a utilisé toute la mémoire utilisable disponible. Quand le cache est plein, certaines données doivent être effacées pour faire de la place à de nouvelles. Cela est appelé "éviction". Un cache vide automatiquement certaines données basées sur une politique d'éviction. Trois politiques sont souvent utilisées : le moins fréquemment utilisé (Least Frequently Used ou LFU), le moins récemment utilisé (Least Recently Used ou LRU), et Priorité (Priority).
- **Topologies de cache:** un cache distribué doit pouvoir fournir différentes topologies pour s'adapter à votre environnement spécifique. Il y a le Cache Répliqué (Replicated Cache) et le Cache Partitionné (Partitioned Cache). Le Replicated Cache réplique le cache sur tous les serveurs de cache; il est destiné à de l'usage intensif en lecture. Le Partitioned cache est un cache à très haut niveau d'évolutivité pour les données transactionnelles; il partitionne le cache. Plus vous ajoutez de serveurs de cache plus il y a de partitions; il y a environ 1/N de cache (N pour le nombre de noeuds) stocké sur chaque serveur de cache. Le cache partitionné fournit aussi l'option de backup de chaque partition sur un serveur de cache différent afin d'améliorer la fiabilité.
- **Cache local ou distant:** vous devriez être capable de stocker le cache soit directement sur vos serveurs Web, soit dans un cache tiers auquel vous pouvez accéder depuis vos serveurs web. Un cache tiers séparé est mieux adapté si votre ferme de serveurs web a plus de 3 serveurs.

Conclusion

L'approche décrite dans cet article vise les mêmes objectifs que celle trouvée dans l'article de M. Yuen. Mais elle est ici quelque peu différente puisqu'elle propose une approche plus évolutive, avec de meilleures performances et une meilleure fiabilité dans le cas de l'usage de la réplication de session.

Parmi les problèmes soulevés par ASP, il y a le fait que vous ne disposez pas d'une flexibilité dans le stockage de session n'importe où en dehors de l'InProc. ASP.NET vous donne plus d'options de stockage en incluant InProc, StateServer, et SQLServer ou d'autres stockages personnalisés. Dans ce cas je propose le cache distribué comme solution personnalisée pour les sessions ASP.NET. Le cache distribué est en train de devenir le Best Practice dans le besoin en haut niveau de transactions et d'applications à fort trafic pour stocker n'importe quelles données, qu'il s'agisse de données d'applications ou de données de sessions utilisateur.

En prenant cette approche, vous pourrez vous assurer que vous n'aurez pas seulement une migration incrémentale entre ASP et ASP.NET, mais vous allez pouvoir bénéficier d'un coup d'accélérateur en performances et en évolutivité pour vos applications ASP et ASP.NET.

Iqbal **Khan**
président / CEO, Alachisoft
iqbal@alachisoft.com

Exportation de fichiers SVG pour le web avec Adobe Illustrator CC

2e partie

En tant qu'artiste et designer web, je suis ravi des possibilités que m'offre Adobe Illustrator CC en termes de gestion du format SVG (Scalable Vector Graphics) et de la souplesse qu'il me donne pour créer des graphiques de qualité. SVG étant un format vectoriel, je peux facilement créer des images qui s'adapteront automatiquement à tous les appareils modernes.

Conservation des fonctions d'édition d'Illustrator

Comme je l'ai dit précédemment, vous pouvez utiliser le format SVG pour enregistrer vos créations Illustrator. Le format XML autorisant l'ajout d'informations confidentielles, Illustrator peut enregistrer des paramètres d'édition dans le fichier. Ces informations complémentaires ne sont d'aucune utilité pour le web et risquent de faire exploser la taille du fichier SVG qui contient les images créées dans Illustrator. Pour éviter cela, désactivez la case à cocher Conserver les fonctions d'édition d'Illustrator.

Voici la différence de taille d'un fichier SVG ne contenant qu'un rectangle : Lorsque l'option Conserver les fonctions d'édition d'Illustrator est sélectionnée : 407 Ko. Lorsque l'option Conserver les fonctions d'édition d'Illustrator *n*'est *pas* sélectionnée : 0,5 Ko

OPTIONS D'OPTIMISATION RECOMMANDÉES

Dans la boîte de dialogue Options SVG, cliquez sur Plus d'options pour accéder à des options avancées indispensables aux designers web. Certaines d'entre elles proposent des astuces pour réduire la taille de vos fichiers SVG (voir Fig.1).

Les boutons Code SVG et Aperçu Web permettent de tester les options d'exportation et de prévisualiser le résultat avant enregistrement.

Définition des positions décimales

Le paramètre Positions décimales permet de définir le niveau de précision des vecteurs exportés (voir Fig.2).

Prenons l'exemple de la lune dans la Fig.3.

Voici le code exporté avec une position décimale définie à 7 :

```
<path fill="#F28E85" d="M31,94.6236572C31,47.644043,65.2392578, 8.6680908,110.1236572,1.270813 C105.0795288,0.4395142, 99.9032593,0,94.6236572,0C42.3644409,0,0,42.3644409,0, 94.6236572 c0,52.2591553,42.3644409,94.6236572,94.6236572, 94.6
```

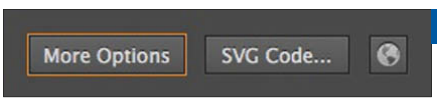


Fig.1 *Sélection des options d'optimisation avancées*

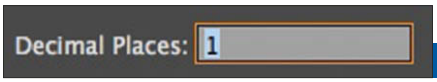


Fig.2 *Le paramètre Positions décimales permet de définir le niveau de précision des vecteurs exportés*

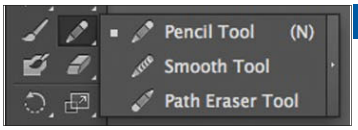


Fig.6 *Utilisation de l'outil Crayon dans vos créations SVG*

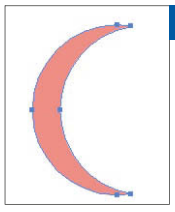


Fig.3 *Lune*

```
23652c5.2796021,0,10.4558716-0.4395142,15.5-1.270813 C65.2392578, 180.5792236,31,141.6032104,31,94.6236572z"/>
```

Voici, pour la même forme, le code exporté avec une position décimale définie à 1 :

```
<path fill="#F28E85" d="M31,94.6c0-47,34.2-86,79.1-93.4C105.1,0.4,99.9, 0,94.6,0C42.4,0,0,42.4,0,94.6c0,52.3,42.4,94.6,94.6,94.6 c5.3,0,10.5-0.4, 15.5-1.3C65.2,180.6,31,141.6,31,94.6z"/>
```

Dans le premier cas, les coordonnées vectorielles sont plus précises. Mais le code est beaucoup plus dense, la taille des fichiers augmente en raison de la position décimale choisie.

Théoriquement, ce paramètre joue sur la qualité des images, mais notez-vous vraiment une différence ? Faisons un test avec un fichier SVG plus complexe, un portrait de Napoléon.

Dans cet exemple, la même image a été exportée avec une position décimale définie d'abord à 7, puis à 1. Honnêtement, voyez-vous une différence ? En fait, ce qui change vraiment, c'est la taille des fichiers : 31 Ko pour le premier et 12,7 Ko pour le second.

Pour le web, je vais utiliser la position décimale 1.

Optimisation d'une illustration en réduisant le nombre de vecteurs

Plus vous ajoutez de points dans une illustration, plus la taille du fichier SVG augmente parce qu'il reprend toutes les coordonnées vectorielles. Pour réduire le nombre de vecteurs, vous pouvez utiliser l'outil Déformation (voir Fig.4). En tant que designer, je l'apprécie beaucoup, car il me permet de dessiner en préservant l'énergie de mon illustration. Cela reste toutefois une opération manuelle.

Illustrator fournit un algorithme fiable et vous permet d'utiliser un processus de simplification, accessible via le menu Objet > Tracé > Simplifier. Le problème avec ce processus, c'est qu'il est du type 'tout ou rien' (voir Fig.5). Avec l'outil Déformation, je peux choisir les parties à simplifier et

Fig.4 *L'outil Déformation (Maj.+R) permet de modeler les objets à l'aide de la souris (comme de la pâte à modeler)*

celles où je souhaite conserver certains détails.
Pour ajouter des détails ou simplifier certaines zones, vous pouvez utiliser l'outil Crayon (voir Fig.6). Appuyez sur la touche Alt pour activer l'outil Arrondi et réduire avec précision le nombre de points sur une courbe.

AUTRES OPTIONS POUR LES DESIGNERS WEB

Les options suivantes sont disponibles, mais pas forcément nécessaires :

- **Optimiser pour l'afficheur SVG Adobe :** cette option n'est pas nécessaire pour le développement web, car le format SVG est pris en charge par tous les navigateurs et ne nécessite donc pas l'utilisation d'un module externe.
- **Inclure les données de tranche :** cette option écrit des données XML dans les fichiers à des fins de référence via JavaScript, ce qui est inutile dans cet exemple.
- **Inclure les informations XMP :** cette option est intéressante si vous travaillez avec une grande équipe ou si vous avez besoin de fournir des métadonnées XML.

Production limitée d'éléments <tspan>

Si votre illustration SVG contient beaucoup de texte, l'option Produire moins d'éléments `<tspan>` peut réduire considérablement la taille du fichier SVG exporté. Mon exemple ne contenant que le terme « Adobe », j'ai désactivé l'option à l'exportation :

```
<tspan x="0" y="0" font-family="'Myriad-Roman'" font-size="136.1154" letter-spacing="-1">A</tspan><tspan x="81.7" y="0" font-family="'Myriad-Roman'" font-size="136.1154">dobe</tspan>
```

Illustrator CC a généré deux éléments `<tspan>` : un premier pour la lettre majuscule « A », et un second pour les lettres « dobe ». Myriad-Roman applique un crénage spécifique pour les lettres majuscules et génère, de ce fait, un élément `<tspan>` distinct.
Le texte est problématique avec le format SVG, car il est difficile de retranscrire parfaitement une police CoolType sans sélectionner de glyphes spécifiques. L'ajout d'éléments `<tspan>` convient mais fait perdre la signification sémantique.
Lorsque vous sélectionnez cette option, Illustrator CC ignore les positions de crénage et crée un élément `<tspan>` :

```
<text transform="matrix(1 0 0 1 36.5907 137.4181)" font-family="'Myriad-Roman'" font-size="136.1154">Adobe</text>
```

La taille du fichier SVG est réduite, donc si vous prévoyez de modifier le texte de manière dynamique, vous pourrez le faire aisément grâce au code JavaScript.

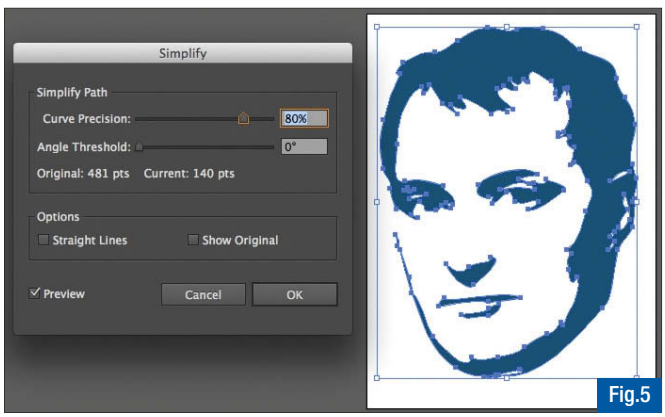


Fig.5

Utilisation du processus Simplifier

Utilisation de l'élément <textPath> pour le texte curviligne

Pour tout texte ancré sur un tracé, je vous conseille d'utiliser l'élément `<textPath>` en sélectionnant l'option Utiliser l'élément `<textPath>` pour le texte curviligne (voir Fig.7). Sinon, un élément `<tspan>` sera déclaré pour chaque caractère. L'élément `<textPath>` simplifie le code :

```
<path id="SVGID_x5F_1_x5F_" fill="none" d="M38.7,308.5c0,0,58.1-234.4, 179.6-160.2s138.7,89.2,202.2,2.2S623.7,59,623.7,59"/> <text> <textPath xlink:href="#SVGID_x5F_1_x5F_" startOffset="5%"> <tspan font-family="'MyriadPro-Regular'" font-size="23">A fool thinks himself to be wise, but a wise man knows himself to be a fool.</tspan> </textPath> </text>
```

Petite astuce : vous pouvez manipuler le paramètre startOffset en JavaScript pour créer une animation différente.

Définition des propriétés CSS et des styles graphiques

L'option Eléments de style a été spécialement conçue pour le web. Les autres options relèvent davantage du traitement XML (avec XSLT, par exemple). Si vous ne prévoyez pas de modifier le code SVG après exportation, sélectionnez l'option Attributs de présentation par défaut.
Illustrator CC détecte les styles graphiques de votre illustration et les convertit en propriétés CSS en haut du document; vous pouvez ainsi définir aisément le style de votre fichier SVG par la suite.
Les noms des calques et des styles d'objet sont préservés.
J'ai défini deux styles graphiques : sandStyle (que j'ai déjà appliqué aux deux formes) et blueSky.
Dans les options d'exportation au format SVG, je sélectionne Eléments de style et Inclure les styles graphiques inutilisés. Les styles sandStyle et blueSky seront ainsi déclarés comme styles CSS dans le document SVG.
Voici le fichier SVG généré par Illustrator CC :

```
<style type="text/css"> .sandStyle{fill:#BFAF8F;stroke:#A6806A;stroke-width:3;stroke-miterlimit:10;} .blueSky{fill:#338AC4;stroke:#3469B7;stroke-width:3;stroke-miterlimit:10;} </style> <g id="mySquare"> <rect x="90.5" y="15.5" class="sandStyle" width="64" height="63"/> </g> <g id="myCircle"> <circle class="sandStyle" cx="42" cy="46" r="34.2"/> </g> v
```

Le code est bien structuré, et vous pouvez modifier aisément vos styles ou en appliquer de nouveaux de manière dynamique via JavaScript. Notez que les noms des calques mySquare et myCircle ont été préservés.

Michaël **Chaize**
Basé à Paris, Michaël est un spécialiste de Creative Cloud chez Adobe. Il travaille essentiellement sur le web, les terminaux mobiles et les jeux. Pour en savoir plus, consultez son blog à l'adresse creativedroplets.com

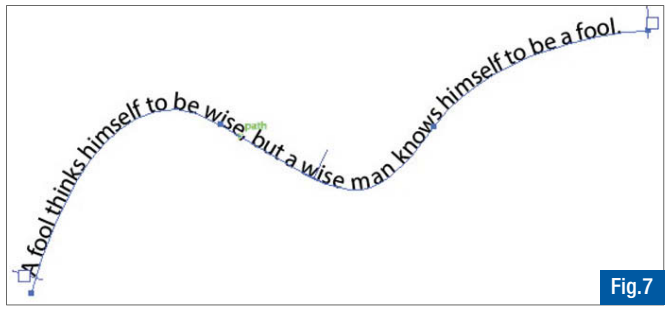


Fig.7

Utilisation de l'élément <textPath> pour le texte curviligne

Java Virtual Machine : optimisation et réactivité de la JVM 2e partie

Parmi les grands axes d'optimisation d'une application Java, il y a la bonne gestion de sa mémoire et du garbage collector. Pour rappel, lorsque celui-ci s'exécute, toute l'application peut être figée pendant quelques millisecondes (ou plus), le temps de nettoyer les références inutiles. Le second axe d'amélioration est d'optimiser les accès au monde extérieur à la JVM (notamment dans le cadre d'accès à la base de données et des requêtes SQL / HSQL ou autres entrées/sorties).

AU CŒUR DES OPTIMISATIONS DE CODE DE LA JVM

Le monitoring de la compilation dynamique

La compilation du bytecode en code assembleur est effectuée sur des portions de méthodes ou sur des méthodes entières si elles sont assez courtes et appelées très souvent.

Cette compilation prend du temps (ce qui peut être tracé) et de la place mémoire (qui peut également être tracée). Sur la JVM Oracle (et OpenJDK), il est possible d'afficher les temps de compilation et la taille du code compilé qui dépendent du compilateur C1, C2 ou Tiered (C1+C2).

L'option à ajouter au démarrage « **-XX:+CITime** »

Informations du -XX:+CITime avec le compilateur C1 (-client)

Accumulated compiler times (for compiled methods only)

```
Total compilation time    :  0.684 s
  Standard compilation    :  0.287 s, Average : 0.008
  On stack replacement    :  0.397 s, Average : 0.011
  Detailed C1 Timings
     Setup time:           0.000 s ( 0.0%)
     Build IR:             0.392 s (57.6%)
       Optimize:              0.006 s ( 0.8%)
     Emit LIR:             0.244 s (35.8%)
       LIR Gen:             0.111 s (16.3%)
       Linear Scan:         0.132 s (19.4%)
     LIR Schedule:         0.000 s ( 0.0%)
     Code Emission:        0.038 s ( 5.6%)
     Code Installation:    0.006 s ( 0.9%)
     Instruction Nodes: 217463 nodes

Total compiled bytecodes : 377139 bytes
  Standard compilation    : 147744 bytes
  On stack replacement    : 229395 bytes
Average compilation speed: 551622 bytes/s

nmethod code size         : 1375616 bytes
nmethod total size        : 4064232 bytes
```

Informations du -XX:+CITime avec le compilateur C2 (-server)

Accumulated compiler times (for compiled methods only)

```
Total compilation time    :  8.833 s
  Standard compilation    :  6.453 s, Average : 0.208
  On stack replacement    :  2.381 s, Average : 0.077

Total compiled bytecodes : 175772 bytes
  Standard compilation    :  98127 bytes
  On stack replacement    :  77645 bytes
Average compilation speed:  19898 bytes/s

nmethod code size         : 374432 bytes
nmethod total size        : 1469764 bytes
```

Pour obtenir des informations sur la compilation JIT, il faut utiliser l'option de la JVM suivante : -XX:+PrintCompilation

A noter que plus la version mineure du JDK augmente, plus les informations tracées sont précises.

Exemple :

```
  36  1    3    java.lang.Object::<init> (1 bytes)
  37  2    1    java.lang.reflect.Field::getName (5 bytes)
  37  3    3    java.lang.Number::<init> (5 bytes)
  39  1    3    java.lang.Object::<init> (1 bytes)   made not entrant
  40  7    1    java.lang.reflect.Method::getName (5 bytes)
  41  9    n 0 java.lang.invoke.MethodHandle::linkToStatic(LL)
L (native)    (static)
  91  54   s 3 java.io.ByteArrayInputStream::read (36 bytes)
  95  62   ! 3 java.io.BufferedReader::readLine (304 bytes)
 188 166 % 3 fib_nc::_c1 @ 106 (157 bytes)
```

Explications :

- 1ère colonne = temps en ms depuis le début du lancement de la JVM
- 2ème colonne = n° d'ordre des méthodes compilées
- 4ème colonne = que des chiffres de 0 à 4. Correspond au niveau de compilation (0 =native, 1 à 3 = C1, 4 = C2 -server)
- ! = code contenant des exceptions (qui elles aussi sont compilées)
- n = native
- s= synchronized
- % compilation de la méthode à la volée (OSR). Sur le libellé "188 166 % 3 fib_nc::_c1 @ 106 (157 bytes)", le @106 indique l'offset à partir duquel la méthode OSR a été compilée
- made zombie = méthode désoptimisée (en général en attente d'exceptions éventuelles)
- made not entrant = compilée une fois mais on ne peut plus lui faire appel (en général en attente d'exceptions éventuelles)
- uncommon trap = changement intervenu sur la méthode (ex: instrumentation JVMTI).

Il est possible d'avoir des informations aussi sur l'inlining et le temps de compilation de cette méthode précise en ajoutant : -XX:+UnlockDiagnosticVMOptions -XX:+PrintCompilation2.

```
  448   101  4  sqli.examples.TestHugeMethodv6::tresGrosseMethode
```

```
17643opcode (17643 bytes)
 3069   101 4 sqli.examples.TestHugeMethodv6::tresGrosseMethode
17643opcode (17643 bytes)   COMPILE SKIPPED: out of nodes before
split (retry at different tier)
 3069   101 time: 2621 inlined: 0 bytes

 119    71  3 java.util.HashMap::get (23 bytes)
 119    71  size: 1392(928) time: 0 inlined: 20 bytes
Pour l'OSR :
 80697    141 % 3 sqli.examples.TestHugeMethodv6::doMainLoops
 @ 717 (1422 bytes)
 80715    141 % size: 180688(104960) time: 17 inlined: 4371 bytes
```

L'inlining peut procurer un gain de temps de traitement important. Il est utile d'analyser son utilisation par la JVM via les 2 options suivantes : -XX:+UnlockDiagnosticVMOptions -XX:+PrintInlining.

Exemple :

```
 @ 1    java.lang.Object::<init> (1 bytes)
 @ 1    java.lang.Number::<init> (5 bytes)
 @ 66   java.lang.String::indexOfSupplementary (71 bytes)   too big
 @ 14   sun.misc.Unsafe::getObjectVolatile (0 bytes)   intrinsic
 @ 3    java.lang.String::indexOf (70 bytes)   callee is too large
 @ 15   java.lang.Double::<init> (10 bytes)   inline (hot)
```

Explications :

- La 1ère colonne avec le « @ » indique l'offset (en bytecode) de la méthode où on a mis en place l'inlining.
- Dans l'exemple ci-dessous, avec «@ 3 java.lang.String::indexOf » puis la ligne en dessous décalée @15, cela veut dire que l'on a fait un « inlining » sur 2 méthodes et qu'elles ont été fusionnées ensemble.
- inline (hot) : c'est quand une méthode ne dépasse pas les 35 octets de bytecode, elle est directement "inlinée".
- intrinsic : remplacement du bytecode par le code natif de la fonction (ex: pour les System.arrayCopy, les String.equals, la plupart des fonctions mathématiques).
- too big : méthode trop grande pour profiter d'un inlining.
- callee is too large : la méthode appelée est trop grande pour fusionner les 2 méthodes appelantes / appelées.

Globalement, en JDK 1.8 pour l'option -XX:+PrintInling, les explications sont plus nombreuses et axées sur la non réussite de l'inlining. En JDK 1.7, on indiquait plutôt les méthodes ayant réussi leur inlining et assez peu les cas d'échecs.

Pour les plus curieux, voici l'option pour voir le code assembleur généré :

Le code assembleur généré apparaît grâce aux options suivantes : -XX:+UnlockDiagnosticVMOptions et -XX:+PrintAssembly -XX:+DebugNonSafepoints

Il faut par contre ajouter une DLL dans le répertoire bin du JDK (hsdis-i386.dll pour un JDK 32 bit).

Exemple de résultat :

```
  0x02495e8a: adc ,0x3a8() ;*invokevirtual doubleValue
     ; - TestHugeMethod::moyenneMethode@1387 (line 1110)
  0x02495e90: mov $0x347e0778,    ;    {oop("445")}
  0x02495e95: xchg    %ax,%ax
  0x02495e97: call    0x0242d700 ; OopMap{ebp=Oop off=4316}
     ;*invokestatic valueOf
     ; - TestHugeMethod::moyenneMethode@1396 (line 1110)
     ;    {static_call}
```

Pour les autres JVM, côté IBM, il y a l'option -Xaot:verbose et -Xjit:verbose
Pour Jrockit, il y a l'option -Xverbose ou en plus complet : -Xverbose:opts,gen

Les limites de l'optimisation

Nombre d'exécutions d'une boucle avant compilation JIT :

- au bout de 1500 en mode -client
- au bout de 10000 en mode -server

Pour changer ce nombre côté Oracle, il faut ajouter l'option suivante au démarrage :
 -XX:CompileThreshold=xxx
Pour IBM, c'est -Xjit:count=xxxx et -Xaot:count=xxxx

Taille maximum du code source d'une méthode :

- 64Ko de code source pour la taille d'une seule méthode (ce chiffre pour une seule méthode peut paraître ridicule mais les compilateurs comme JSC de Rhino peuvent produire un code source java supérieur à 65ko sur une seule méthode selon la taille du fichier javascript à compiler).

Taille maximum en bytecode d'une méthode :

- 8Ko de bytecode : seuil de la JVM Oracle à ne pas dépasser, sinon il n'y a **pas de compilation**, on reste en interprété contournable par l'option -XX:-DontCompileHugeMethods en Oracle ou passer à IBM Java 7 (qui n'a pas de limite de taille).
- 4870 octets de bytecode : seuil de la JVM 1.6 d'IBM où toute méthode en dessous de cette taille a un gain significatif (jusqu'à 10 fois) de la vitesse d'exécution.
- 4Ko de bytecode : seuil de la JVM JRockit en dessous duquel la vitesse est significativement plus élevée.
- 2Ko de bytecode : seuil en dessous duquel j'ai constaté sur la JVM 1.7 d'Oracle que toute méthode répond beaucoup plus vite que dans l'intervalle]2Ko, 8Ko[
- 1000 octets de bytecode : seuil au-delà duquel il n'y a **pas d'inlining direct** de la méthode (seulement une partielle) Contournable par l'option -XX:InlineSmallCode=xxxx
- 35 octets de bytecode. Seuil au-delà duquel il n'y a pas de **full inlining** contournable par l'option -XX:MaxInlineSize=xxxx.

Un outil minimaliste pour calculer la taille des méthodes d'une classe

Pour déterminer si on dépasse un seuil important 8Ko, 4Ko, 1Ko ou 35 octets de bytecode, on a l'outil de base via javap (du JDK) qui permet de décompiler un fichier .class et de connaître la taille de chaque méthode au prix d'une analyse non intuitive.
Exemple avec javap -c MaClasseXxx.class

```
public static void loopFib(long);
  Code:
     0: lconst_0
    ...
    16: lstore_2
    17: goto          2
    20: return
```

Ici, la dernière ligne de cette méthode contient "20", c'est la taille en bytecode de la méthode.
Voici l'outil minimaliste qui permet d'analyser la taille des méthodes potentielles lors de l'intégration continue; pour rester concis, cet article ne contient que le squelette de code pour scruter une classe entière en se basant sur l'api ASM pour l'analyse du bytecode.

```
import org.objectweb.asm.ClassReader;
import org.objectweb.asm.MethodVisitor;
import org.objectweb.asm.Opcodes;
import org.objectweb.asm.Type;
import org.objectweb.asm.commons.CodeSizeEvaluator;
import org.objectweb.asm.tree.AbstractInsnNode;
import org.objectweb.asm.tree.ClassNode;
import org.objectweb.asm.tree.MethodNode;
import org.objectweb.asm.tree.analysis.Frame;

…
      File maClasse = new File("MaclasseDeTest.class");
      ClassReader cr = new ClassReader(new FileInputStream
(maClasse));
      ClassNode cn = new ClassNode();
      cr.accept(cn, ClassReader.SKIP_DEBUG);
      List<MethodNode> methods = cn.methods;
      for (int i = 0; i < methods.size(); ++i) {
        MethodNode method = methods.get(i);
        if (method.instructions.size() > 0) {
          try {
            MethodNode m1 = new MethodNode();
            m1.access = method.access;
            m1.tryCatchBlocks = method.tryCatchBlocks;
            CodeSizeEvaluator mv2 = new CodeSizeEvaluator(m1);
            method.accept(mv2);
            int tailleByteCode = mv2.getMinSize();
            // TODO faire quelque chose avec la taille
            // trouvée en bytecode de la méthode...
          } catch (Exception e) {
            log.error(e);
          }
        }
      } // fin du FOR chq méthode
```

TESTS : RELATION ENTRE LA TAILLE DES MÉTHODES ET LE COMPORTEMENT DE LA JVM

Test sur l'influence de la taille des méthodes

Voici un test contenant de mauvaises pratiques sur la taille des méthodes. Le code possède volontairement une initialisation de tableau très longue (remplie de « Double.valueOf("445"), ... ») dans les méthodes à tester.

```
public static void tresGrosseMethode17643opcode() {
  double[] testBidon = new double[] {
    Double.valueOf("445"), Double.valueOf("445"),
    ….. } ;
    double sum = 0.0d;
    final int taille = 100;
    for (int i = 0; i < 10000; i++) {
      for (int idx = 0; idx < taille; idx++) {
        sum += testBidon[idx];
        sum /= 1.00001d;
      }
    }
}
```

Bilan sur le test de vitesse d'exécution selon la taille des méthodes :

- Plus les JVM sont récentes (java 7 et +), plus le gain en réduisant la taille des méthodes est perceptible.
- La taille idéale d'une méthode semble être vers les 80 à 300 octets de bytecode qui est difficilement traduisible en nombre de lignes mais disons autour des 20 à 30 lignes de code. Cependant, sans JVM java 7 ou 8 en -server, aucune amélioration de vitesse ne sera constatable.
- Le découpage à l'extrême des méthodes en sous-méthodes plus petites peut donner de mauvais résultats surtout avant compilation et optimisation globale de l'ensemble. Après une phase de « warm up » et lorsque la JVM le permet, les temps de réponse sont optimisés.
- Oracle Java 6 aussi bien 32bits que 64bits n'arrive pas à optimiser des méthodes de taille intermédiaire (entre 1 et 7999 octets de bytecode)
- L'option « -XX:DesiredLimitMethod=xxxx» activable uniquement en JDK 1.6 n'a aucune influence directe sur la rapidité d'exécution des méthodes contrairement à ce que l'option laisse supposer.
- Pour Oracle Java 7, entre la version -client et -server, il y a une différence de vitesse de 770 en faveur du mode -server. De même, Java 7 (en mode -server) a des performances bien meilleures que Java 6.
- Sur la JVM Oracle, les paramètres de tuning de l'inlining suivants semblent inefficaces : -XX:-ClipInlining, -XX:InlineSmallCode=xxx et -XX:MaxInlineSize=xxx et n'ont aucune influence directe.
- De même, l'inlining s'arrête lorsque la limite de « DesiredLimitMethod » (8Ko) de bytecode est atteinte. La JVM parle de « clipping » de l'inlining. En conséquence, si les méthodes fusionnées arrivent au total à moins de 8Ko, elles seront plus rapides que lorsqu'on a atteint la limite des 8Ko et qu'aucun inlining ne peut plus arriver en fin de méthode appelante.
- Depuis la version 7 (et supérieure), le paramètre « DesiredLimitMethod est en dur dans globals.hpp du code source du JDK (voir ligne 3553).
- L'option de compilation « -XX :TieredCompilation et -XX :TieredStopAtLevel=1 à 4 » est dépendante du -client ou -server. Globalement les performances sont meilleures avec le niveau 4 (qui est la valeur par défaut).
- Sur la JVM java 7 d'Oracle, une amélioration de vitesse se fait sentir lorsque celle-ci est en dessous du seuil de 2Ko de bytecode (jusqu'à 3 à 4 fois plus rapide par rapport à des méthodes comprises entre 2Ko et 8Ko) et encore plus vers 300 à 80 octets de bytecode.
- Sur la JVM 1.6 d'IBM (J9) en 32bits, dès que la taille d'une méthode arrive en dessous du seuil de 4870 octets de bytecode, la méthode voit sa vitesse s'améliorer d'un facteur 54 minimums (268 fois après un warm up sur le test effectué).
- Sur la JVM 1.6 d'IBM (J9), le « warm up » d'une méthode fortement découpée peut être lent mais une fois en place, la méthode peut être jusqu'à plus de 1000 fois plus rapide que si la méthode dépasse 4870o de bytecode.
- Sur la JVM IBM en java 7 64bit, il n'y a pas de limitation de taille sur les méthodes à compiler en code assembleur. Les méthodes sont plus rapides qu'en Java 6 IBM. De même que pour Java 7 Oracle (en -server), en dessous de 2Ko de bytecode, les méthodes sont encore plus rapides.
- La JVM Jrockit java 6 32bit a également un seuil à 4Ko de bytecode où lorsqu'une méthode est en dessous de quelques octets de code, elle voit sa vitesse augmenter par 10 minimum.

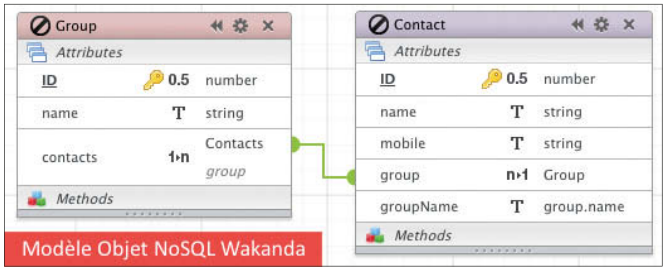
La suite dans Programmez! n°179

Olivier **Guilloux** - *SQLi*

Développer des Applications Web avec AngularJS et WakandaDB

2e partie

Dans la 1ère partie, nous avons créé, en Server-Side JavaScript, un modèle avec des Classes d'objets persistants (DataClasses) dans la base NoSQL WakandaDB, et vu les premières bases d'AngularJS. Nous allons maintenant créer l'interface graphique de l'application.



Retrouvez le code des 1ère et 2ème parties de cet article sur https://github.com/Wakanda/NG-Wakanda-Pack

Single-Page Application

Commençons par préparer notre contrôleur pour y gérer une navigation légère du type de celles que l'on retrouve dans les interfaces mobiles.

```
function Controller($scope, $wakanda) {
  $scope.panels = ['Groups', 'Contacts', 'Contact infos', 'Edit Contact'];
  $scope.panel = 0;
  $scope.back = function () { $scope.panel -= 1; };
$scope.next = function () { $scope.panel += 1; };
}
```

Nous créons ensuite un en-tête avec titre et boutons de navigation.

```
<table><tr>
  <th> <button ng-click="back()" ng-show="panel > 0">
    &lt; {{panels[panel - 1]}}
  </button></th>
  <th><!--Title--><h1>{{panels[panel]}}</h1></th>
  <th><!--Create/Add/Modify Button--> 
    <button ng-click="createGroup()"
            ng-show="panel === 0">Create</button>
    <button ng-click="addContact()"
            ng-show="panel === 1">Add</button>
    <button ng-click="next()"
            ng-show="panel === 2">Modify</button>
  </th>
</tr></table>
```

Comme pour ses propriétés, les méthodes de $scope sont disponible depuis la vue. On les appelle ici depuis « ng-click ». Des expressions sont utilisées pour retrouver le titre des pages courante et précédente, ainsi que pour fixer sur « ng-show » les règles d'affichage des boutons.

Gestion des Groupes de Contact

Nous allons ajouter à notre contrôleur les appels à l'API du service $wakanda. Dès que le service est prêt, « ds » fournit toutes les DataClasses du Modèle du serveur. On peut alors :

- récupérer les objets « Group » avec « $find() »;
- créer des objets localement avec « $create() »;
- et les sauver sur le serveur avec « $save() ».

$find() retourne des collections sous forme de tableaux JavaScript enrichis (on peut donc utiliser leur méthode push()).

La méthode setCurrent() est chargée de définir un « group » ou un « contact » courant, et d'en afficher le détail sur le panel suivant.

```
$wakanda.init().then(function (ds) {
  $scope.groups = ds.Group.$find();
  $scope.createGroup = function () {
    var name = prompt("Choose the name of your new group");
    if (!name) return;
    var group = ds.Group.$create({name: name});
    group.$save();
    $scope.groups.push(group);
  };
  // Select an entity from a row
  $scope.setCurrent = function (name, value) {
    $scope[name] = value;
    $scope.next();
  };
  // Load the contacts when the current group change
  $scope.$watch('group', function (group) {
    if (!group) {
      $scope.contacts = ds.Contact.$find();
    } else {
      group.contacts.$fetch();
      $scope.contacts = group.contacts;
    }
  });
  // Unselect current contact when the contact list change
  $scope.$watchCollection('group.contacts[0]', function(contact) {
    if ($scope.panel < 3) $scope.contact = null;
  });
});
```

La méthode Angular « $watch() » détecte lorsque le groupe sélectionné « $scope.group » est modifié. On charge alors ses contacts associés avec la méthode wakanda « $fetch() ». « $watchCollection() » détecte quand une propriété d'objet ou un élément de tableau change. Nous nous en servons ici pour réinitialiser le contact sélectionné quand la liste est mise à jour. Nous pouvons maintenant préparer le premier panel :

```
<div ng-show="panel === 0">
  <table><tbody>
    <tr class="{{(!group) && 'info'}}">
      <td ng-click="setCurrent('group')">All Contacts</td>
    </tr>
    <tr ng-repeat="g in groups" class="{{(g === group) && 'info'}}">
      <td ng-click="setCurrent('group', g)">{{g.name}}</td>
    </tr>
```

```
    <tbody></table>
  </div>
```

L'utilisateur pourra depuis celui-ci:
- créer des groupes (depuis le bouton de l'entête),
- lister les groupes enregistrés,
- sélectionner un groupe pour voir ses contacts
- aller voir la liste non filtrée de tous les contacts

« ng-repeat » parcours et affiche les groupes chargés. « g » représente le groupe courant. Une classe CSS « info » est appliquée pour identifier le groupe sélectionné, ou à défaut, la ligne « All Contacts ».

Infinite Scroll

Le protocole de Wakanda limite le nombre maximum d'objets d'une collection retourné par requête. Il crée un « entityset » temporaire pour récupérer les suivants. Nous allons utiliser une petite snippet proposée par Vojta Jina de l'équipe d'AngularJS pour gérer le scroll vertical (http://jsfiddle.net/vojtajina/U7Bz9/). En y associant la méthode « $more() » des collections de $wakanda, on obtient un live scrolling fluide quelle que soit la complexité de la requête initiale ou le nombre de résultats.

```
<div ng-show="panel === 1">
  <h2>{{group.name || "All Contacts"}} ({{contacts.$totalCount}})</h2>
  <table><tbody when-scrolled="contacts.$more()">
    <tr ng-repeat="c in contacts"
        class="{{c === contact) && 'info'}}">
      <td ng-click="setCurrent('contact', c)">{{c.name}}</td>
    </tr>
  <tbody></table>
</div>
```

« $totalCount » permet d'afficher le nombre d'objets de la collection disponibles sur le serveur.

Alias

Nous allons maintenant créer le troisième panel pour afficher le détail du contact sélectionné.

```
  <div ng-show="panel === 2">
    <div>
      <h2>{{contact.name}}</h2><br>
      <label>Group</label> {{contact.groupName}}<br>
      <label>Mobile</label> {{contact.salary}}
    </div>
  </div>
```

Chaque contact a un attribut alias « groupName ». Les alias sont très puissants et permettent de mettre à disposition immédiate des données liées passant par un ou plusieurs attributs relationnels successifs, ou encore de faire abstraction des DataClasses intermédiaires de relations many-to-many.

Retrouvez un exemple concret sur :
http://slideshare.net/alexandre_morgaut/wakanda-nosql-for-modeldriven-web-applications/34

De la lecture à l'écriture

Lire ou créer des objets « stockables » sur un serveur c'est bien, mais il faut également pourvoir les modifier, voire les supprimer. C'est ce que nous allons permettre de faire dans le dernier panel. Nous ajoutons les méthodes « addContact() » et « saveContact » dans notre contrôleur :

```
$scope.addContact = function(contact) {
  $scope.contact = ds.Contact.$create({group: $scope.group});
  $scope.panel = 3;
};
$scope.saveContact = function() {
  $scope.contact.$save();
  // if new, add to list
  var contacts = $scope.contacts;
  if (contacts.indexOf($scope.contact) !== -1) return;
  contacts.push($scope.contact);
  $scope.back();
};
```

La 1ère, associée au bouton « Add » dans la liste des contacts, permet de fixer le contact sélectionné à une nouvelle instance avant de passer sur le panel d'édition. La 2ème permet de sauver le contact sur le serveur avant de passer sur le panel de lecture détaillée.

```
<div ng-show="panel === 3">
  <h2>{{group.name}}</h2>
  <form ng-submit="saveContact()">
    <label>name</label> <input ng-model="contact.name" required><br>
    <label>Group</label>
    <select ng-model="contact.group"
            ng-options="c.name for c in groups" required>
      <option value="">-- select a group --</option>
    </select><br>
    <label>mobile</label> <input ng-model="contact.mobile">
    <input type="submit">
  </form>
</div>
```

Angular, avec « ng-submit », va prendre le contrôle de ce formulaire HTML5. Il appellera bien sûr la méthode « saveContact() » au moment opportun, mais appliquera également les contraintes HTML5 comme « required » quel que soit le navigateur. « ng-model » permet ici de lier les champs de formulaire aux valeurs associées du scope courant.
« ng-options » va créer dans la balise <select> un élément <option> pour chaque valeur d'une collection. Sa syntaxe permet de lier des objets par référence, avec ou sans ID, et de choisir la propriété à afficher comme label.

Aller plus loin

Angular et Wakanda s'associent de manière efficace pour développer des applications Web. Le service angular-wakanda s'intègre très naturellement et fait presque oublier que l'on travaille sur des données relationnelles distantes. $login() et $logout() vous permettront ensuite de gérer les permissions quel que soit l'annuaire d'utilisateurs utilisé. Vous apprécierez également les avantages des « calculated attributes », « restricting queries », ou des DataClass étendues. Et bonus, vous aurez bientôt un générateur yeoman « angular-wakanda »

Ressources

https://angularjs.org
http://yeoman.io http://wakanda.org
https://github.com/Wakanda/NG-Wakanda-Pack

Alexandre **Morgaut**
Community Manager chez 4D
https://github.com/AMorgaut

Timeline : 1957

Objet : 8 traitres volent l'invention qui engendrera les processeurs

Plus qu'une affaire de science, l'histoire de la mise au point et de l'évolution des processeurs repose sur un business implacable où tous les coups bas sont gagnants.

Les Huit Traitres sont, de gauche à droite : Gordon Moore, Sheldon Roberts, Eugène Kleiner, Robert Noyce, Victor Grinich, Julius Blank, Jean Hoerni et Jay Last.

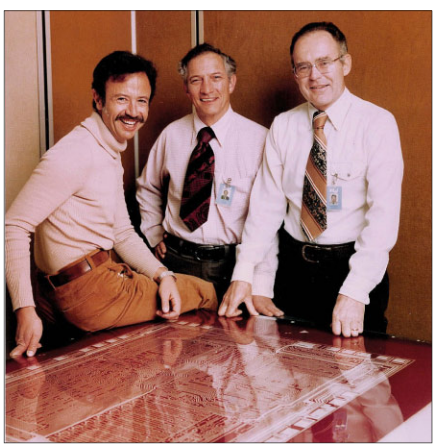
Si les premières machines programmables datent de la seconde guerre mondiale, ce n'est qu'au début des années 50, grâce au projet de calculateur BINIAC de l'US Air Force, que l'on comprendra que le moyen le plus prometteur de faire fonctionner un ordinateur serait de le faire reposer sur le système binaire (bits 0 et 1). Et William Shockey, qui vient justement d'inventer le transistor, est persuadé qu'il dispose de la technologie pour implémenter le système binaire sous la forme de composants électroniques. Enthousiaste mais naïf, il recrute huit anciens collègues des Bell Labs pour lancer avec lui l'aventure Shockley Semiconductor Laboratory en Californie, entreprise qui aurait dû devenir le premier fabricant au monde de microprocesseurs. Sauf que les huit collègues, un tantinet véreux, n'ont aucune confiance dans les capacités de businessman de ce savant fou de William Shockey. Sitôt arrivés en Californie, ceux que l'on appellera par la suite « les Huit Traitres » empochent les brevets et tous les plans techniques du transistor, puis partent vendre le tout à l'homme d'affaires Sherman Fairchild. Celui-ci était alors connu pour avoir de nombreux contacts avec les hauts gradés de l'armée. Fairchild est séduit. Au point d'embaucher les Huit Traitres pour les mettre à la tête d'une nouvelle filiale dédiée aux composants électroniques, la Fairchild Semiconductor.

Fairchild invente la puce sur silicium

En 1958, soit un an après la création de l'entreprise, l'un des Huit, le suisse francophone Jean Hoerni, invente le procédé de gravure des transistors sur un wafer de silicium, permettant de fait de fondre en une fois tout un circuit électronique. IBM est le premier à acheter des droits pour cette technique, grâce à laquelle il devient possible de fabriquer des composants électroniques très peu chers, et lance la première usine de puces électroniques au monde en 1960, à New York. Mais le premier gros client de Fairchild Semiconductor est la NASA. Celle-ci lui commande au début des années 60 des puces électroniques pour équiper l'ordinateur de bord de ses missions lunaires Apollo. À ce stade, Fairchild Semiconductor voit son chiffre d'affaires doubler tous les ans. Mais la maison mère de Sherman Fairchild dilapide les bénéfices. Excédés par ce manque patent d'ambition, deux membres des Huit claquent la porte et emmènent avec eux tout le savoir- faire, en particulier la technologie MOS Silicon Gate technology juste inventée par une nouvelle recrue, pour l'exploiter dans une nouvelle entreprise. Ces deux membres sont Robert Noyce et de Gordon Moore. Nous sommes en 1968 et ils fondent Intel.

Et Intel lance le premier microprocesseur

Tandis qu'Intel passe ses premières années à fabriquer des composants mémoires, IBM progresse de son côté sur le design des processeurs, c'est-à-dire des composants électroniques qui ne sont pas juste capables de commuter des signaux, mais qui comportent suffisamment de transistors pour exécuter dans une seule puce des opérations complexes. De ces travaux nait le concept de jeu d'instructions : une suite de bits simple déclenche une séquence de plusieurs opérations. Fort des avancées qu'IBM lui communique (Andy Grove, qui assurait les liens entre IBM et Fairchild, a été le premier à être débauché par Noyce et Moore), Intel met au point en 1971 le premier composant électronique qui corresponde à un processeur, le 4004, pour l'industrie des calculatrices. Celui-ci sera suivi en 1972 par le 8008, un processeur avec un jeu d'instruction plus élaboré et qui sert encore aujourd'hui de colonne vertébrale aux processeurs Core et Xeon d'Intel. Le jeu d'instructions du 8008 est en fait une adaptation de celui mis en place dans les terminaux DataPoint 2200 dont on se sert à cette époque pour manipuler les grands systèmes.



Andy Grove, Robert Noyce et Gordon Moore s'enfuient de Fairchild en 1968 pour fonder Intel.

Le processeur pour les grands systèmes, le microprocesseur pour les jeux vidéo

Jusque dans les années 80, les industriels de l'informatique - IBM, HP et Dec, principalement - construisent des ordinateurs haut de gamme (mainframes et minis) dont le processeur est composé de plusieurs puces électroniques (les Dec PDP-11, puis VAX, les HP3000 et HP9000, ainsi que les IBM S/360, puis S/370). En effet, Il n'est pas encore possible à ce moment de graver des puces avec suffisamment de transistors pour supporter les instructions 16 et 32 bits, et les plages mémoire 24 à 32, voire 36 bits, de ces machines.

Alors, en attendant l'arrivée des premiers micro-ordinateurs domestiques, c'est le marché des jeux vidéo qui s'empare des processeurs en une puce - ou microprocesseurs, à l'époque. La demande est telle qu'Intel voit rapidement fleurir des concurrents. En 1974, alors qu'Intel vient de lancer son 8080 - un 8 bits à 2 MHz, 10 fois plus rapide que le 8008 et capable d'adresser 64 Ko de RAM - le fabricant de téléphones Motorola

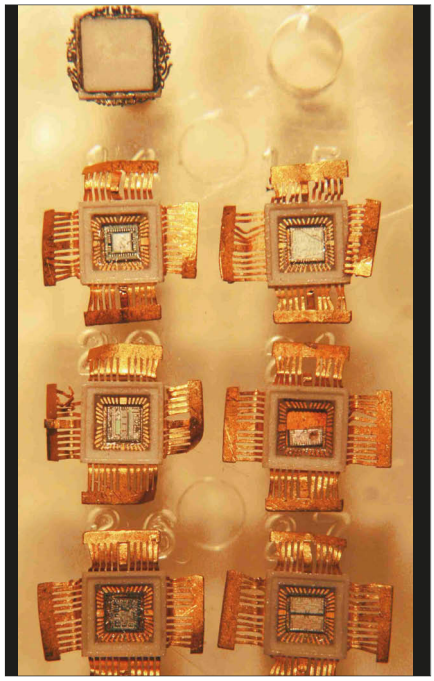
propose son MC6800, aux caractéristiques similaires, mais avec un jeu d'instructions inspiré du DEC PDP-11.

Gordon Moore invente sa loi

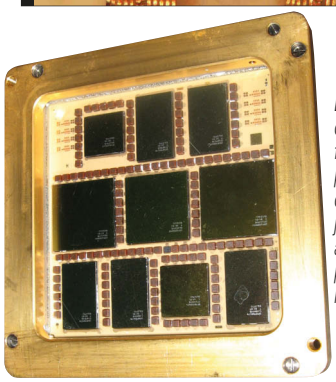
C'est en 1975, lors de l'édition annuelle de l'EEE International Electron Devices Meeting, que Gordon Moore expose sa fameuse prédiction, qui fera ensuite office de loi : tous les deux ans, les composants électroniques doivent voir leur nombre de transistors doubler. En réalité, les observateurs de l'époque comprennent que cette prédiction est surtout une loi d'équilibre économique : pour continuer à vendre des processeurs au fil du temps, il faudra les améliorer. Or, sachant que l'on pourra économiquement optimiser les processus optiques tous les deux ans - le système de gravure des processeurs d'Hoerni consiste à projeter l'image d'un circuit au travers d'une loupe pour la graver en miniature - il suffira donc de multiplier par deux des transistors dont on peut réduire la taille par deux, afin que les processeurs soient toujours gravés sur un rectangle de silicium dont la taille ne varie pas.

Le succès éphémère des clones

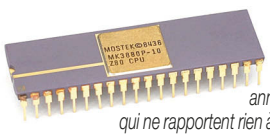
En 1975, le chef de projet du 8080, Federico Faggin, quitte Intel pour fonder Zilog et vendre une version optimisée du 8080, le Z80. Il se passe la même chose chez Motorola : l'équipe en charge du 6800 ne supporte pas que sa direction l'empêche de plancher sur une déclinaison à bas coût du processeur et part chez MOS Technology (créée par des anciens de Texas Instrument pour fabriquer des puces de calculatrices) pour produire le 6502. Les 6502 et Z80 reprennent chacun le jeu d'instruction de leur aîné, mais coûtent dix fois moins cher, soit environ 30$ au lieu de 300$. De fait, ce sont surtout ces modèles-là que choisissent les premiers fabricants de micro-ordinateurs : les américains (Apple, Commodore...) prennent le 6502, tandis que les européens optent en majorité pour le Z80. Le succès du Z80 en dehors des États-Unis s'explique surtout par le fait que Zilog a distribué assez librement des licences de sa puce, laissant des entreprises européennes et asiatiques en fabriquer des copies pour fournir leurs marchés locaux. Si bien que Zilog ne sera pas le fabricant de plus de la moitié des Z80 vendus. Du côté du 6502, les caisses ne se remplissent pas non plus : MOS Technology est rachetée en 1976 par Commodore, qui redoutait la pénurie de puces pour ses calculatrices, et n'investit que peu dans la R&D des futurs composants. De fait, en 1979, seuls Motorola et Intel ont les moyens de lancer des versions 16 bits de leurs microprocesseurs. Il s'agit respectivement des 68000 et 8086, lesquels équiperont respectivement les Mac et les PC.



Le processeur de l'IBM ACS-360 était composé de plusieurs puces pour cumuler suffisamment de transistors.



La multiplication des puces pour former un seul processeur perdurera chez IBM jusque dans les années 2000 pour sa gamme mainframe.



Plus de la moitié des microprocesseurs Z80 vendus dans les années 80 sont des copies qui ne rapportent rien à son inventeur, Zilog.

Alors qu'il équipe des best-sellers comme l'Apple II ou la Nintendo NES, le processeur 6502 n'évoluera pas car les bénéfices de ses ventes sont réinjectés dans la mise au point des ordinateurs de Commodore.

La multiplication des processeurs condamnée par la loi de Moore

C'est entre 1991 et 1992, qu'IBM, HP et Dec équipent enfin leurs ordinateurs Unix et grands systèmes de processeurs en une puce, lorsque les usines sont suffisamment modernisées pour atteindre la finesse de gravure de 0,8 µm, soit 800 nm, ce qui autorisait la disposition de leurs 600 000 à 1 million de transistors nécessaires sur un seul rectangle de silicium. Ces processeurs, l'IBM Power-1 version RSC, le HP PA-Risc 7000 et le Dec Alpha 21064 ont des designs beaucoup plus poussés que ceux d'Intel ou de Motorola (beaucoup plus de registres, notamment, pour simuler des fonctions de haut niveau sans avoir à les implémenter sous forme de circuits supplémentaires : c'est le principe dit de RISC). Et, bien sûr, ils ont leur propre jeu d'instruction, plus ou moins hérité des processeurs à plusieurs puces qu'ils développent depuis les années 70. Mais voilà. En s'obligeant à suivre la loi de Moore, tous les industriels des processeurs parviennent en 1995 à la finesse de gravure de 0,35 µm, ce qui les oblige désormais à peupler toujours le même rectangle de silicium de 4 à 5 millions de transistors. Dès lors, le Pentium Pro d'Intel destiné aux PC a de fait exactement les mêmes fonctions et la même puissance que les processeurs destinés aux grands systèmes. Mais il s'en vend plus et, donc, il est bientôt le seul à générer suffisamment de revenus pour investir dans la génération suivante. A partir de ce moment, toute l'évolution des processeurs ne sera plus qu'une affaire d'usines : face à Intel qui est le seul à rentabiliser son propre parc industriel, les autres fabricants de puces devront être toujours plus nombreux pour investir dans des usines qu'ils se partagent. Dommage collatéral : tous les jeux d'instructions disparaîtront au fur et à mesure au profit de celui du x86, celui de Gordon Moore, inventeur de la loi et membre des Huit Traitres.

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 : Yann Serra, F. Bordage, J. Chatard
Secrétaire de rédaction : Olivier Pavie
Experts : K. Alnijres, A. Lemar-Verrier, F. Dupont, J. de Olivereira, F. Bellahcene, N. De Loof, S. Nichele, L. Cros, T. Larbi, V. Kovalsky David, C. Villeneuve, J. Allèbe, N. Loye, L. Prada, M. Grosrenaud, J. de Cuyper, J. Guittard, S. Escandell, B. Moulès, P. Lopez, P-A Marendat, I. Khan, M. Chaize, O. Guilloux, A. Morgaut, CommitStrip

Crédits couverture : © 02-14-07 © Inok

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

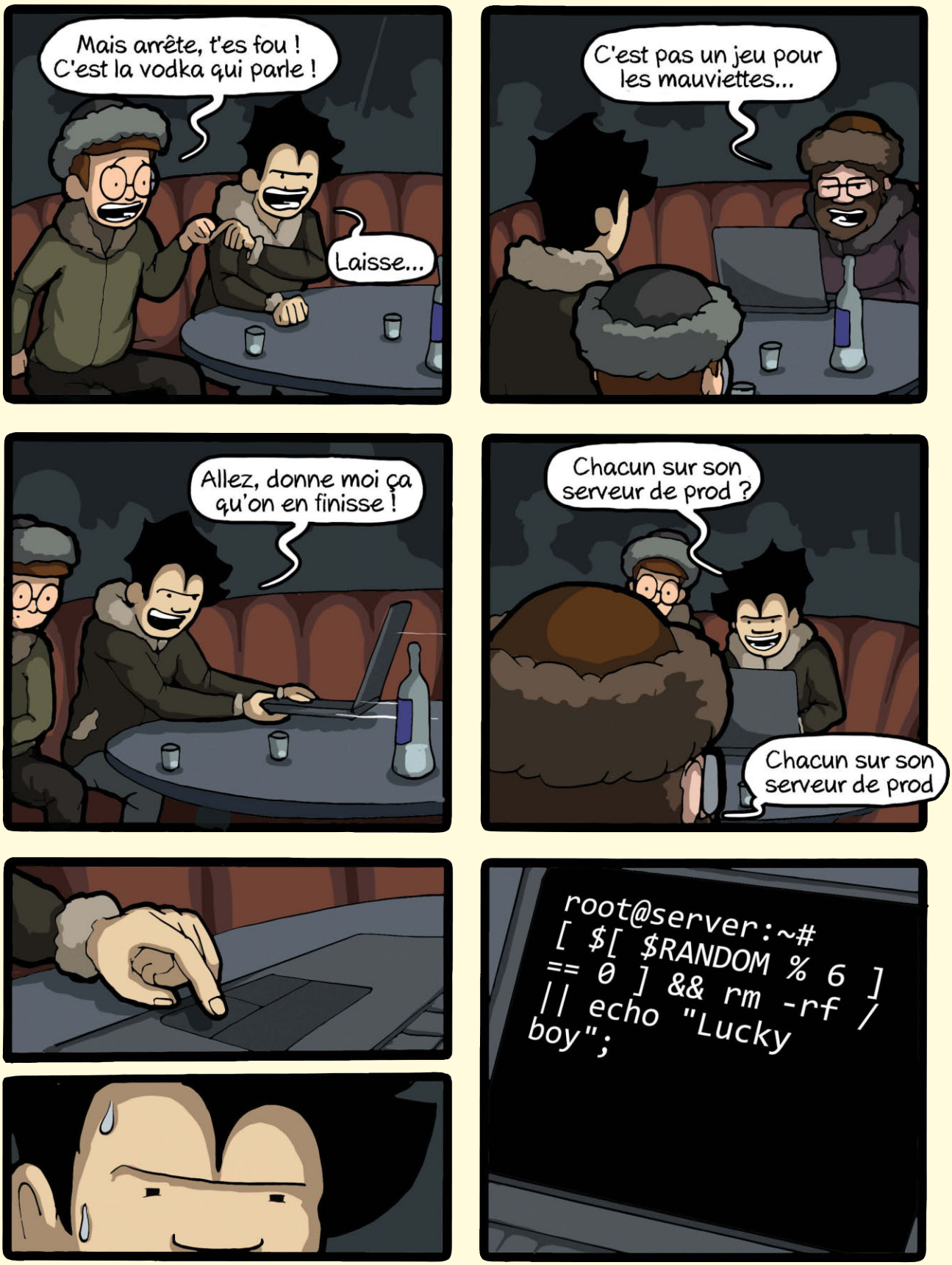
Responsable titre : Terry MATTARD
Téléphone : 0967320934

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

© NEFER-IT / Programmez, septembre 2014
Toute reproduction intégrale ou partielle est interdite sans accord des auteurs et du directeur de la publication.

Chaque mois, retrouvez CommitStrip dans **Pro**grammez!



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

Quelle interopérabilité entre mes différents fournisseurs Cloud ?

Avec Aruba Cloud,

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

OFFRE SPECIALE *2 fois plus pour le même prix!***

- **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.

***Pour tout premier versement lors de votre première création de compte entre le 01/10/2014 et le 30/11/2014, sous la forme d'un coupon émis sous quinzaine d'un montant égal au premier montant versé.*