PROGRAMMEZ

PROGRAMMEZ!

le magazine du développeur

www.programmez.com

Quel futur pour les langages de programmation ?

C++, C#, VB, PHP, Méta-programmation...

12-19-13 © NABIHARIA!

La guerre des moteurs 3D

Unity3D vs UDK vs CryEngine3 vs Anarchy

Carrière

Devenez votre propre patron !

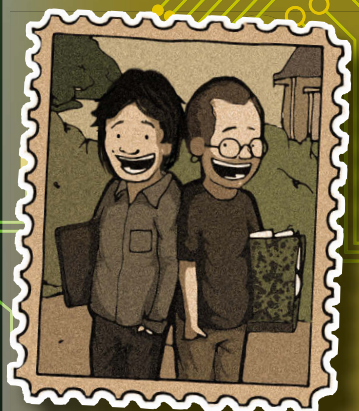
Web

Créer un site «one-page»

Les pièges du référencement

Créer une carte interactive en HTML 5

CommitStrip :



les coulisses d'un succès

Node.JS

Google Maps

Mensuel n°173 - Avril 2014

M 04319 - 173 - F: 5,95 € - RD

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

WINDEV 19 : DÉVELOPPEZ 10 FOIS PLUS VITE POUR WINDOWS, IOS, ANDROID, WEB...

VOUS AVEZ UN EXISTANT WINDEV ?

PASSEZ LE SUR MOBILE EN QUELQUES HEURES !

Les applications **WINDEV** passent très rapidement sur mobile: retaillez les fenêtres pour les adapter à la taille des mobiles, supprimez les traitements qui ne sont pas nécessaires sur mobile, adaptez un peu le code, et hop, vous voilà en possession d'une super application mobile !

La portabilité entre **WINDEV** et **WINDEV Mobile** vous permet de disposer d'applications mobiles très performantes en un délai record.

Là où vos concurrents qui n'ont pas choisis les bons outils de développement doivent re-développer, **vous ré-utilisez intelligemment votre existant** !

Vous gagnez en temps, en qualité, en fonctionnalités et en budgets: **bravo** !

Applications natives

iOS

Applications natives

TOUT EST COMPATIBLE

Avec WINDEV, tout est compatible: le code bien entendu, mais également les fenêtres, les états, les requêtes, les bases de données, les analyses...

Depuis le mobile, vous accédez aux données soit en local, soit à travers le SI de l'entreprise, soit via le cloud: tout est facile.

WINDEV 19 est également compatible avec WEBDEV 19: transformez vos applications en sites **INTERNET** et **INTRANET**

Fournisseur Officiel de la Préparation Olympique

Siège PC SOFT 3 rue de Puech Villa - 34197 MONTPELLIER Cedex 05 **Tel. 04 67 032 032**
Agence 142 Avenue des Champs Elysées 75008 PARIS Tél. 01 48 01 48 88

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

édito

Le langage informatique est comme le serpent, il change de peau

Pour les non-initiés, l'informatique c'est un ordinateur, un tableau, un smartphone, des logiciels. Le reste ce n'est pas leur problème. Tout développeur aguerri sait que la programmation est un élément mouvant, évoluant régulièrement. Mais fondamentalement, les paradigmes n'ont pas changé depuis plus 30 ans : interface, orienté objet, etc.

Mais, le langage de programmation n'est pas un gaz inerte. Il est obligé de s'adapter aux nouvelles contraintes, de rajouter de nouvelles fonctions. Au risque de proposer des milliers d'API, de librairies, de frameworks pour étendre le langage et d'arriver à une véritable indigestion. Mais nous voyons aussi les limites de certains langages sur des utilisations et optimisations particulières. Ainsi, le parallélisme (= la capacité à monter en charge et à utiliser plusieurs processeurs et plusieurs cœurs) reste le point faible des langages actuels. Pour pallier ce problème, on nous rajoute des fonctions, des mots-clés. Ce n'est qu'une rustine.

Un langage non fixé dans ses spécifications peut aussi devenir un standard de facto. HTML 5 en est le parfait exemple. La recommandation officielle sera disponible au 4e trimestre 2014. C'est seulement à ce moment-là que l'on pourra dire que HTML 5 est officiellement disponible... La version 5.1, version mineure, devrait être disponible fin 2016. Mais HTML 5 a été rapidement adapté malgré les aléas sur certaines fonctions non stabilisées. Ce qui a précipité la chute de Flash d'Adobe.

Une autre question, plus ou moins tabou en France, faut-il continuer à privilégier l'enseignement de la programmation objet ? Ne faudrait-il pas casser cette « pensée unique » ? Trois experts échangent leurs arguments et contre-arguments. A se demander si nous ne ratons pas quelque chose...

Ce mois-ci, beaucoup de choses pour vos claviers et IDE : les moteurs 3D, Node.JS, HTML 5, ASP.Net, le développement pour tablettes.

Sans vous pas de magazine

Nous souhaitons remercier tous les partenaires technologiques qui contribuent régulièrement à Programmez ! : DZMob (développement du nouveau site web), InfiniteSquare, Cellanza, les laboratoires Epitech, Xebia, Osaxis, Itelios, Sfeir, Palo-IT... Nous remercions vivement tous nos auteurs / contributeurs qui chaque mois fournissent de nombreux articles. Sans eux, il serait impossible de publier le magazine et de vous proposer une si grande variété de thèmes et de langages. Sans oublier l'équipe de Programmez ! qui travaille tous les jours sur le magazine et le site internet. Nous tenons aussi à vous remercier, vous, lectrices et lecteurs de Programmez ! Certains nous sont fidèles depuis le n°1, il y a déjà 15 ans ! Sans vous, Programmez ! n'existerait pas. Merci de votre fidélité. Et n'hésitez pas à parler de Programmez ! auprès de vos collègues et de vos amis.

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

sommaire

À LIRE DANS LE PROCHAIN NUMÉRO
n° 174 en kiosque le 30 avril 2014

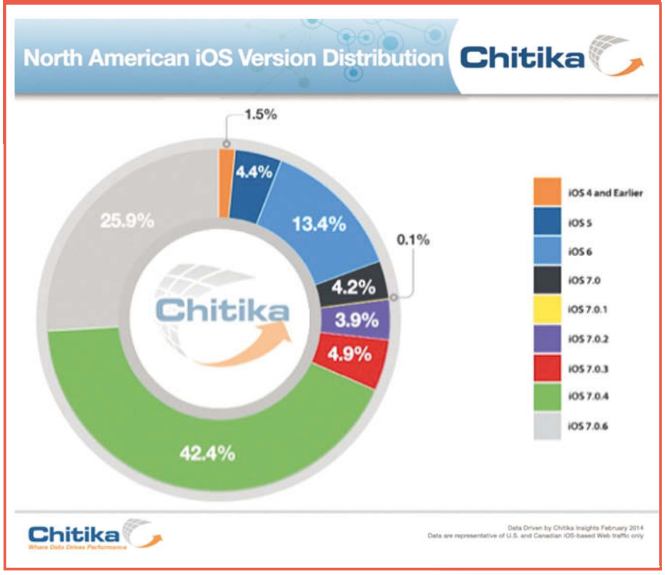
Langage
Langages fonctionnels : où et comment les utiliser ?

Matériel
Un poste de développement universel avec Linux

Web
Redécouvrez Grails

iOS 7.0.6 progresse vite

Apple a livré une mise à jour pour combler une importante faille de sécurité. En quelques jours, le quart des terminaux l'utilisait déjà (au moins en Amérique du Nord)



Les tablettes explosent...

+68 %
le marché en 2013 par rapport à 2014

61,9 %
la domination des tablettes sous Android est sans contestation possible, soit 121 millions de tablettes

36 %
la part de marché de iOS en 2013, soit 70,4 millions de tablettes

2,1 %
soit 4 millions de tablettes Windows 8 !

sources : Gartner

500 millions $?

Le vol de bitcoin réalisé contre le site d'échange Mt.Gox

Snow Leopard

Apple ne supporte plus officiellement cette version

Hadopi

non vous n'aurez pas accès aux documents liés à votre dossier Hadopi

Whonix 8

est disponible. Il s'agit d'un système basé sur Tor et Linux. Sécurité, anonymat.

LES NAVIGATEURS EN FRANCE

Le groupe CCM Benchmark a dévoilé le marché français des navigateurs :

1er : Chrome 31,32 %
(nette progression en 1 an)

2e : Safari (sic !) avec 20,63 %

3e : Internet Explorer 19,93 %, sauve de peu le podium mais subit une forte baisse

Et Firefox, la médaille en chocolat.

Apple

après le rachat de PrimeSense, l'accès aux outils et API OpenNI sera fermé le 23 avril prochain.

Gratuit

Windows 8.1 + Bing sera-t-il gratuit ? Ou comment séduire les utilisateurs de Windows 7.

Sony

vente des immeubles suite à sa réorganisation globale !

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

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

Février 2014	Février 2013	Tendance	Langage	%	Evolution (en %)
1	2	↑	C	18.334 %	+1.25%
2	1	↓	Java	17.316%	-1.07%
3	3	↑	Objective-C	11.341%	+1.54%
4	4	↓	C++	6.892%	-1.87%
5	5	↓	C#	6.450%	-0.23%
6	6	↓	PHP	4.219%	-0.85%
7	8	↑	(Visual) Basic	2.759%	-1.89%
8	7	↓	Python	2.157%	-2.79%
9	11	↑	JavaScript	1.929%	+0.51%
10	12	↑	Visual Basic .NET	1.798%	+0.79%

Le trio de tête reste identique : C, Java et Objective-C. Ce dernier connaît une petite progression. La plupart des autres langages subissent une baisse, exceptés VB et JavaScript.

Les applications mobiles sont partout ! Mais les tests sont-ils eux aussi partout ? La qualité des applications mobiles est-elle au rendez-vous ? Lors du dernier baromètre Borland du développement logiciel publié en janvier 2014 par Micro Focus France, 52% des répondants ont dit développer des applications pour les mobiles. 52% privilégient HTML5 comme langage de programmation et 38,2% d'autres codes spécifiquement dédiés au développement d'applications mobiles. Or dans le même temps, selon divers chiffres marché, moins d'une app sur 10 peut être mise à jour dans le mois qui suit les modifications des OS sur lesquels elle tourne.

Source : Borland

Samsung S5

une nouvelle version dès le printemps ?

La valse des media center pour Raspberry Pi

XBMC 3, OpenELEC 4 !

4K

OS X 10.9.3 supportera bien mieux les écrans 4K (mode Retina)

Corona

l'assistant vocal de Microsoft sera inclus dans Windows Phone 8.1

Android

un kit de développement Office 365 est disponible

Android, bis

Google veut imposer Android partout et sur tous les terminaux possibles et objets connectés. Un kit de développement spécifique a été dévoilé (ou le sera très très rapidement).

Google

une enquête des autorités indiennes anti-trust ?

5G

Anglais et Allemands veulent travailler ensemble sur la 5G !

Steve Jobs et la SOURIS

c'est confirmé, Jobs n'aimait pas mais alors pas du tout la souris à plusieurs boutons !

http://goo.gl/1omUbD

eclipse

Eclipse 4.4 (Luna) arrivera le 25 juin 2014. La version 4.5 est prévue pour le 24 juin 2015.

Programme de reprise d'iPhone en France :

Apple lance officiellement le programme de reprise en France. Jusqu'à 215 € sur un « vieux » iPhone pour en acheter un neuf !

76 milliards $

la fortune de Bill Gates (selon Forbes). Pauvre Larry Ellison (Oracle) et ses ridicules 48 milliards $.

Rachat de WhatsApp par Facebook contesté ?

Des utilisateurs s'inquiètent de la possibilité d'utilisation des données privées par Facebook...

Toi aussi

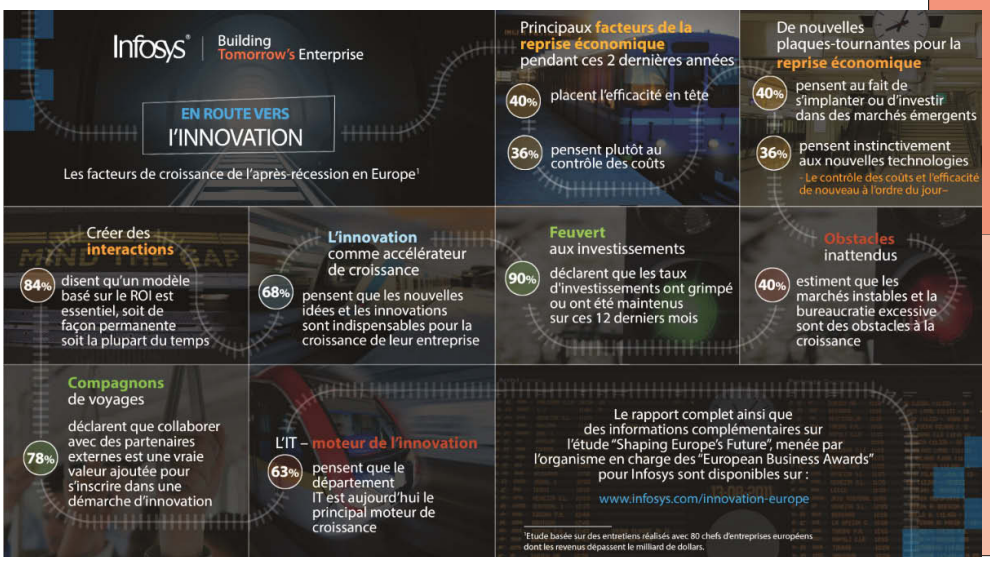
fabrique ton malware Andoid avec Dendroid (contre 300 €) !

Google I/O 2014

Google va-t-il battre son record de 49 minutes pour jouer à guichet fermé ? Objectif : faire mieux qu'Apple et sa WWDC 2013, à peine 120 secondes pour 5000 places...

NetBeans 8.0

est disponible



1&1 BOUTIQUES

UNE BELLE BOUTIQUE POUR VENDRE PLUS

- Débutant ou expert : concevez vous-même votre boutique en ligne.
- Au choix : plus de 100 designs professionnels adaptés à votre activité.
- Nom de domaine inclus. Vous pouvez aussi relier votre boutique à un domaine existant.
- Mobile : votre boutique s'affiche parfaitement sur tous les écrans, smartphones, tablettes ou PC.

CRÉDIT PAYPAL 25€ OFFERTS*

MEMBRE DE united internet

DOMAINES | MAIL | HÉBERGEMENT | E-COMMERCE | SERVEURS

* Crédit de 25 € pour chaque compte marchand PayPal ouvert en tant que nouveau vendeur avant le 31/05/2014, depuis la page Web PayPal dédiée mise à disposition par 1&1, à condition d'activer le compte chez 1&1 et de réaliser au moins 10 transactions PayPal effectives dans les 3 mois suivant son ouverture.

EN LIGNE

NOUVEAU !
OFFRE DE LANCEMENT
1 AN À 0,99 € HT/mois**
~~29,99~~
Vous économisez 348 €

LANCEZ-VOUS DANS LA VENTE EN LIGNE !

FONCTIONS AVANCÉES. VENTES RENFORCÉES.

- Totale flexibilité : votre boutique évolue avec votre entreprise.
- Vendez de manière ciblée : produits personnalisables, offres promotionnelles et vente croisée.
- Évaluations et recommandations : donnez la parole à vos clients.
- Vendez dans le monde entier : grand choix de devises, langues et modes de paiement sécurisés comme PayPal.

GAGNEZ DES CLIENTS. GARDEZ-LES LONGTEMPS.

- Obtenez une bonne place sur les résultats des moteurs de recherche grâce au référencement (SEO).
- Vendez aussi sur Amazon, eBay et autres marketplaces.
- Créez facilement votre propre boutique Facebook.
- Fidélisez vos clients avec des newsletters et des bons de réduction.

TOTALEMENT SÛR. TELLEMENT PRO.

- Sécurité certifiée avec Trusted Shops. Exclusivité 1&1 : modèles de textes juridiques à personnaliser.
- Choix du prestataire de livraison : Chronopost, So Colissimo, etc.
- Disponibilité maximale : hébergement dans deux datacenters 1&1 distincts.
- Support 24h/24, 7j/7 : nos experts répondent à toutes vos questions.

1 MOIS POUR ESSAYER

1 CHOIX VOTRE DURÉE D'ENGAGEMENT

1 APPEL UN EXPERT VOUS RÉPOND

1&1

0970 808 911
(appel non surtaxé)

1and1.fr

** 30 jours « satisfait ou 100 % remboursé ». 1&1 Boutique en ligne Basic est à 0,99 € HT/mois (1,19 € TTC) pendant 12 mois. À l'issue des 12 premiers mois, son prix habituel de 29,99 € HT/mois (35,99 € TTC) s'applique. Offre à durée limitée, soumise à un engagement de 12 mois. Offre sans durée minimale d'engagement également disponible. Conditions détaillées sur 1and1.fr.

devoxx 2013

Devoxx 2014 : du 16 au 18 avril à Paris

Devoxx est l'événement Java de l'année ! Ce sera déjà la 3e édition en France. Ce projet fut initié par Antonio Goncalves et le JUG Paris. Plus de 150 sessions techniques, ateliers, Labs, Tools in Action seront proposés durant trois jours. Plus de 1500 personnes sont attendues. Nicolas Martignole revient pour Programmez ! sur la genèse de Devoxx France et l'édition 2014.

Quand tu as initié Devoxx France, le pari n'était pas gagné...Peux-tu revenir sur les débuts, les difficultés ?

On a organisé en fait Devoxx France après avoir déjà organisé d'autres évènements. Le fait d'avoir passé différentes étapes nous a aidé à apprendre, comprendre et à monter une équipe. En 2011, nous avons par exemple organisé le 3ème anniversaire à la Cité Universitaire au sud de Paris, avec 500 personnes, des sponsors, etc. Après ce succès, cela nous a donné envie de faire plus grand, et différent. L'idée de départ était une conférence pour les développeurs, sur 2 ou 3 jours.

Antonio est allé rencontrer le fondateur de Devoxx Belgique, et a eu l'idée de proposer une version Française. Entre le mois de juin 2012 et le mois de novembre 2012, il a fallut travailler en secret, n'étant pas certain de pouvoir monter le projet. De septembre 2012 à novembre, nous avons travaillé à 4 (Antonio, Zouheir, José et moi-même) en allant démarcher les sponsors historiques du Paris JUG. Le financement au départ était délicat. Mais grâce aux aides de nos sponsors, qui ont accepté de jouer le jeu, alors que le projet était secret, cela a fonctionné. Nous avons signé la franchise avec Devoxx quelques jours avant l'annonce. J'étais en Belgique, et lorsqu'enfin nous avons su que le projet se ferait, c'était beaucoup d'émotions et de stress en moins. 9 sponsors ont signé dès le départ. La suite, on la connait, c'est un succès qui nous a dépassés. Nous avions prévu 900 personnes, nous en avons eu 1250 pour 2012. Et en 2013, nous avons vendu toutes les places 5 semaines avec la conférence. 1400 personnes au total. Enfin cette année, au rythme où vont les achats de places, nous serons sold-out 7 semaines avant la conférence.

Quelle est la philosophie, l'esprit de Devoxx ?

Offrir un contenu de qualité autour de la plateforme Java. Pour cela, nous suivons les bonnes pratiques et les idées qui ont marché pour Devoxx Belgique (la 12ème édition a fait 3500 développeurs). L'esprit c'est « Développeur Passionné » . Celui qui aime son métier, coder et rencontrer d'autres développeurs. Devoxx c'est la voix du développeur. Les sujets sont donc présentés par des gars comme nous. Du débutant à la super-star, tout le monde a sa chance. La sélection cependant est difficile. Pour 670 sujets reçus, notre équipe de 14 sélectionneurs n'a retenu que 180 sujets. Dans ces 180 sujets, tu as 13 conférences sponsors, ce qui est très peu finalement

Le JUG Paris joue un rôle moteur mais tu n'oublies jamais de citer les autres JUG, quel est le rôle des JUG en France et pour la communauté Java ?

Ce sont les relais de la communication de Devoxx France. La France était en retard en 2008, il n'y avait que le JUG de Tour. Antonio en décidant de créer le Paris JUG, a initié un mouvement. Je crois qu'il y a 18 ou 19 Java User Group en France. Chaque mois, ce sont des conférences et surtout, des rencontres. Le rôle des JUG est d'offrir un espace de rencontre pour les développeurs. Cela permet d'échanger et de débattre, avec beaucoup de passion et de questions sur notre métier.

Devoxx parle bien entendu de Java mais pas uniquement. Quels sont les thèmes et les axes pour cette édition 2014 ? Et les speakers à ne pas rater ?

L'axe central est, et restera, la plateforme Java. Cette année 2014 est placée sous le signe de la sortie de la version 8 de Java. Nous aurons par exemple José Paumard et Rémi Forax, qui viendront chacun parler sur ce sujet à Devoxx France. C'est aussi la sortie de Java EE 7. Arun Gupta, Antoine Sabot-Durand, Antonio Goncalves, David Delabasse pour ne citer qu'eux, présenteront des sujets sur Java EE. Côté Spring, Josh Long de Pivotal viendra parler de Spring 4. Ensuite, si je regarde du côté Web, nous aurons Sébastien Deleuze, qui parlera de Dart et de Spring 4. Nous aurons aussi James Ward de Typesafe sur Play2 en compagnie de Nicolas Leroux. Côté Langages alternatifs, cette année nous sommes sortis de Java. Et je te l'annonce aussi : nous aurons du .NET, du F#, de l'Erlang (utilisé par les 32 petits gars de What'sApp, le machin que Facebook a racheté 15 ou 16 milliards). Nous aurons du Clojure, du Golo et un peu de Dart.

Javascript, le Web et tout ce qui est HTML est aussi bien représenté, avec pas moins de 23 sujets dans ce thème.

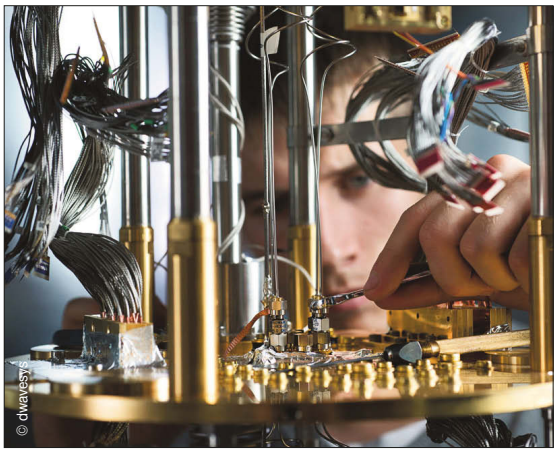
En suivant aussi le mouvement de Devoxx Belgique 2013, nous aurons beaucoup de robotique, domotique, ce que l'on regroupe dans la track Future<Devoxx>. Internet of Things, robot NAO, hackaton, il va y avoir du code et du matériel cette année. Pas moins de 3 ateliers pratiques sont prévus pendant les 3 jours, pour hacker et coder.

Concernant la track « Startup/Innovation » nous aurons Zach Holman de Github, Pierre-Yves Ricau de Square Inc., ainsi qu'une présentation des Googles Glass par Alain Regnier. Il y aura aussi un événement « Seed Networking » le jeudi soir, afin de permettre à des entrepreneurs de trouver des développeurs. (http://www.seednetworking.fr)

Le programme complet est disponible sur le site officiel : http://www.devoxx.fr

Chapitre 6 : I.A. ?

Supply Chain de l'Intelligence !



La physique quantique, un des futurs de notre civilisation ?

« Vous me mettrez bien un peu d'I.A., un peu **d'Intelligence Artificielle** dans votre pitch ! La valeur de votre startup n'en sera que plus élevée ». Eh oui, l'IA ou entendez Intelligence Artificielle, est plus que jamais une des expressions à la mode dans bon nombre des pitchs startups, meetups de la Silicon Valley, conversations ou rencontres entre Geeks dans l'une des places branchées de University Avenue de Palo Alto.

Et pour cause, nous nous sommes définis en tant qu'« Homo Sapiens », l'Homme sage, tant notre intelligence est importante pour nous. Depuis plusieurs siècles, notre espèce n'a cessé de tenter de comprendre comment nous pensons, percevons, comprenons, anticipons ou encore manipulons notre environnement si large et tellement plus complexe que nous !

L'Intelligence Artificielle (IA) se veut ainsi être le champ d'action où l'être humain ne se limitera pas au simple fait de comprendre son intelligence, mais pourra s'initier à la construction d'une entité intelligente, d'une Intelligence Artificielle.

Ce terme ou domaine au goût du jour ne désemplit pas, et bon nombre de personnes l'utilisent pour tout et son contraire.

Le recrutement de Ray Kurzweil par Google, de Yann LeCun par Facebook ou encore les annonces hebdomadaires d'IBM autour de Watson ne font qu'amplifier la force du signal et la nécessité d'en comprendre toute la subtilité afin de ne pas se faire dépasser ou de simplement passer à côté de l'opportunité, preuve étant faite par cet article qui lui est dédié ! Allons alors au bout de la démarche !

Intelligence Artificielle, de quoi parlons-nous ?

Les pieds bien fixés au sol, définissez-moi le terme IA dans vos propres mots. Allez-y, faites l'exercice en silence ou lancez simplement le débat lors de la prochaine pause-café avec vos collègues Geeks !

Eh oui, pas si simple de trouver un consensus ou de définir clairement ce qui se cache derrière l'Intelligence Artificielle, et pourtant ce terme nous semble tellement familier grâce à Asimov et ses héritiers hollywoodiens.

Ainsi je dois vous avouer que me rompre à l'exercice après toutes ces années de passion pour l'Intelligence Artificielle est bien complexe, voire impossible.

Peut-être est-ce la voix de la raison ou de la sagesse au regard des années d'investigations cumulées à explorer seulement quelques-unes de ses facettes que peuvent être les agents intelligents, le traitement du langage naturel, les systèmes experts, l'intelligence sociale, la communication, la perception ou encore l'action autonome... et bien plus encore !

Retournons-nous vers l'une des références mondiales, directeur R&D Google à Mountain View, dont j'ai eu la chance de suivre les succulents enseignements sur le sujet : « ***Peter Norvig*** » et plus particulièrement son œuvre littéraire sur le domaine en binôme avec ***Stuart J. Russel*** : « ***Artificial Intelligence – a modern approach 3rd Edition*** ».

Livre (ou brique) que je ne peux que chaudement vous recommander pour votre bibliothèque de Geek, voire en livre de chevet ce qui est mon cas ! Ainsi, l'intelligence artificielle peut être approchée sur 4 facettes dominantes :

Les systèmes pensant comme des êtres humains

En clair les sciences cognitives ou l'approche cognitive par extrapolation de modèles de pensées de l'être humain.

Adhérer à cette démarche est tout aussi intéressant qu'intrigant, car elle vous demandera une étape anthropomorphique préalable à toute mise en œuvre.

La compréhension de l'être humain et de son modèle de pensées est au centre des recherches expérimentales de toute programmation d'une Intelligence Artificielle de ce type. Lemme nécessaire pour ensuite évaluer ses similarités avec le raisonnement humain.

Cette première facette de l'IA est très certainement l'une des plus fascinantes tant le spectre des possibles, à l'image de la pensée humaine, est sans limite !

Les systèmes qui agissent comme des êtres humains

C'est ici que nous retrouvons le fameux « test de Turing » considérant une machine comme intelligente si elle peut converser de telle manière que les personnes humaines ne puissent la distinguer d'un de leurs congénères.

Cette facette couvre les disciplines :

- Le ***natural language processing*** permettant une communication linguistique avec le système.
- La ***représentation de connaissance*** permettant le stockage des informations à utiliser par le système pour répondre à l'utilisateur, exemple dbpedia.
- Le **raisonnement automatique** permettant l'utilisation des informations stockées pour répondre aux questions posées par l'utilisateur ainsi que créer de nouvelles connexions.
- Le **machine learning** permettant au système de s'adapter aux nouvelles circonstances ou détecter et extrapoler les nouveaux patterns.

À ces 4 premières disciplines, n'apportant pas de couvertures aux interactions physiques entre l'individu et la machine, Peter Norvig et Suart Russel proposent d'en ajouter deux que sont :

- La **vision digitale** permettant la perception d'objets
- La **robotique** permettant alors de les bouger.

Les systèmes qui pensent rationnellement

Inspirée du philosophe grec Aristote qui fut l'un des tout premiers à tenter de « codifier la pensée » et à l'origine des processus de raisonnement proposant des patterns de réflexions logiques en vue d'aboutir à un maximum de réponses correctes (exemple : vous êtes un Geek, les Geeks adorent les derniers gadgets high-tech, vous adorez les derniers gadgets high-tech !).

Cette troisième facette aborde la compréhension d'agents (entités d'actions venant du latin *agere*, agir ou opérer) ou systèmes devant raisonner de manière logique ou rationnelle. Cette approche est très régulièrement contre-versée de par son incapacité à couvrir certaines capacités de l'être humain, comme la perception, difficilement exprimable en logique ou algorithmes.

À cette première limite de représentation algorithmique s'ajoute celle relative aux technologies ne permettant pas encore réellement de réaliser les traitements et calculs nécessaires afin de couvrir la majorité de cette facette.

Les systèmes qui agissent rationnellement

Cette dernière facette peut-être présentée comme celle de l'agent rationnel, l'agent ayant à charge d'agir en conséquence afin d'atteindre

le meilleur résultat, ou en environnement incertain, le meilleur résultat espéré.
L'agent ayant la faculté de s'adapter (d'être polymorphe) afin de satisfaire au mieux ses objectifs.
La mise en œuvre des disciplines présentées en seconde facette, et plus particulièrement la représentation de connaissance ou le raisonnement, permet à des agents rationnels de prendre les bonnes décisions.
Par exemple, la capacité d'intégrer un certain formalisme de phrases en langage naturel par machine learning d'un historique de dialogues afin de générer des comportements effectifs de réponses d'un agent en est une résultante. Vous y retrouvez les composantes ou disciplines de systèmes agissant comme des êtres humains avec de nombreuses capacités d'extension ou d'apprentissage pour l'agent afin de l'optimiser dans sa conduite d'atteinte du ou des objectifs de manière rationnelle.
Cette facette présente deux avantages majeurs à ces congénères que sont la minimisation de l'inférence qui devient dans ce cas précis un mécanisme pour atteindre la rationalité, ainsi que la mise en exergue d'une démarche plus mathématique qu'une démarche comportementale de l'être humain.
Je ne pourrais que difficilement vous cacher que cette dernière facette correspond aux évènements et pratiques mises en œuvre au sein de la Silicon Valley, ainsi que la vision de l'Intelligence Artificielle partagée au sein de l'équipe xBrainSoft, sachant que nous n'hésitons pas de temps à autre à revenir passer quelques instants dans les autres facettes en quête d'inspiration disruptive.
À la fin de la journée, le passage en revue de ces facettes nous clarifie les périmètres, mais ne nous fait pas émerger définitivement une définition claire et précise de l'Intelligence Artificielle. Et pour cause, l'accumulation d'expériences dans ce domaine force le respect et l'humilité tant sur sa description formelle que sur son objectif ultime. Penchons-nous quelques instants sur l'histoire de l'IA qui peut-être ajoutera sa pierre à l'édifice.

Il était une fois l'IA !

Nous pourrions démarrer la découverte de l'histoire de l'IA et de ses périodes d'incubation successives depuis Aristote (384-322 BC) qui était le premier à formuler des lois permettant de rationnellement régir une partie des raisonnements humains, fondement de certaines facettes de l'être humain.
Cette ligne du temps de l'IA est merveilleuse à comprendre tant elle reflète ce que nous sommes au plus profond de notre Être. Je vous encourage à parcourir les informations disponibles au sein du livre de Norvig ou encore sur les nombreux sites internet synthétisant pour certains ces étapes.

- ***1943-1955 : Gestation de l'IA*** avec les travaux de McCulloch, Pins, Pitts et Hebb autour des modèles de neurones artificiels ou encore Minsky et Edmonds avec le premier réseau de neurones. C'est ici que Turing publie son article introduisant le « Test de Turing ».
- ***1956 : Naissance de l'IA*** avec l'apparition des fondements de celle-ci grâce à McCarthy qui avait convaincu 10 des meilleurs scientifiques du domaine à se réunir durant 2 mois.
- ***1952-1973 : Premiers succès, premières déceptions*** avec l'apparition de nombreux programmes comme le Logic Theorist de Newell et Simon ou encore leur General Problem Solver permettant de résoudre des puzzles simples, la naissance du langage de programmation de haut niveau **Lisp** par McCarthy au sein du MIT Lab ou encore du programme ANALOGY par Tom Evans et la démarche vers le sens commun. Mais c'est aussi sur un second temps une période de déception par péché d'optimisme avec le cinglant rapport de Lighthill en 1973 qui stoppa net la quasi-totalité des projets d'IA en Grande-Bretagne.
- ***1969-1979 : Émergence des systèmes experts*** comme DENDRAL ou MYCIN basé sur des grands nombres de règles heuristiques.
- ***1980— présent : l'industrie et l'IA, les réseaux neuronaux – épisode 2*** avec des effets d'économies d'échelle pour les industries essentiellement Américaines et Japonaises, mais aussi la mise en œuvre de la règle de « Back-Propagation » donnant naissance à l'apprentissage automatique, aujourd'hui l'un des domaines les plus actifs de l'IA.
- ***1987— présent : l'IA en tant que Science*** avec l'avènement des HMMs (Hidden Markov Models) ; du Data Mining et du Bayesian Network.
- ***1995— présent : Les Agents Intelligents*** et la définition de l'AGI (**Artificial General Intelligence**) tentant de définir un algorithme universel d'apprentissage et d'action en tout environnement. Ainsi que le concept d'une AI amicale.
- ***2001— présent : l'avènement du Big Data*** permettant l'enrichissement des modèles par extraction de patterns via machine learning.

Retenons ainsi ces quelques dates clés afin de comprendre à quel point l'IA n'est pas un sujet neuf et à quel point elle a déjà fait couler beaucoup d'encre, créé des succès, mais aussi des mécontentements.
D'où l'intérêt de clarifier ou limiter sa médiatisation afin d'éviter trop de déceptions.

IA pour qui, pour quoi ?

Comme vous l'aurez compris, l'IA est un domaine vaste, très vaste et ses applications le sont tout autant. Nous ne pourrons pas parcourir l'intégralité des métiers pouvant contribuer et/ou exploiter l'intelligence artificielle, en voici quelques-uns : traitement du langage naturel pour la détection d'humeur, la robotique, l'aviation, détection de fraude e-commerce, les jeux, la traduction automatique, la médecine, la finance ou encore la logistique.
À ces domaines d'applications, nous pourrons également ajouter des facettes complémentaires ou sous-facettes non parcourues préalablement comme :

- L'intelligence sociale ou l'affective computing,
- L'ontologie,
- La programmation génétique,
- L'épistémologie,
- ... et bien d'autres encore que je vous laisserai découvrir par vous-même.

L'élément intéressant dans l'ensemble de ces exemples est qu'il est rare de voir plus de 3 ou 4 compétences ou disciplines comme par exemple le langage naturel, le machine learning de l'Intelligence Artificielle exploitée en même temps; la robotique ou les Assistants Personnels sont peut-être parmi les plus complets.

Conclusion

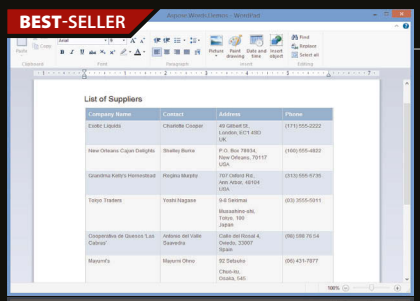
L'Intelligence Artificielle force l'humilité et le respect, peut-être n'avons-nous pas encore compris l'amplitude des enjeux réels ou l'objectif final ? Et si la création d'IA ou plus particulièrement d'une IA générale était notre Tâche avec un T majuscule, travail non pas de l'individu unitaire, mais de l'humanité afin de permettre à notre hôte stellaire et son intelligence naturelle de se répandre au-delà de ses simples frontières et contraintes physiques qui l'en empêchent ? La conquête de l'espace et de l'univers passerait ainsi par l'intelligence et sa distribution, la mise en œuvre et l'exécution d'une Supply Chain de l'intelligence !

Gregory **Renard** [Redo] - *Geek in Silicon Valley* - http://gregoryrenard.wordpress.com

Liens et références :

http://aima.cs.berkeley.edu
http://norvig.com/
http://www-formal.stanford.edu/jmc/whatisai/whatisai.html
http://en.wikipedia.org/wiki/Artificial_intelligence
http://aihub.net
http://www.artificialbrains.com/
https://ai.stanford.edu
http://www.csail.mit.edu/
http://en.wikipedia.org/wiki/John_McCarthy_(computer_scientist)

BEST-SELLER

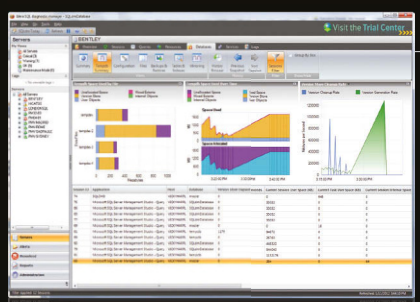


Aspose.Words for .NET | à partir de € 718

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



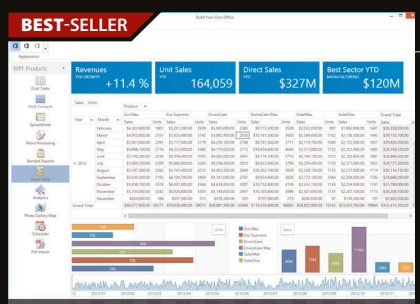
SQL Diagnostic Manager | à partir de € 1 719

idera

Suivi des performances SQL 24h/24, 7j/7, alertes et diagnostics.

- Suivi des performances SQL Server physiques et virtuelles
- Analyse approfondie des requêtes pour identifier les délais et les consommations de ressource excessifs
- Planification de la capacité révélant les tendances de croissance des bases de données et réduisant l'utilisation du serveur

BEST-SELLER



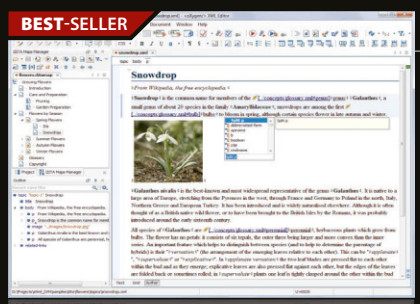
DevExpress Universal Suite | à partir de € 1 580

DevExpress

400+ outils et contrôles WinForms, ASP.NET, WPF, Silverlight et Windows 8.

- Exploitez votre base de codes pour développer des applications tactiles multiplateformes
- Inclut un tableau de bord de visualisation interplateformes et un serveur de rapports
- Nouveaux contrôles DevExpress incluant Tableur et Carte
- Codage, débogage et refactorisation avec CodeRush pour Visual Studio
- Inclut la nouvelle galerie unifiée de modèles applicatifs DevExpress

BEST-SELLER



oXygen XML Editor Professional | à partir de € 350

<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

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 AMERICAN EXPRESS DISCOVER EUROCARD MasterCard

Comment « performer » face aux TOP développeurs ?

Quand je dis TOP développeur, ce n'est pas un excès de zèle, mais bien une réalité admise par mes pairs. En général, les gens qui bossent sur les systèmes, les jeux ou encore les outils de dev sont souvent très bons. Spécialement dans les grandes boîtes comme Microsoft, Google... qui ont les moyens pour investir dans les développeurs et leurs formations. On entend à longueur de journée « nous voulons d'avantage de VISAS, non pas qu'il n'y a pas assez de développeur aux USA, mais parce que nous voulons les meilleurs au monde ».

Présentation de mon produit au Microsoft Store avec mon super Product Owner

1 an = 10 ans ?

Ma première année chez Surface puis Windows m'ont appris beaucoup de choses : humainement tout d'abord. L'anglais, la façon de se comporter dans une entreprise américaine, les processus et méthodologies, la hiérarchie. Puis la technique, avec le C++, le kernel Windows, le hardware, la simplicité plutôt que la complexité, les revues de codes, les architectures, l'algorithmique... Bref, une année extrêmement enrichissante qui m'en parait 10. J'ai l'impression d'avoir muri techniquement et humainement.

Ceci étant dit, il était temps de me demander une chose essentielle « comment rentrer dans la compétition et améliorer ma carrière ». Pendant un an, j'avais omis cette question, je m'étais concentré sur mon intégration (réussie !) et mes fonctionnalités à développer.

Je n'ai pas voulu me projeter comme un réel compétiteur, mon équipe est là depuis 15 ans et étant le plus jeune, je me suis mis en position « young padawan ». J'ai fait une bonne année en tant que SDE I, mon manager est content et j'ai une eu bonne revue de fin d'année.

Ceci étant dit, je me suis enfin posé la question : quelles sont les choses à mettre en œuvre pour que je puisse passer SDE II ? Que dois-je faire pour que mes collègues me voient comme un compétiteur plutôt qu'un petit jeune plein de motivation.

L'ALGORITHME ET LA RÉSOLUTION DE PROBLÈMES

Ce sont ces deux notions qui sont la « base » de tout bon ingénieur logiciel. Sans eux, aucune compétition n'est possible. C'est pourquoi, j'ai entamé pendant un an une remise à niveau sur ces sujets. Car pour moi, l'informatique c'est de l'ingénierie mais AUSSI une science (d'ailleurs on dit bien computer science & engineering).

La méthode d'apprentissage que j'ai utilisée est vraiment performante pour moi, pas sûr que ça marche pour tout le monde.

Pour chaque « chapitre » du livre de Princeton (Algorithms 4eme Edition), il faut :

- Lire le chapitre du livre de Princeton : Il est vraiment top ce livre, accessible, vraiment clair. Il ne faut pas hésiter à taper les exemples pour les mémoriser et faire les exercices de fin.
- Utiliser Wikipedia

  Une fois que vous avez compris la notion, c'est souvent difficile de voir comment l'utiliser dans le monde réel. Pour cela la meilleure solution est de lire ce que dit Wikipédia sur le chapitre concerné. Wikipédia est juste la MINE d'OR pour apprendre l'algo. Il faut lire la version anglaise ET Française, car des petits détails se cachent souvent dans l'une ou l'autre version. Wikipédia donnera souvent des bons conseils de quand/où/pourquoi utiliser la notion d'algorithme que vous venez d'apprendre. Ce qu'il faut, c'est rester dans le concret. Apprendre un arbre binaire pour juste savoir ce que c'est, ça ne sert à rien. Mais se demander où il a déjà été implémenté (ou pas) et pourquoi permet de mieux retenir. On peut aussi compléter ça par les documents de l'école polytechnique qui traînent sur le net, ou les cours en ligne du MIT/Stanford ou de facs françaises.
- Faire les exercices du livre en rapport avec le chapitre concerné
- Puis enfin finir par lire le chapitre du livre de Cormen (il existe une version Française) « Introduction to Algorithms ».

Je comprends les choses. Mais cela ne fait pas de moi un bon en algo, juste quelqu'un qui connaît les bases. Il faut maintenant pratiquer.

DES PRINCIPES FONDAMENTAUX

Etre bon en résolution de problèmes n'est pas suffisant, il faut également suivre certains principes. C'est là où on voit vite que devenir un bon ingénieur logiciel ne se limite pas à l'informaticien mais aussi à l'Homme que vous êtes.

Il faut chaque jour essayer de montrer du ownership, du leadership, l'obsession de la perfection pour nos clients, la créativité, être autocritique... Bref, devenir un ingénieur logiciel reconnu et compétitif est un travail de longue haleine et exige chaque jour de se remettre en question.

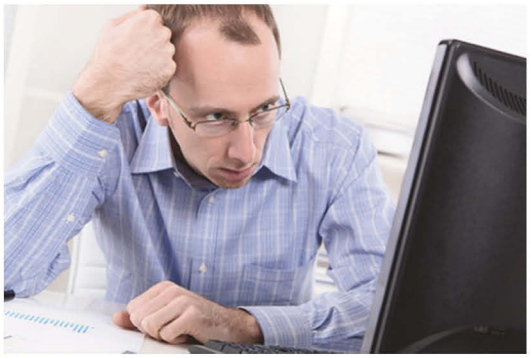
Se remettre en question et toujours apprendre : c'est le seul moyen de performer face à ces ingénieurs.

Rendez-vous le mois prochain !

Ras le bol de coder?

Quand le commercial annonce
la livraison pour la semaine prochaine...

Quand les specs changent 3 jours
avant la deadline...



Quand ça fait 2 heures que
ça ne compile pas...

Quand je corrige mon code
en production...

Ne codez plus, paramétrez !

NOUT Builder est le premier framework de
création de logiciels et sites web avec 0% de code.

Par paramétrage, la réalisation est plus sûre et plus rapide.
Le résultat est plus évolutif et personnalisable par votre client.

Vidéos et infos sur www.nout.fr/nout-builder/

Les plaisirs des failles GOTO

Le mois de février a été cruel pour la sécurité informatique. Une des plus belles failles découvertes pour ce début 2014 est la fameuse faille GOTO dans OS X et iOS ! Apple a laissé passer une énorme faille...

Le problème se loge dans le fichier source sslKeyExchange.c. L'extrait du code est éloquent :

```
hashOut.data = hashes + SSL_MD5_DIGEST_LEN;
  hashOut.length = SSL_SHA1_DIGEST_LEN;
  if ((err = SSLFreeBuffer(&hashCtx)) != 0)
    goto fail;

  if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
    goto fail;
  if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
    goto fail;
  if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
  if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
  if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

  err = sslRawVerify(ctx,
              ctx->peerPubKey,
              dataToSign,        /* plaintext */
              dataToSignLen,     /* plaintext length */
              signature,
              signatureLen);
  if(err) {
    sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
          "returned %d\n", (int)err);
    goto fail;
  }
```

On repère très rapidement la double instruction goto fail. Les experts en sécurité de Lexsi analysent la faille ainsi : *« Bien que l'indentation du code source puisse nous induire en erreur, le problème se situe ici. En effet, l'analyse minutieuse du code source montre que quel que soit le résultat de l'appel à la fonction SSLHashSHA1.update(&hashCtx,&signedParams) le flux d'exécution ira systématiquement en goto fail retournant, en cas de réussite de la fonction précédente (valeur de retour égale à 0), la valeur err qui n'aura pas été affectée par les appels aux fonctions SSLHash SHA1.final()etsslRawVerify()... Notons qu'il faut que les trois fonctions SSLHashSHA1.update et que la fonction ReadyHash ne retournent pas d'erreur afin que la valeur de err soit égale à 0, permettant un retour « valide » de la fonctionSSLVerifySignedServerKeyExchange. La signature du message ServerKeyExchange ne sera donc jamais vérifiée permettant d'usurper l'identité du serveur.»*

lib/x509/verify.c (+10/-6)

```
141 141   if (result < 0)
142 142     {
143 143       gnutls_assert ();
144         -   goto cleanup;
    144     +   goto fail;
145 145     }
146 146
147 147   result =
...
150 150   if (result < 0)
151 151     {
152 152       gnutls_assert ();
153         -   goto cleanup;
    153     +   goto fail;
154 154     }
155 155
156 156   result =
...
158 158   if (result < 0)
159 159     {
160 160       gnutls_assert ();
161         -   goto cleanup;
    161     +   goto fail;
```

Cette faille permet de capturer et de modifier des données échangées entre l'utilisateur et un serveur distant, dans une connexion SSL... Mais aussi étonnant que cela puisse être, et malgré que cette partie du code OS X / iOS soit disponible publiquement (il fait partie des couches Open Source), la faille n'a été découverte qu'en janvier 2014. Malveillance ? Erreur d'un développeur ? Apple n'a rien dit. Comment une telle erreur a-t-elle pu être introduite ?

Quoi qu'il en soit, Apple a procédé à des mises à jour pour OS X et iOS.

On peut rigoler mais il n'y a pas qu'Apple !

Il ne faut jamais rire trop vite des failles des autres. Le monde Linux n'est pas épargné par les failles. Ainsi, début mars, un joli bug dans la librairie gnuTLS a été découvert, dans la sécurité SSL et TLS (Transport Layer Security). Or, cette librairie est cruciale dans les communications SSL / TLS pour les systèmes Linux et les applications l'utilisant. Ce n'est pas une faille identique à celle d'Apple, mais le parallèle est singulier. En urgence, la communauté a modifié le code et il est recommandé de déployer la version 3.2.12 de la librairie.

Ce bug a été découvert lors d'un audit de code effectué par Red Hat !

Pour en savoir plus :

http://arstechnica.com/security/2014/03/critical-crypto-bug-leaves-linux-hundreds-of-apps-open-to-eavesdropping/

et https://rhn.redhat.com/errata/RHSA-2014-0247.html

François **Tonic**

Une année pleine de technologies et de codes avec

PROGRAMMEZ!
le magazine du développeur
www.programmez.com

1 an 11 numéros
49€
seulement (*)

2 ans 22 numéros
79€
seulement (*)

Spécial étudiant
39€
1 an 11 numéros

réservée à la France Métropolitaine

(*) Tarifs France métropolitaine

© IStock - dolgachov

Toutes nos offres sur www.programmez.com

Oui, je m'abonne

ABONNEMENT retourner avec votre règlement à
Programmez, 17, route des Boulangers 78926 Yvelines cedex 9

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

Tarifs France métropolitaine

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

Tarifs France métropolitaine

☐ M. ☐ Mme ☐ Mlle Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

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

E-mail : @

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

PROG 173
Offre limitée, valable jusqu'au 30 avril 2014

Les tests et le développement sécurisé peuvent-ils réduire les failles d'un système et les risques de hacking ?

D'une étude statistique menée par SourceFire(1) sur 25 ans (cf graphique), avec des résultats similaires à d'autres sources comme CVE(2) , OSVDB(3) , OWASP(4) , il ressort clairement que les failles de sécurité sur les applications proviennent :

- Soit de défauts introduits dans le système vulnérable (par exemple débordement de tampons, XSS, injection SQL), lors de son développement et sa maintenance,
- Soit d'une mauvaise adéquation entre le système et son environnement opérationnel (par exemple, mauvaise configuration de serveurs), lors de son déploiement et sa maintenance.

A partir de ces typologies de vulnérabilités, les causes possibles, erreurs faites par les maîtrises d'ouvrage et les maîtrises d'œuvre de nos organisations, peuvent être classifiées :

- Lacunes dans l'expression de besoins de sécurité : quelles sont les biens sensibles à protéger ?
- Absence de spécifications des fonctionnalités et mécanismes de sécurité : comment les biens sensibles doivent être protégés ?
- Perte de maîtrise des architectures de sécurité qui deviennent de plus en plus complexes : quand, comment et où faut-il déployer des composants de sécurité ?
- Mauvaise maîtrise des socles techniques sur lesquels sont installés les applications : quels sont les mécanismes de sécurité offerts ? Comment doivent-ils être configurés ?

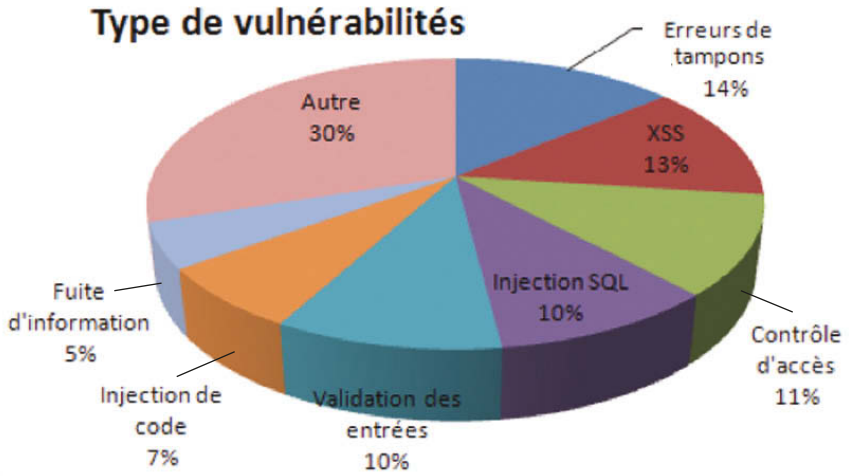
Il est donc primordial de prendre en compte la sécurité tout au long du cycle de vie d'un système. Mais comment? Il s'agit : d'exprimer des besoins de sécurité basés sur une identification des biens sensibles à protéger, en prenant en compte les contraintes métier et réglementaires (comme par exemple, celles édictées par l'ANSSI)

- De spécifier les fonctions et mécanismes de sécurité à développer pour protéger les biens sensibles
- De définir les règles à respecter en termes de développement, déploiement et maintenance sécurisés
- De vérifier le niveau de sécurité des mesures organisationnelles et techniques mises en place par du test

Développement sécurisé

Pour limiter les failles de sécurité, des règles d'architecture et de développement doivent être définies et respectées dont voici quelques exemples :

- Utilisation de bibliothèques offrant des mécanismes de sécurité, comme l'OWASP Enterprise Security API ou JAAS(5) (authentification), Apache commons validator (validation de zones de saisie)
- Définition de la qualité des mots de passe (longueur, caractères à utiliser)
- Utilisation de mécanismes pour éviter le "rejeu" et la découverte de mots de passe (par exemple, Captcha)
- Filtrage d'informations saisies, en utilisant des expressions régulières
- Utilisation de pages d'erreur par défaut (pour éviter de dévoiler des informations système)



Tests de sécurité

Le respect des règles fixées doit être vérifié. D'où la nécessité de faire du test de sécurité, et pas uniquement lorsque le système est déployé ! Une politique de tests de sécurité doit être définie. « Tout tester est impossible », donc :

- Quelles sont les informations sensibles ?
- Comment sont-elles protégées
- Comment peuvent-elles être atteintes ?

Les tests de sécurité doivent être réalisés au bon moment :

- Revue (avant le codage) : spécifications (exigences de sécurité) et architecture (utilisation de bibliothèques de sécurité, chiffrement des informations sensibles)
- Revue de code et Analyse statique (pendant le codage) : respect des règles de codage, recherche d'erreurs de codage (utilisation de variables non initialisées, double désallocation, etc.)
- Tests dynamiques (tests unitaires, intégration, système et acceptation) : conformité aux spécifications (tests aux limites, classes d'équivalence, fuzzing, attaque par force brute, injection de code, etc.)
- Revues et tests de configuration (lors du déploiement) : mots de passe par défaut, services non utilisés
- Revues et tests de maintenance : tests de non régression suite à mise à jour socle technique et mise à jour applicative (analyse d'impact)

Cas pratique : Mesures à prendre pour les failles critiques XSS

Même si les vulnérabilités Cross-site scripting (XSS) sont désormais largement connues des développeurs Web et des frameworks applicatifs, leur fréquence d'apparition reste très alarmante. A ce jour, 40 % des vulnérabilités remontées dans les bases telles que CVE sont de type XSS. Plusieurs raisons justifient de tels chiffres :

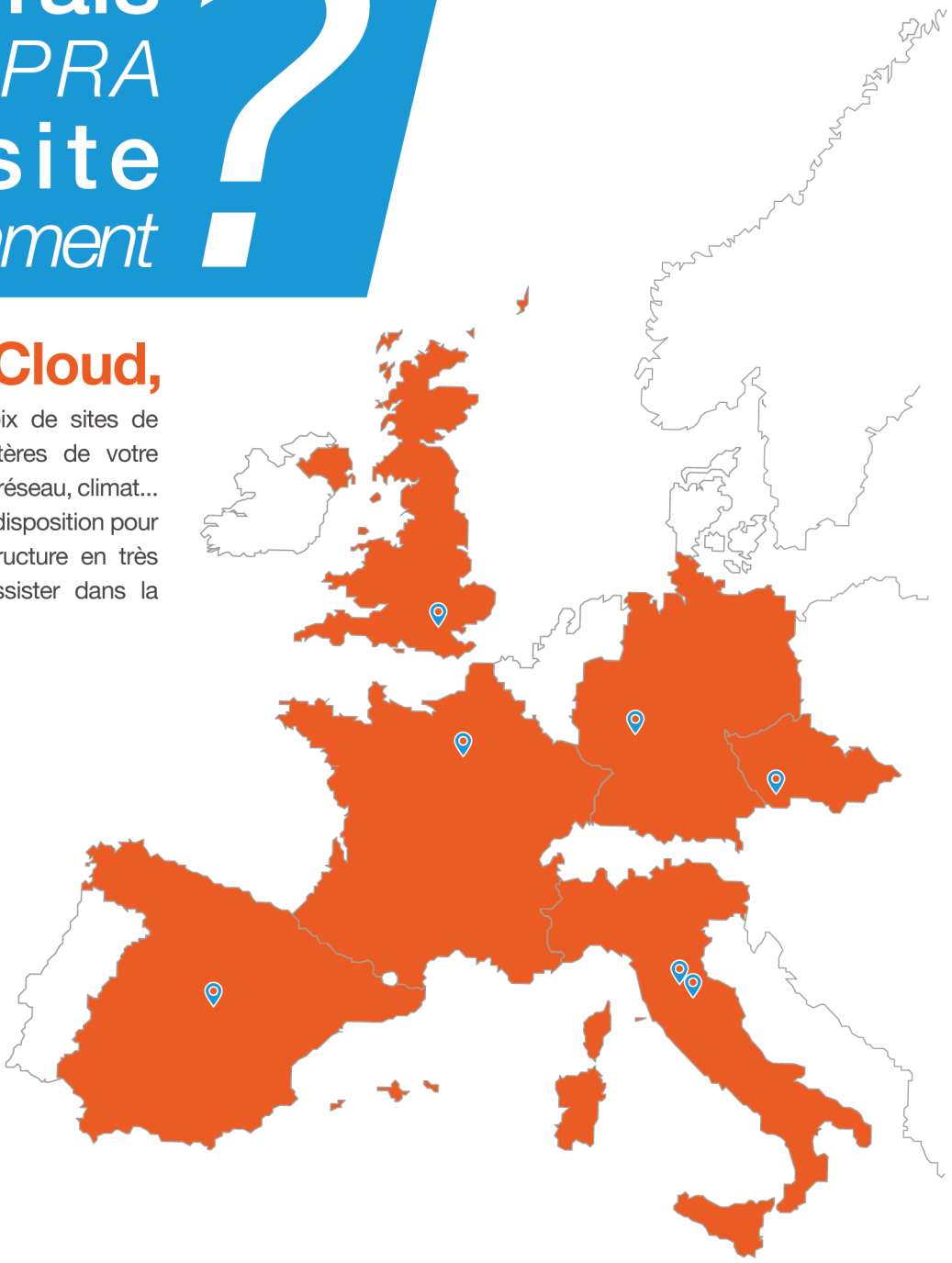
- Les développeurs Web n'ont pas toujours les connaissances suffisantes en sécurité ni les réflexes de programmation nécessaires pour prévenir ces failles ;
- La forte concurrence dans le domaine de la création d'applications Web pousse les entreprises à réduire les temps de développement. Est ainsi privilégiée la finalisation des modules d'un point de vue fonctionnel plutôt que d'un point de vue sécurité.
- La complexité des applications livrées ne permet parfois pas au développeur d'avoir une représentation mentale complète de l'applica-

(1) 25 Years of vulnerabilites : 1988-2012, Research Report, Yves Younan - http://www.sourcefire.com/information-center
(2) Base de vulnérabilités gérée par le MITRE (US) – Common Vulnerability and Exposures – cf http://cve.mitre.org/
(3) Open Source Vulnerability DataBase
(4) Open Web Application Security Protocol
(5)Java Authentication and Authorization Service

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

Avec Aruba Cloud,

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



3 hyperviseurs

6 datacenters en Europe

APIs et connecteurs

70+ templates

Contrôle des coûts

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

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

aruba CLOUD

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

MY COUNTRY. MY CLOUD.*

*MON PAYS, MON CLOUD.

tion, et de ce fait omet une partie des interactions possibles.

Le XSS est une attaque par injection qui exploite les interactions entre une application Web et ses utilisateurs. Ces interactions peuvent se traduire par un formulaire (par exemple un formulaire d'inscription), des ensembles clés/valeurs contenues dans des cookies, ou toute autre forme qui implique que l'utilisateur fournisse une information textuelle à l'application. En effet, une attaque XSS consiste à insérer un script malicieux s'exécutant côté client (Javascript, ActiveX, Java, VBScript, etc) à partir d'une interaction utilisateur/application. Cette attaque se déclenche si l'application utilise la donnée fournie par l'utilisateur dans le rendu d'une autre page, et si la donnée fournie n'est pas analysée et vérifiée (on parle de "sanitisation" ou de stérilisation). Le navigateur interprète le script injecté au chargement de la page, et la victime subit l'attaque. Ainsi, il faut être vigilant au XSS à chaque fois que le résultat d'une interaction utilisateur/application est réutilisé dans le rendu d'une autre page.

Les attaques XSS ont toujours le même impact, puisqu'il est lié au langage de script, mais le mode opératoire pour l'injection peut varier. On parle donc de XSS non-persistant si la donnée à injecter est immédiatement rendue, de XSS persistant si la donnée est stockée par le serveur et rendue plus tard et de façon permanente. Une troisième catégorie est parfois évoquée, il s'agit des XSS basés sur le DOM, lorsque la donnée ne passe pas par le serveur. Ce type d'attaque est très dangereux du fait des risques encourus par le client, mais aussi des nombreux vecteurs d'attaques existants, chacun lié à un contexte ou un format/encodage spécifique.

Protéger et tester son application contre les XSS

Lorsqu'il s'agit de se défendre contre les XSS, il existe de nombreuses solutions. Nous les regroupons selon trois familles : la programmation défensive, la détection, et la prévention d'attaques en temps réel.

Programmation défensive

Puisque les failles XSS proviennent d'un manque de stérilisation des données obtenues par le client, adopter une stratégie de programmation défensive est certainement la manière la plus saine et la plus efficace pour se protéger de toute attaque. Il existe deux solutions distinctes : la stérilisation par liste noire, et la stérilisation par liste blanche.

La stérilisation par liste noire consiste à analyser les données reçues à la recherche de caractères ou de mots définis comme dangereux, pour ensuite les supprimer, les remplacer par d'autres caractères ou mots considérés comme sûrs, ou les échapper (pour les caractères spéciaux). Par exemple, la technique de remplacement va scanner une donnée utilisateur à la recherche de la séquence <script>[Quelquechose]</script> pour la remplacer par [Quelquechose] et ainsi contrer l'attaque. Cependant, la stérilisation par liste noire est peu recommandée. Il existe de nombreuses manières d'élaborer une attaque XSS, et ce genre de méthode ne prend généralement pas en compte la multitude de cas à envisager pour être pleinement protégé. La stérilisation par liste blanche raisonne de manière opposée. L'objectif est de n'autoriser que ce qui est considéré comme sûr et en rapport avec la nature de la donnée attendue. Si la donnée représente un nombre positif (l'âge de l'utilisateur), il s'agira de n'autoriser que les nombres positifs, et retourner une erreur dans le cas contraire. De la même manière s'il s'agit de texte brut, seulement autoriser les caractères a-z et A-Z en plus des nombres. Des librairies spécialisées, comme HTMLPurifier, permettent de stériliser correctement toutes les données qu'elles reçoivent. La technique de stérilisation par liste blanche est celle recommandée par l'OWASP foundation pour se protéger contre les XSS.

Détection de failles XSS

Même si une application a été développée en utilisant les techniques de stérilisation de type liste blanche, il reste nécessaire de s'assurer qu'aucune entrée utilisateur n'a été oubliée ou sous-estimée. Pour cela, il existe plusieurs manières.

D'abord, le test de pénétration est certainement la technique la plus utilisée. L'ingénieur de test agit à la manière d'un attaquant, et tente d'injecter des scripts malicieux dans les points d'entrée de l'application, avec l'aide d'outils comme les proxys intrusifs (Burp Suite, ZAP, etc). Cette technique est très efficace, mais elle est dépendante des compétences de l'ingénieur de test. De plus, le test de pénétration représente un coût en temps important, ce qui pousse l'ingénieur à ne traiter que certaines parties de l'application qu'il considère comme les plus à risque, suite à une analyse de risque.

Ensuite, une autre technique est basée sur l'analyse statique du code source (automatiquement ou non) de l'application. L'objectif est de repérer les données d'entrées liées à un « évier » (« sink » en anglais), c'est à dire une méthode qui va rediriger la donnée vers le navigateur de l'utilisateur (par exemple la méthode echo() en php). Lorsqu'une telle donnée d'entrée est découverte, l'ingénieur de test ou l'algorithme va vérifier que des méthodes de stérilisation sont en place. Cette technique est efficace, mais encore une fois coûteuse en temps, et est fortement liée à un langage en particulier. De plus, si l'analyse est faite manuellement, le grand nombre de lignes de code favorise l'apparition d'oublis dus au facteur humain.

Enfin, il existe des outils semi-automatiques qui réalisent des tests de pénétration, tout comme l'ingénieur de test. L'attrait vers ces outils se justifie par leur capacité à attaquer l'application avec des milliers de vecteurs en quelques minutes. D'après les études récentes, ces outils appelés scanners de vulnérabilité obtiennent de très bons résultats pour une majorité de types de vulnérabilité, dont le XSS non-persistant. Cependant, ces outils ne permettent pas de détecter les vulnérabilités complexes nécessitant une connaissance logique de l'application (ces outils n'ont pas d'intelligence).

Prévention d'attaques en temps réel

La prévention d'attaque consiste à placer un « gardien » entre le client, l'application et le serveur, pour observer ce qui transite et inhiber toute tentative d'attaque. Ce gardien peut être installé dans le navigateur du client, ou du côté serveur (Pare-feu Applicatif, ou Web Application Firewall). La technique de stérilisation est ensuite similaire à la programmation défensive.

Conclusion

Comme nous avons pu le voir, la prise en compte de la sécurité tout au long du cycle de vie d'un système peut réduire les vulnérabilités et les risques de hacking mais il faut ...

- Se donner les moyens de prendre en compte efficacement la sécurité
- Sensibiliser aux risques de sécurité et à comment s'en protéger
- Protéger les systèmes vulnérables existants (antivirus, firewalls, chiffrement, etc.)
- Former les développeurs, les testeurs et les autres acteurs (utilisateurs finaux, administrateurs, etc.) au respect des règles de sécurité au quotidien

Sur les aspects sécurité, un syllabus expert « testeur sécurité » est en cours d'élaboration par l'ISTQB(6). Il sera proposé en France par le CFTL (7).

Alain **Ribault**,
Directeur Technique de la société KEREVAL, membre actif CFTL et corédacteur du syllabus expert sécurité ISTQB.

Bruno **Legeard**,
Professeur à l'Université de Franche-Comté, Co-fondateur Smartesting, Membre du Bureau du CFTL.

Alexandre **Vernotte**,
Chercheur en Sécurité, Institut FEMTO-ST / Université de Franche-Comté.

(6) International Software Testing Qualifications Board
(7) Comité Français des Tests Logiciels

Les outils des Décideurs Informatiques

Vous avez besoin d'info sur des sujets d'administration, de sécurité, de progiciel, de projets ? Accédez directement à l'information ciblée.

www.solutions-logiciels.com

☐ **OUI, je m'abonne** (écrire en lettres capitales)

Envoyer par la poste à : Solutions Logiciels, service Abonnements - 17 route des Boulangers 78926 Yvelines cedex 9 - ou par fax : 01 55 56 70 20

***1 an : 60€** (Tarif France métropolitaine) - Autres destinations : CEE et Suisse : 70€ - Algérie, Maroc, Tunisie : 75€ , Canada : 90€ - Dom : 85€ Tom : 110€ - 5 numéros par an.*

☐ M. ☐ Mme ☐ Mlle Société .....

Titre : ..... Fonction : ☐ Directeur informatique ☐ Responsable informatique ☐ Chef de projet ☐ Admin ☐ Autre .....

NOM ..... Prénom .....

N° ..... rue .....

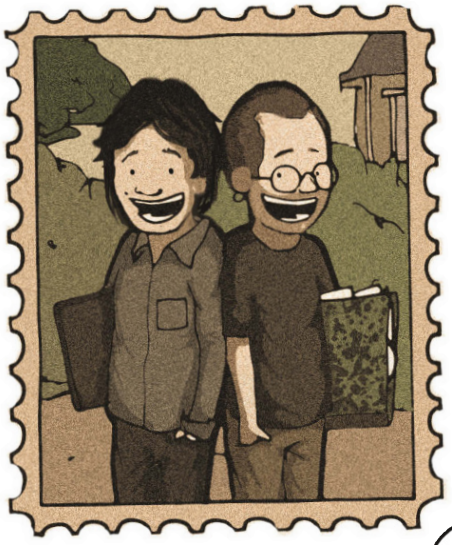
Complément .....

Code postal : | | | | | | Ville .....

Adresse mail .....

☐ *Je joins mon réglement par chèque à l'ordre de SOLUTIONS LOGICIELS* ☐ *Je souhaite régler à réception de facture*

Editeur : K-Now sarl au cap de 8000€ siège social : 21 rue de Fécamp, 75012 Paris



Aux origines de CommitStrip

Ce mois-ci un développeur du mois un peu particulier. Nous allons nous plonger au coeur de CommitStrip que vous connaissez sans doute. Chaque strip raconte des "mésaventures" de codeurs, de geeks. Toutes ces histoires sont vraies ! Rencontre avec Thomas Guenoux et Etienne Issartial.

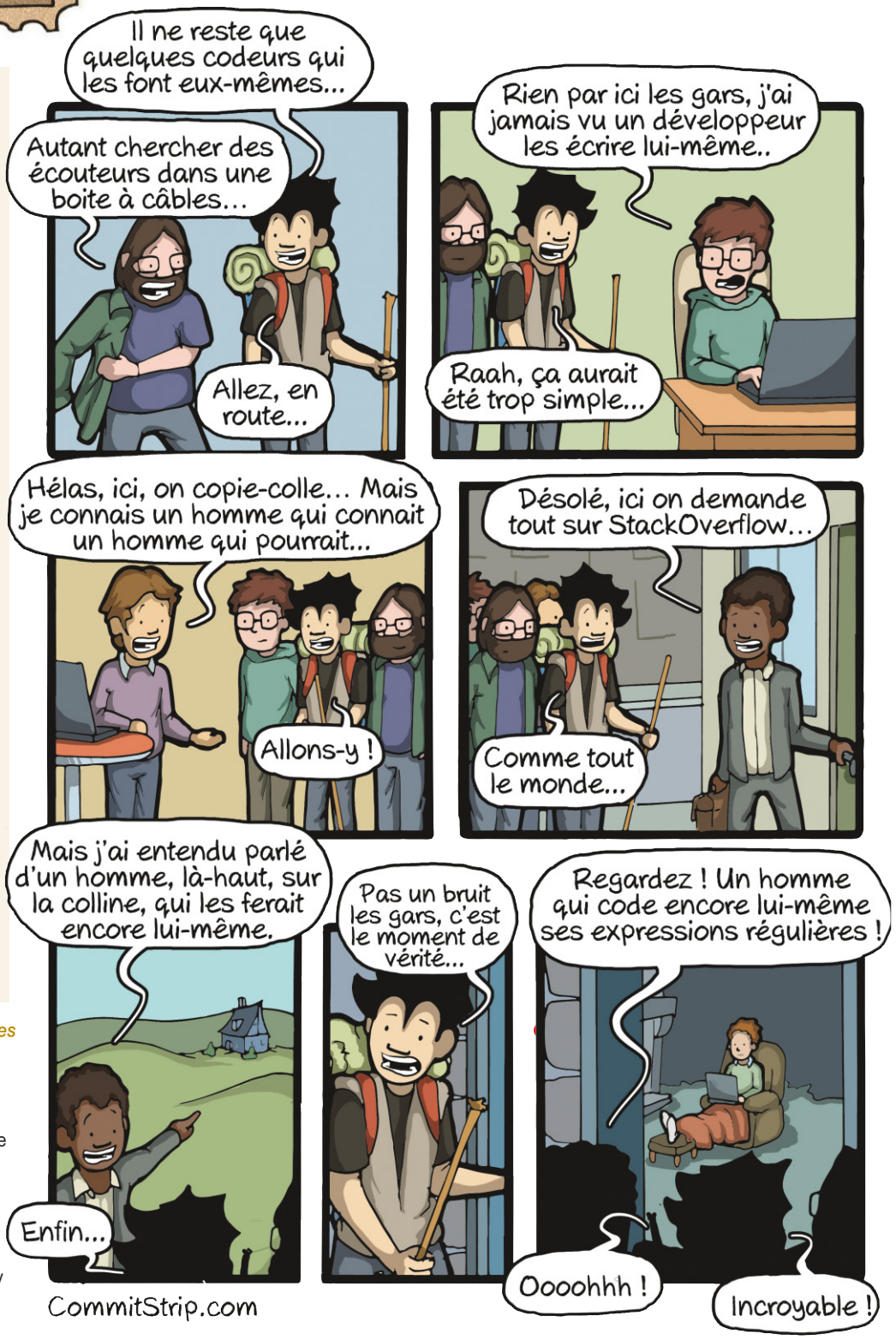
Bios rapides

Thomas Guenoux

Thomas Guenoux est un entrepreneur spécialisé dans les médias sociaux, maîtrisant aussi bien la création, les technologies web, que les processus projet. Pionnier du développement des réseaux sociaux en France dès 2006 en lançant Cancoon, il co-fonde en 2007 l'agence KRDS, première agence spécialisée en social media marketing et aujourd'hui leader en Europe et en Asie. Il y occupe aujourd'hui la Direction de la Production, dirigeant un studio primé et une équipe de développeurs web ayant produit près de 1000 projets. Thomas a également fondé CommitStrip en 2012, le blog BD de référence pour les codeurs web.

Etienne Issartial

Etienne Issartial est dessinateur de presse et illustrateur. Issu des Beaux Arts de Nantes, il lance Blogobic, son premier blog de dessin de presse, avant de rejoindre l'aventure CommitStrip, le blog BD bien connu des codeurs. Il illustre egalement l'actualité dans la presse percheronne, sa région natale.



Quels ont été vos parcours dans les technologies et le développement ?

CommitStrip est porté par Etienne, dessinateur de presse, et moi-même (Thomas), co-fondateur et directeur de la production de KRDS, l'agence de marketing social bien connue. Du coup, j'apporte les anecdotes et les histoires de codeurs Etienne les met en scène et en image. Etienne n'y

connait pas grand chose en développement, mais comme on est amis de longue date, je l'ai un peu contaminé avec ma passion pour le web ! C'est donc moi le geek du duo : développeur de sites web depuis le plus jeune âge, j'ai étudié le marketing et les systèmes d'informations à Telecom SudParis avant de cofonder KRDS.

Comment observez-vous l'évolution perpétuelle du monde informatique, du développement, des technologies ?

On attend avec impatience les cyber humains ! L'homme est tellement faillible, augmentons-le grâce à la technologie et au code ;) Blague à part, on aime bien se dire que le progrès technologique est avant tout porté par des hommes qui passent des fois leurs vies entières pour faire avancer la science. Et CommitStrip, c'est un peu un hommage à ces hommes et ces femmes qui restent souvent dans l'ombre, passionnés par la création et le développement, et qui préparent pour nous tous un futur meilleur.

L'ultime réponse est 42 mais quelle est donc l'ultime question ?

Pourquoi le codeur est un ninja ?

Racontez-nous la naissance de CommitStrip ? Etait-ce un pari perdu durant une nuit blanche à coder tout et n'importe quoi ?

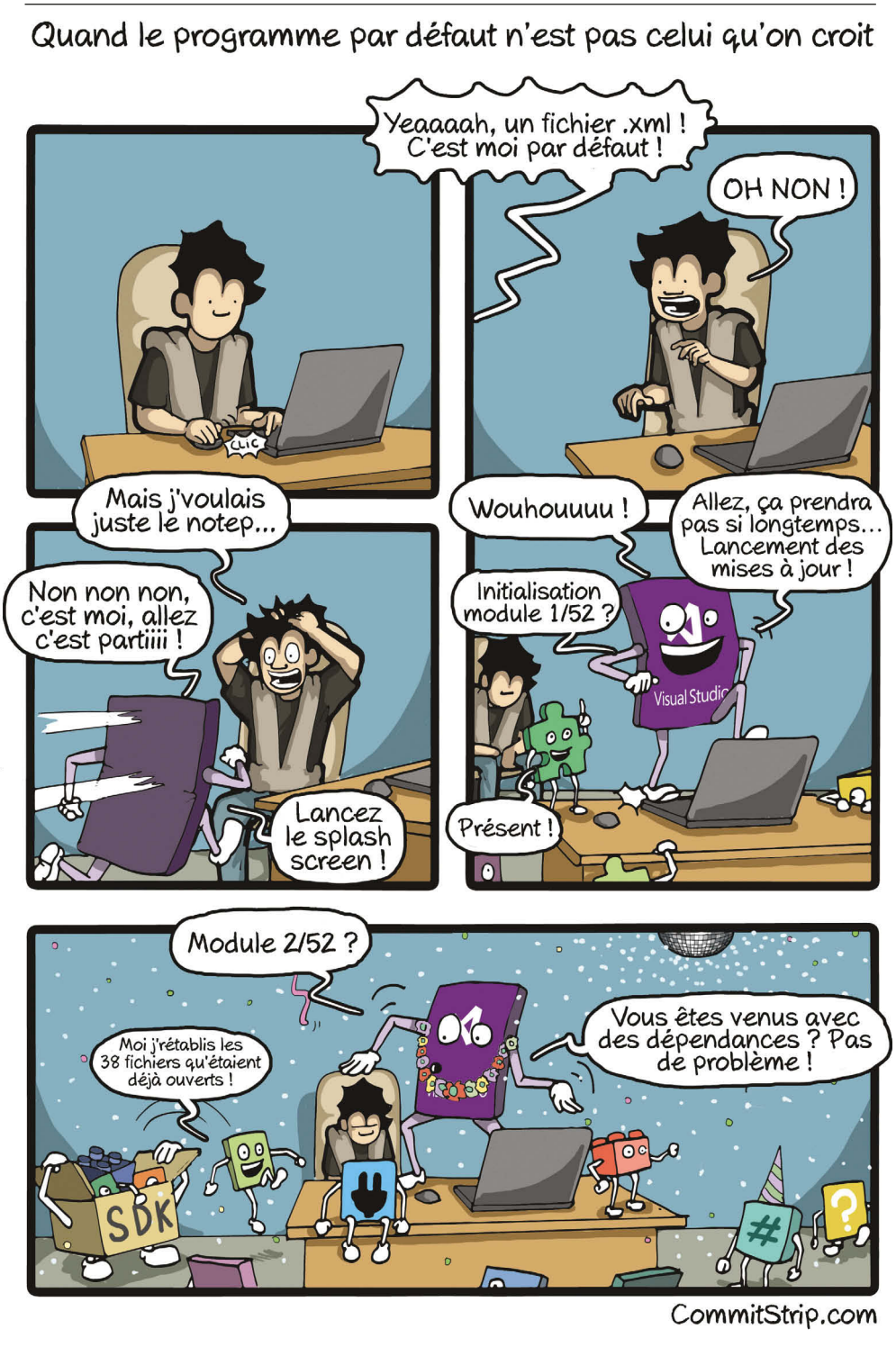
Je suis directeur de la production de l'agence conseil en social media KRDS. C'est une agence qui s'est bien développée depuis six ans, et j'ai donc eu à faire à pas mal de projets, et qui dit projets dit bugs et anecdotes de codeurs ! Un jour, j'ai demandé à Etienne, un ami d'enfance dessinateur de presse, s'il voulait participer avec moi à un projet un peu fou de dessiner ces anecdotes de codeurs sous forme de blog BD. Même s'il n'est pas codeur, mes blagues l'ont fait marrer, alors on s'est lancé !

Quels sont les défis de Commitstrip : l'inspiration, le dessin, les textes ?

Clairement, nos plus gros défis sont les idées et la mise en scène ! Aujourd'hui, le style est affirmé, les personnages clairs, le ton bien défini. Par contre, il faut savoir se renouveler, traiter des sujets d'actualités comme des sujets de fond de manière drôle et pertinente. On a une longue liste d'idées en attente, le plus gros challenge est de trouver la bonne mise en scène et le bon mot, bien équilibré pour être compréhensible mais pas simplet. Après près de 500 planches, on sait maintenant ce qui marche ou pas, alors on arrive souvent à viser juste !

Toutes les situations ont-elles été vécues ?

Pratiquement toutes ! Et à vrai dire, c'est un peu notre créneau, la true story de codeur ;)



Comment se déroule la création d'un dessin

Ça va très vite !

- **14h** : brainstorming pour trouver l'idée
- **Avant 15h** : rédaction des dialogues
- **15h-18h** : dessins et mise en couleur
- **18h** : retravail des dialogues et traduction
- **18h30** : mise en place des bulles
- **A partir de 19h** : publication !

Retrouvez chaque mois CommitStrip dans Programmez!

Quelques mois avec le Surface Pro 2

Les tablettes et moi, une longue histoire ...

Tout a commencé en regardant les premiers épisodes de Star Trek où le capitaine Kirk recevait ses rapports sur une tablette (LCARS), qui servait en même temps à bien d'autres tâches... Un rêve de gosse. En attendant la sortie officielle de l'LCARS, j'ai successivement utilisé un SimPAD (une tablette sous Windows CE 4.2), Samsung Q1 (UMPC), iPad, Microsoft Surface RT, Acer Iconia W3-810. Aujourd'hui j'utilise, en dehors d'un laptop, une Dell Venue 8 Pro et une Surface Pro 2 (processeur i5-4200, 8GB de mémoire & 256 GB de SSD). Mon quotidien se compose d'environ 70 % de développement (Visual Studio, Xamarin, ...) et 30 % de tâches administratives (MS Office 2013 & co) et commerciales en clientèle. Mon temps se partage entre deux bureaux.

Au quotidien

En termes de confort d'utilisation on est à l'aise avec la Surface Pro 2. Elle surclasse mon laptop qui est pourtant équipé d'un i7. Avec un SSD et un processeur Intel i5 de dernière génération la Surface Pro 2 est particulièrement véloce. J'utilise les grands classiques Office 2013 et Visual Studio 2013. Pour plus de confort j'utilise un clavier externe. Je trouve par contre que les logiciels modern UI (mail, agenda, One Note, ...) sont bien plus adaptés à une utilisation au doigt qu'Office 2013. Je n'ai rencontré aucun problème lors des installations des différents logiciels, que ce soit à travers le réseau ou à travers un lecteur DVD USB externe. Ce qui est un peu déroutant au début, ce sont les changements de DPI entre écran externe et écran interne. Dans le passé, et c'était un peu stressant, la Surface avait une tendance à l'auto réparation. De temps en temps au démarrage, elle se mettait à brouter quelques minutes sur les écrans de ..., disait qu'il n'y avait rien à réparer et quand on redémarrait tout se passait bien. Je n'ai jamais rien perdu. Ce problème semble néanmoins résolu par une des dernières mises à jour du firmware.

Plutôt « Type Cover » que « Touch Cover » ?

La différence en termes de vitesse de frappe est énorme. La « Touch Cover » se situant plutôt au même niveau que le clavier on screen, alors que la « Type Cover » à le même rendu qu'un clavier de portable. Seul bémol, comme l'assemblage n'est pas rigide il faut toujours avoir une surface plane sous le coude.

Mes configurations

Les deux configurations qui ont fait leurs preuves pour moi ces derniers mois sont :

En déplacement

Surface Pro 2 + Type Cover + Logitech Ultrathin Touch Mouse (oui, je suis réfractaire au touchpad et on ne peut pas tout faire au doigt sur l'écran) sur la route. D'ailleurs, en dehors d'un prix exorbitant la souris Logitech (environ 70 €) elle dispose de deux atouts majeurs. D'une part elle se recharge très rapidement sur un port USB, d'autre part, elle peut être couplée avec deux maîtres.

A la maison

Surface Pro 2 + Station d'accueil HP 3005pr qui exploite l'USB3 et me donne en plus d'être un hub USB & USB3, un port Gigabit Ethernet et une sortie HDMI pour un deuxième écran (environ 140 €). J'utilise une souris et un clavier Microsoft en Bluetooth, ainsi qu'un écran 24 pouces branché sur le port HDMI de la station d'accueil. Je n'ai pas testé la station d'accueil de Microsoft, mais je trouve personnellement, qu'à l'âge de l'USB3, une station d'accueil dédiée ne se justifie plus et je dois dire qu'en vue de l'utilisation des derniers mois je ne regrette pas mon choix. L'affichage est très performant (je ne l'ai pas testé pour jouer des jeux) et très confortable à l'utilisation. On branche le câble USB et tout fonctionne. Petite astuce pour l'installation des drivers : utilisez le driver de DisplayLink qui est plus récent et pose moins de soucis lors de l'installation. Dans mon home-office la surface a remplacé le classique desktop.

Et Visual Studio ?

Dans le cadre de mon métier, le développement d'application mobiles pour les pros (logistique, ...) et le grand public, j'utilise la Surface Pro 2 tous les jours, ou presque. Notamment pour le développement des applis Modern UI; c'est très pratique parce que je peux tester les applis sur l'écran interne en tactile et sur l'écran externe avec la souris sans être obligé de faire des grandes manipulations. Il va de même pour le simulateur Windows Phone qui peut être utilisé directement avec les doigts comme un téléphone physique. Le problème lorsqu'on travaille avec la Surface Pro 2 sur Visual Studio est la résolution DPI. A résolution égale de 1920 x 1080 entre mon écran externe de 24 pouces et la Surface Pro, je vois respectivement 45 lignes de code source dans Visual Studio sur l'écran externe et 26 sur la Surface Pro 2. On peut, bien entendu jouer avec les réglages pour avoir plus de lignes sur l'écran, elles seront parfaitement nettes, mais on aura l'impression de travailler sur du micro film sans la loupe. Ce qui vaut pour les lignes de source est également vrai pour le design d'écran. On peut travailler sur un design d'écran de Windows Phone mais pour la conception d'un écran Modern UI ce n'est vraiment pas adapté. On est plus proche de la neurochirurgie que du développement. Donc pour le développement c'est super avec un écran externe, sinon elle est un peu petite mais cela dépanne ; l'utilisation d'un clavier externe est obligatoire.

Conclusion

Le point négatif, comme pour toutes les tablettes de sa taille d'aujourd'hui, reste son poids. On n'est pas encore au niveau d'un bloc note. D'un autre côté, lors de la prise en main, c'est également son poids qui inspire confiance. Son pied intégré, qui permet deux inclinaisons différentes, est très agréable au quotidien.

En fonction de l'utilisation l'autonomie est entre la demi-journée en utilisation intensive et plus d'une bonne journée de travail en mode réunion ou séminaire comme les TechDays. Depuis les mises à jour du début de l'année l'autonomie est assez confortable. La Surface Pro 2 n'est peut-être pas encore la LCARS du capitaine Kirk, mais elle commence à s'en approcher dans sa polyvalence.

Michael **Engelmann**

Les liens

HP 3005pr (http://www8.hp.com/fr/fr/products/oas/product-detail.html?oid=5303779#!tab=features)
Logitech Ultrathin Touch Mouse (http://www.logitech.fr/fr-fr/product/ultrathin-touch-mouse-t630)
Driver DisplayLink (http://www.displaylink.com/support/downloads.php)
LCARS (http://fr.wikipedia.org/wiki/LCARS)

PCSoft organise son tour de France

L'éditeur de WinDev organise son tour de France depuis la mi-mars. Sur une demi-journée, vous allez pouvoir découvrir les dernières nouveautés des versions 19, rencontrer les experts PCSoft : 1er avril - Paris, 2 avril - Lille, 3 avril - Bruxelles, 8 avril - Strasbourg, 9 avril - Genève, 10 avril - Lyon, 15 avril - Marseille. Pour en savoir plus : http://www.pcsoft.fr

Cloud Computing World Expo

Les 9 et 10 avril, le Cloud Computing sera à l'honneur à Paris. De nombreuses conférences seront organisées sur les deux jours : Open Cloud Forum, le poste connecté, comment choisir son Cloud, Opex ou Capex, le forum OW2... Site : http://www.cloudcomputing-world.com

ARA Developers' conference

Les 15 et 16 avril prochains Google organise une grande conférence développeur (en ligne) pour un projet assez fou : un smartphone modulaire. Ce projet est issu de Motorola, le projet Ara. Cette conférence va se concentrer sur le kit de développement, la plateforme matérielle, la roadmap. Site : http://www.projectara.com/#ara-developers-conference

Devoxx France 2014

L'évènement du monde Java se déroulera du 16 au 18 avril prochains à Paris. Ce sont des conférences de 3h et des ateliers. En fin de journée, des sessions courtes se succéderont. Les 17 et 18 avril, c'est la conférence proprement dite avec les multiples conférences.
Site : http://www.devoxx.fr

ScrumDay 2014

L'évènement phare de l'agilité et de la méthode Scrum en France aura lieu cette année les 10 et 11 avril à Disneyland Paris. Le Scrum User Group France est toujours le maître d'œuvre. Site : http://www.scrumday.fr

Hackito Ergo Sum

Une des principales conférences hacking françaises revient les 24, 25 et 26 avril, à Paris. http://2014.hackitoergosum.org

AgoraCMS

25 avril. Une journée 100 % CMS. On y parlera Drupal, Wordpress, eZ, ametys, projets, développements, marketing. Plus de 20 conférences seront proposées toute la journée. Tous les détails de la journée sur http://www.agoracms.com

CONFÉRENCES DES JUG

Alpes JUG : soirée Java 8 Lambdas and devices, Grenoble, 3 avril. Site : http://www.alpesjug.org

Toulouse Jug : Java 8 Lambdas and devices, 8 avril. http://toulousejug.org

Jug Lyon : atelier Scala, 9 avril, à partir de 19h. Site : http://www.lyonjug.org

Jug Bordeaux : soirée Java 8 Lambdas and devices, 9 avril. Site : http://bordeauxjug.org/meetings

Tours Jug : soirée quickies, 9 avril. Site : http://toursjug.cloud.xwiki.com

Riviera JUG – Sophia-Antipolis : soirée Lambda, sex and sun... 13 mai. Site : http://rivierajug.org

Unity3D vs UDK vs CryEngine : quel moteur choisir ?


D.R.

Avec la croissance du chiffre d'affaires de l'industrie, on ne peut que constater que le jeu vidéo a le vent en poupe. La montée en popularité du courant indépendant, favorisée par des plateformes telles que Steam Greenlight, Desura, Kickstarter et autres qui permettent aux développeurs de financer et distribuer leurs créations, en est un bon indicateur.

Dans la multitude de moteurs de jeux disponibles, certains sont présents depuis près de 20 ans, d'autres se sont fait connaître assez récemment. Parmi les problématiques actuelles auxquelles ceux-ci se doivent de répondre, on citera le déploiement multiplateformes, la facilité de prise en main, la puissance du moteur de rendu, ou encore le prix.

Dans cet article, nous présenterons différents moteurs et tenterons de répondre à ces problématiques, afin de guider le lecteur dans son choix quel que soit son niveau d'expérience et son lien avec le milieu du jeu vidéo.

Notre choix s'est porté sur trois moteurs majeurs de l'industrie vidéo ludique : Unity3D, Unreal Development Kit et CryEngine. Nous exposerons les principales caractéristiques de chacun, avant de les comparer entre eux.

UNITY3D

Présentation

Développé par Unity Technologies, Unity3D est avant tout un moteur de jeu multiplateformes destiné aux développeurs aussi bien amateurs que professionnels. Au moment de la rédaction de cet article (janvier 2014), l'éditeur est disponible sur Windows et Mac OS X dans sa version 4.3.4.

Déploiement

La magie de Unity3D opère au moment du déploiement. Une fois le jeu prêt à être testé, le logiciel permet en quelques clics de sélectionner la plateforme désirée et de créer les fichiers nécessaires pour faire tourner votre jeu sur ordinateur (Windows, Mac ou Linux), mais aussi dans le Web Player (une extension pour les navigateurs web Chrome, Firefox, Internet Explorer et Opera), sur smartphones (Android, iOS, Windows Phone et Blackberry) ainsi que la plupart des consoles actuelles. Il suffit de répéter l'opération pour chacune des plateformes que vous souhaitez cibler.

Technologies

Pour fonctionner sur toutes ces différentes plateformes, Unity3D compile le jeu en utilisant les technologies natives appropriées ; on retrouve donc Direct3D pour Windows et Xbox 360, de l'OpenGL pour Mac, Windows, Linux et PS3, ainsi qu'OpenGL ES pour Android, iOS et NaCl (Native Client de Google Chrome) et des APIs propriétaires pour la Nintendo Wii. Côté shaders, HLSL et Cg sont supportés, et l'on peut également utiliser la technologie ShaderLab de Unity qui permet de travailler sur les effets de manière plus « haut-niveau ».

Showcase

De nombreux jeux Unity3D ont vu le jour récemment, tels qu'Endless Space (PC, Mac, Unity Web Player), Deus Ex: The Fall (Android, iOS), Wasteland 2, Shadowrun Online... On trouve également du serious gaming (« jeux » destinés à des professionnels dans un objectif de formation) ou des jeux de casino, démontrant l'intérêt de différentes industries pour cet outil.

Version gratuite et version professionnelle

Unity3D est disponible en version gratuite (parfois appelée « Unity Indie ») et en version professionnelle (« Unity Pro »); la version payante apportant des fonctionnalités et améliorations utiles sans forcément être indispensables (ombres dynamiques, support de la vidéo, « splash screen » personnalisé, textures 3D...). Une limitation toutefois : si vous êtes une entreprise dont le chiffre d'affaires a dépassé les 100 000 dollars l'année fiscale précédente, vous devez utiliser la licence Unity Pro. Celle-ci vous coûtera de base 1425 euros (ou 71 euros par mois dans sa version abonnement) par personne ; c'est également le montant à ajouter pour chaque module permettant de déployer votre jeu en version Pro pour les différentes plateformes mobiles Fig.1.

Le logiciel

Unity3D, c'est d'abord un logiciel très complet aux capacités complexes ; mais ne laissez pas son aspect légèrement austère vous impressionner car c'est surtout une solution très simple à prendre en main. L'interface

est découpée en fenêtres que l'on peut redimensionner, déplacer, détacher, bref que l'on peut manipuler à volonté pour personnaliser son environnement de travail. La fenêtre *Scene* permet de se déplacer et de manipuler les objets de la scène courante, même pendant que le jeu tourne. Celle nommée *Game* permet de visualiser la scène à travers la caméra active et de jouer, directement depuis l'éditeur, dans n'importe quelle résolution. La fenêtre *Project* liste la totalité des scripts, textures, scènes, sons, modèles 3D et autres «assets» présents dans le projet, alors que *Hierarchy* liste uniquement ceux présents dans la scène courante. La fenêtre *Inspector* présente toutes les informations de l'objet sélectionné dans la fenêtre *Scene* (nom, tag, position, composants et leurs propriétés...), alors que la *Console* affiche les erreurs et avertissements liés aux scripts et objets de la scène. Enfin, le *Profiler* - une fonctionnalité réservée à la licence Pro - permet d'analyser le temps d'exécution de chaque partie du moteur durant le jeu.

Le moteur

Côté moteur, il est important de comprendre le fonctionnement des différentes classes pour tirer parti des avantages de la programmation orientée composants. Chaque objet d'une scène est un GameObject, auquel peuvent être attachés des composants à volonté. Le composant Transform, présent de base dans tout GameObject, définit la position, la rotation et la taille de l'objet et permet le «parenting» (l'affectation d'un GameObject en tant que parent ou enfant d'un autre GameObject). La liste de composants permet de gérer aisément la physique (Rigidbody, Colliders), les animations, la GUI, le son, l'intelligence artificielle, et, bien entendu, le rendu graphique. Pour définir des comportements spécialisés, on utilisera un composant Script se basant sur la classe MonoBehaviour. On hérite donc des fonctions de base de cette classe, que l'on n'a plus qu'à remplir (souvent au minimum les méthodes Start et Update), afin de facilement contrôler les autres composants de l'objet auquel le script est attaché et de changer ses valeurs directement depuis l'interface du logiciel (variables membres publiques). Le scripting, basé sur Mono, peut se faire en C#, en Boo script ou en JavaScript et Unity3D est livré avec le logiciel MonoDevelop.
Le moteur facilite également la gestion des entrées en tentant d'unifier clavier, manette et écran tactile.

Assets et extensions

La force du logiciel vient aussi de sa simplicité : tout est très graphique, beaucoup d'actions se résument à un simple glisser-déposer (importer des textures, charger des add-ons, ajouter un composant tel qu'un script ou un matériau...) et même si tout est accessible depuis le code, le scripting est grandement facilité, ce qui permet à des amateurs ne disposant que des bases en programmation de réaliser un jeu simple. Même si le moteur est très orienté 3D, des outils officiels intégrés pour la 2D ont été récemment rendus disponibles. L'*asset store*, qui regorge de packages gratuits ou payants, est intégré au logiciel et permet en quelques clics d'importer des modèles 3D, textures, effets de particules, scripts, sons, extensions et même parfois des projets complets.

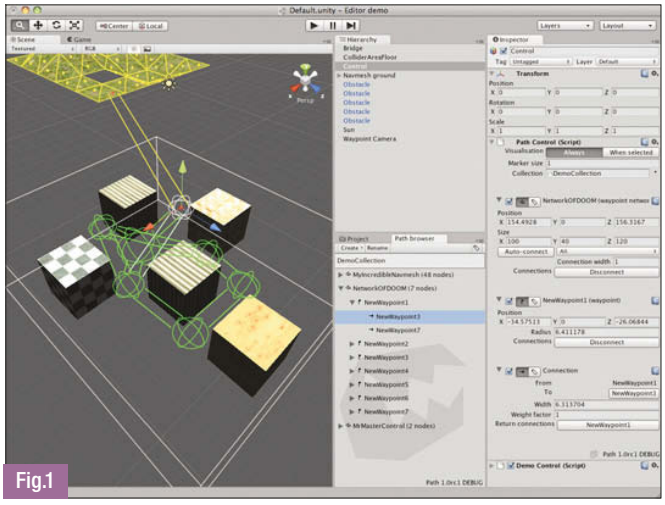


Fig.1

L'interface de l'éditeur Unity3D

Communauté et ressources

Enfin, les ressources et la communauté sont des facteurs importants dans la réussite d'Unity3D. Le site officiel fournit une documentation souvent complète, mais si votre question reste sans réponse vous pouvez toujours vous tourner vers le forum, le site d'aide (answers.unity3d.com), les tutoriels officiels, le support premium ou les centaines d'autres sites et blogs maintenus par la communauté, forte de deux millions de développeurs.

CRYENGINE

Présentation

CryEngine, actuellement dans sa version 3 (CE3) et disponible gratuitement depuis octobre 2011, est un moteur de jeu propriétaire créé par la firme allemande Crytek. A l'origine développé pour une démonstration technologique de nVidia, CryEngine est un moteur spécialisé dans les FPS qui a été transposé par la suite pour divers jeux dont les non moins populaires Far Cry et Aion : The tower of Eternity. Pour développer avec CryEngine3 la configuration recommandée minimale est :

- Windows XP
- Processeur Dual Core 2.0 GHz
- CPU 64 bits
- 4 GB RAM
- 5 GO d'espace disque
- Support Shader 3.0, soit : GeForce 6600+ ou GFX d'ATI/AMD équivalente

Plateformes supportées

Les jeux créés sous CryEngine3 peuvent être portés sous Windows, Playstation 3, Xbox 360 & Wii U. Cependant, la version 4 qui sortira bientôt supportera les consoles de nouvelle génération Playstation 4, Xbox One & Wii U. A noter qu'une version allégée pour Android et iOS est également prévue cette année.

Showcase

On retrouve CryEngine3 majoritairement dans des FPS tels que Crysis et HomeFront. Mais on le retrouve également dans d'autres types de jeux de la firme CryTek tel que Ryze Son of Rome

Technologies natives

Le moteur de CryEngine est écrit en C++, une surcouche en Lua est utilisée pour les interactions et le scripting.

Le logiciel

Nous retrouvons plusieurs interfaces distinctes au sein de CryEngine, dont l'éditeur de contenu qui permet de créer les scènes en s'appuyant sur des assets existants ou en créant les siens.
D'une façon comparable à Blender, l'éditeur permet en un clic de passer de l'éditeur au design actuel : « What You See Is What You Play Feature » Fig.2.
Le pack du SDK contient également une fenêtre nous permettant de tester nos jeux. Celui-ci propose un panel permettant de tester en solo, en multijoueur, et de définir différents paramètres allant des contrôles claviers, en passant par les sons jusqu'aux paramètres graphiques du jeu.

Versions gratuite et version professionnelle

CryTek propose une version gratuite pour un usage non commercial, dans le cas contraire, les tarifs sont définis dans un accord avec Crytek en fonction de la situation de votre entreprise et de l'utilisation désirée (développeur indépendant, studio de développement...)

Fonctionnalités

CryEngine s'appuie sur un panel complet de fonctionnalités, on retrouve notamment la gestion des cinématiques inverses, un module d'intelligence artificielle, l'utilisation de FMOD pour l'environnement sonore, un client réseau et serveur, la gestion des shaders (textures animées, per pixel shading, réfraction), les ombres pré-calculées ou dynamiques, le brouillard (volumétrique et par couche), gestion du polybump (autonome ou intégré à 3DSMax, Maya...), le scripting en Lua (surcouche au C++ pour les actions telles que les paramètres du jeu, le chargement des images).

Scripting

Lua est une extension de langage de programmation embarquée dans un hôte client. L'hôte va donc pouvoir invoquer des fonctions de Lua.

Assets

Afin d'importer des objets externes dans l'éditeur bac à sable de CryEngine, il est nécessaire de passer par l'installation de certains plugins afin d'assurer la compatibilité du logiciel désiré, parmi ceux-ci :

- Pour la modélisation et les animations, CryEngine3, supporte 3DS MAX, Maya et XSI via l'ajout de plugins.
- Pour les images, le format .tif est supporté via le plugin CryTIF.

Pas d'asset Store officiel à l'heure actuelle mais il est toujours possible de trouver des modèles 3d sur des sites tels que http://turbosquid.com.

Sites officiels http://mycryengine.com/

Regroupe toutes sortes de renseignements globaux sur l'utilisation du moteur. http://docs.cryengine.com/

Documentation technique du moteur de jeu, présente sous forme de wiki.

Communauté

http://www.crydev.net/

Plateforme officielle de la communauté des développeurs pour CryTek. En outre, on retrouve des tutoriaux sur Internet faits par des développeurs indépendants. En revanche, si vous souhaitez tenter l'aventure CryEngine, il est fortement recommandé de maîtriser l'anglais, très peu de ressources sont disponibles en français à l'heure actuelle sur le sujet.

UNREAL DEVELOPMENT KIT

Présentation

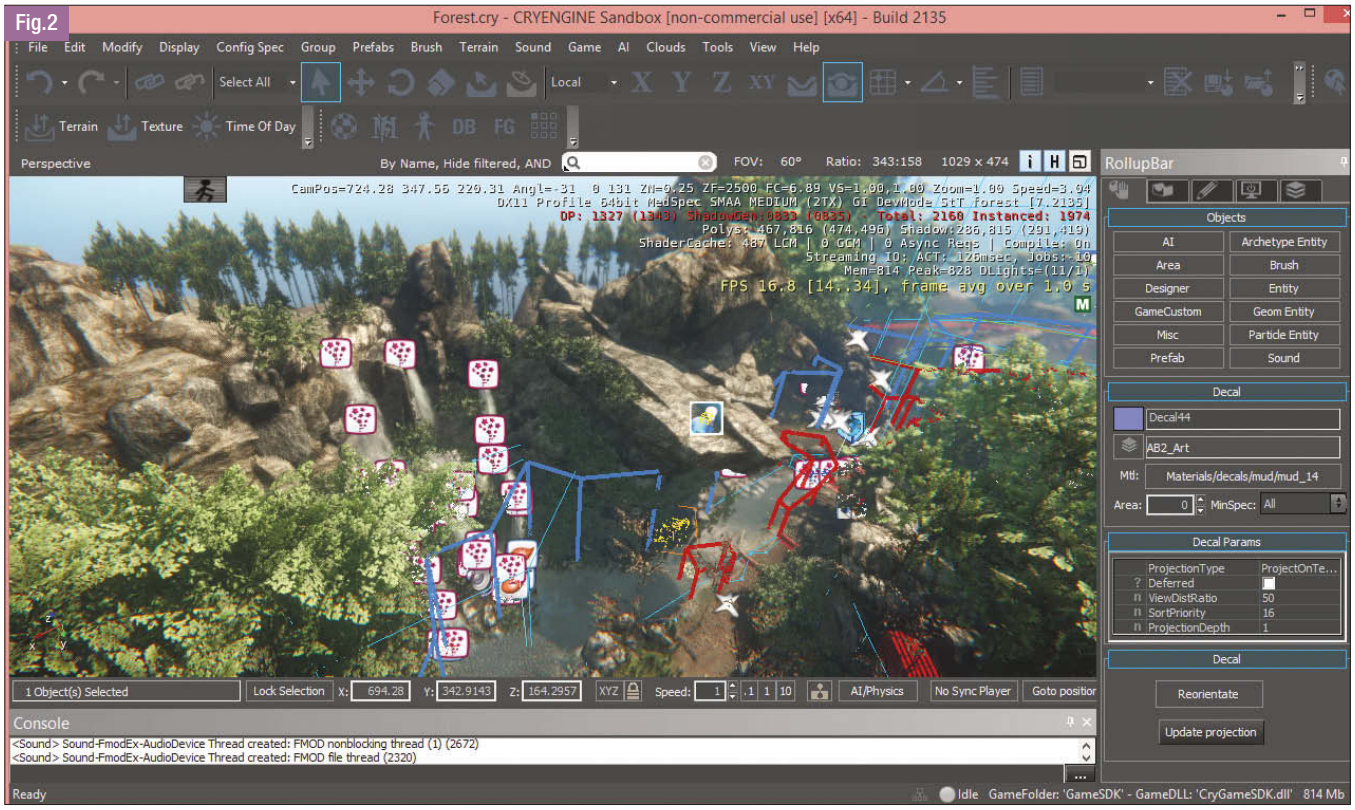
L'Unreal Engine 3 est probablement le moteur le plus utilisé de la génération de console Xbox 360 / PS3. Créé par Epic Games, le moteur fit ses débuts avec le jeu Unreal Tournament, avant d'être utilisé sur de nombreux titres sortis ces dernières années comme Mass Effect, Bioshock Infinite, Gears of War ou Batman Arkham... Difficile d'ignorer l'impact du moteur sur le marché du jeu vidéo tant la liste de jeux l'utilisant est importante.

Afin de permettre à tous les développeurs de se former aux technologies d'Epic Games, le studio propose une version gratuite de son moteur de jeu, nommé UDK (Unreal Development Kit).

Dans l'UDK, tous les outils utilisés par les grands studios sont disponibles gratuitement pour tout développeur souhaitant les prendre en mains. Les seules différences avec la version payante sont l'absence du code source du moteur de jeu, ainsi que l'impossibilité de publier sur une plateforme autre que Windows ou iOS. La publication d'un jeu payant est possible après s'être acquitté du prix de la licence (99$). Si le jeu créé dépasse les 50 000 dollars de revenus, il faut payer une redevance à Epic Games à hauteur de 25% sur le surplus. La configuration minimale correspond à celle d'un pc bas de gamme :

- Windows XP 32 bit
- Processeur 2.0 GHz

Fig.2



Aperçu de l'éditeur de contenu.

- 2 GB de RAM
- Carte graphique compatible SM3
- 3 Go d'espace disque

Le logiciel

L'interface de l'UDK est claire : on retrouve plusieurs vues représentant la scène sous différents angles, un sélecteur de contenu, une fenêtre pour gérer l'animation des objets... Il est également possible de lancer le jeu depuis l'éditeur pour tester ses dernières modifications.
Contrairement au moteur Unity3D dans sa version gratuite, l'éclairage dynamique est fourni nativement avec l'UDK. Si le moteur ne possède pas de système de plugins tel que celui de son concurrent, il est tout de même fourni avec des composants permettant de faciliter le travail des développeurs :

- Scaleform pour la création d'interface utilisateur
- Kismet pour construire des scripts à travers l'interface graphique
- MaterialEditor pour l'édition d'objets, de textures et de forme
- Matinee pour l'animation d'objet
- Cascade pour les effets de particules (feu, brouillard, pluie, poussière) Fig.3.

Le moteur

Le code source du moteur n'étant pas accessible dans la version gratuite, il faut passer par le langage UnrealScript pour décrire les évènements et le comportement du jeu. Le langage s'inspire du Java et du C++.
Pour les développeurs les plus expérimentés, il est possible de créer des DLL en C++ qui seront chargées au lancement du jeu, permettant d'avoir un code beaucoup plus optimisé qu'en UnrealScript. Pour certains comportements comme les cinématiques, il peut être également intéressant d'utiliser Kismet, qui permet de créer des scripts à travers une interface graphique.

Assets & extensions

En ce qui concerne l'import de modèle ou de texture, l'UDK accepte les principaux formats malgré une certaine limitation pour l'audio et la vidéo :

- Les formats de Maya, Blender, FilmBox, 3D Studio Max pour les modèles
- bmp, pcx, tga, float, psd et png pour les images
- wav 16 bit pour l'audio
- pour la vidéo, il est nécessaire de convertir le fichier au format Bink

Epic Games ne propose pas d'asset store officiel, même si des sites tel que http://udkresources.com proposent des packages prêts à être utilisés.

Communauté

Même si le site officiel propose une documentation très complète, le forum n'est aujourd'hui pas un exemple en termes de réactivité. La documentation officielle n'est pas traduite en français, mais de nombreux tutoriels sous forme de texte ou de vidéo existent sur le net. Malgré une communauté moins réactive, ils sont souvent d'une qualité appréciable pour les développeurs de tous niveaux.

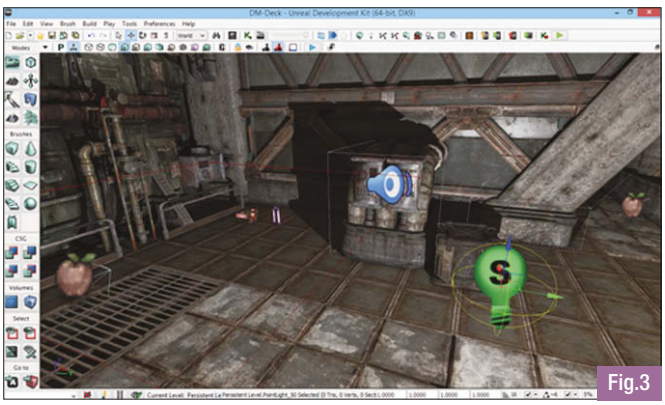


Fig.3

Aperçu de l'éditeur

Utilisation du moteur

Il est important de rappeler que l'UDK fut à la base créé pour le développement de jeux de tir tel qu'Unreal Tournament et que ses possibilités ont été étendues au fil du temps. Ainsi, si vous souhaitez vous lancer dans le développement d'un jeu de tir à la première ou troisième personne (FPS et TPS) ou d'un jeu 2D à défilement, l'UDK va être particulièrement adapté à ce genre d'utilisation.
En revanche, si vous développez un jeu basé sur une jouabilité plus complexe que dans le cadre défini précédemment, il est possible que le moteur soit moins adapté qu'un autre moteur. En cas de doute, la communauté UDK sur le forum officiel saura vous aiguiller sur ce genre de problématiques.

Liens

Site officiel pour télécharger l'UDK http://www.unrealengine.com/udk/
Documentation technique http://www.unrealengine.com/en/udk/documentation/
Forum Officiel http://forums.epicgames.com/forums/366-UDK

POUR CONCLURE

En comparaison

Il est relativement difficile de ne conseiller qu'un seul de ces trois moteurs, tant ils se positionnent différemment sur le marché. Le tableau comparatif suivant expose certaines forces et faiblesses de chacun afin de les mettre en perspective.

	Unity3D	UDK	CryEngine
Prix	Versions gratuite et Pro (1425 EUR)	Versions gratuite et pro (prix variable)	Versions gratuite et pro (prix variable)
Communauté	2 millions de développeurs, nombreux sites et blogs non-officiels	Communauté active	Petite communauté
Prise en main	Documentation claire, interface simple, très adaptable pour tous types de jeu	Interface simple. Particulièrement adaptée pour les jeux de tirs.	Interface agréable. Particulièrement adaptée pour les jeux de tirs. (nécessite des bases de C++)
Graphismes	Bonne qualité de base, plus poussée dans la version Pro	Très bonne qualité globale	Excellent en extérieur, bon en intérieur
Scripting	C#, Boo, JavaScript	UnrealScript	Lua
Plateformes	Ordinateurs, web, mobiles, consoles	Windows, iOS	Windows, consoles
Assets	Nombreux formats de données supportés, importation très simple, asset store officiel	Principaux formats supportés, pas de banque d'assets officielle.	Principaux formats supportés, pas de banque d'assets officielle
Configuration	Tourne sur la plupart des machines, même peu puissantes	Editeur gourmand en ressources	Editeur gourmand en ressources
Support	Mises à jour et corrections de bugs très fréquentes	Mises à jour peu fréquentes	Mises à jour fréquentes pour la version payante

On remarque que même si Unity3D semble rendre une copie quasi-parfaite grâce à sa maîtrise de la plupart des aspects comparés, UDK et CryEngine sont toujours des options de choix selon les besoins.

Et côté code ?

On peut également comparer de courts extraits de code pour chacun de ces 3 moteurs pour montrer, par exemple, comment effectuer un simple mouvement contrôlé via les touches fléchées du clavier.
Sur Unity3D, on peut redéfinir la fonction Update (héritée de MonoBehaviour) dans notre script qu'il suffira d'attacher à l'objet que l'on désire contrôler. Les entrées pour les déplacements verticaux et horizontaux sont définies et paramétrables dans l'interface du logiciel.

```
void Update()
{
 Vector3 direction = new Vector3(Input.GetAxis(«Horizontal»),
Input.GetAxis(«Vertical»), 0f);
 float movementSpeed = 10f;

 transform.Translate(direction * Time.deltaTime * movement
Speed);
}
```

Pour CryEngine3, on paramètre les actions du joueur dans DefaultProfile.xml dans le nœud **<actionmap name=»player»>** où sont placés les différents contrôles de base, par exemple pour les déplacements au clavier :

```
<action name=»moveleft» onPress=»1» onRelease=»1» retriggerable
=»1» keyboard=»a»/>
<action name=»moveright» onPress=»1» onRelease=»1» retriggerable
=»1» keyboard=»d»/>
<action name=»moveforward» onPress=»1» onRelease=»1» retriggerable
=»1» keyboard=»w»/>
<action name=»moveback» onPress=»1» onRelease=»1» retriggerable
=»1» keyboard=»s»/>
```

Pour récupérer les mouvements du joueur dans le code, on utilisera du C++ en faisant appel aux classes du moteur. CPlayerInput est responsable du traitement des actions de jeu, via son constructeur nous pourrons récupérer les différents paramètres, par exemple pour récupérer les appuis sur la touche W :

```
bool CPlayerInput::OnActionMoveForward(EntityId entityId, const
ActionId& actionId, int activationMode, float value)
```

Ici OnAction nous permet de récupérer les actions suivantes : appui sur une touche, relâchement de la touche, déplacement de la souris ou joystick. Cette méthode comprend 3 arguments : l'action est-elle activée, le mode d'activation (appui/relâchement/maintien) et la valeur de la position de la souris. Enfin, l'UDK demande de définir les touches utilisées dans un fichier DefaultInput.ini puis de déclarer dans une classe la fonction qui sera appelée. Voici la déclaration dans le fichier .ini qui appellera la fonction DKeyPressed lorsque la touche D sera utilisée :

```
Bindings=(Name=»D»,Command=»DKeyPressed»)
```

Ensuite, il faut créer la fonction DKeyPressed. Cette fonction déclare un vecteur de déplacement qui modifiera la position sur l'axe X de l'objet Block que vous souhaitez déplacer.

```
exec function DKeyPressed (int ParameterA)
{
 var vector deplacement;
 deplacement.X = 30;
 deplacement.Y = 0;
 deplacement.Z = 0;
 Block.move(deplacement);
}
```

La syntaxe ressemble beaucoup au C++ et au Java. Les programmeurs ayant déjà utilisé un de ces langages ne devraient pas avoir trop de problèmes à s'y habituer.

Les autres solutions

Mais le choix ne s'arrête pas là ! Même avec toute la volonté du monde, il serait impossible de lister et comparer la totalité des solutions existantes. Si la question du moteur parfait pour votre projet se pose, vous vous intéresserez certainement aux autres options telles que Source (le moteur 3D de Valve Corporation, utilisé entre-autres pour Half-Life 2 et Team Fortress 2), Gamebryo, LÖVE, Torque Game Engine, Cocos2D, Libgdx et bien d'autres. Enfin, la dernière solution reste de créer son propre moteur, mais le temps de développement nécessaire est rarement justifié par rapport aux avantages des moteurs existants.

Le meilleur moteur ?

Tout dépend ainsi de vos besoins et ambitions. Afin de faire votre choix, vous devrez avant tout définir le genre de jeu que vous comptez développer, les bases de *gameplay* nécessaires et votre capacité à créer les fonctionnalités manquantes. Le choix pour un jeu de plateforme en deux dimensions pour consoles ne sera pas le même que pour un jeu de combat en 3D pour PC et OS X.
A ce jour, Il n'y a pas encore de réponse parfaite à la question « quel est le meilleur moteur ? ». Chaque solution dispose d'avantages particuliers qui ont fait sa réputation : si Unity3D est un très bon choix pour un développeur indépendant désirant créer un jeu d'aventure, UDK et CryEngine sont de très bonnes alternatives pour des jeux de tir par exemple. Ces derniers seront en revanche moins attrayants pour le développement d'autres types de jeux, alors qu'ils pourront vous faire gagner du temps en début de développement comparé à Unity3D si vous comptez créer un jeu d'action.

Vincent **Hugues**, Cédric **Burceaux**, Vincent **Munoz**
InfiniteSquare

INFINITE SQUARE

Unity, UDK et Anarchy : comparatif !

Rentrons encore un peu plus dans les moteurs Unity, UDK et un « petit nouveau », Anarchy. Dans les pages qui suivent, nous allons comparer ces trois moteurs. La rédaction.

Rappels sur les trois moteurs

Unity3D est un moteur de jeux très facile d'utilisation, développé par Unity Technologies. Il existe une version gratuite et une version payante qui coûte 1500$, ou 400$ pour publier des jeux sur iPhone ou Android. La version gratuite contient nombre d'outils tels qu'un système de physique simple d'utilisation, un système de particules...

Le Project Anarchy est quant à lui, un moteur de jeux utilisant le moteur physique et réputé d'Havok : Havok Vision Engine. Ses outils sont multiples et proviennent de chez Havok, ce qui donne par exemple la possibilité de gérer des IA et des animations. Il permet avant tout la création de jeux sur mobile en version gratuite. Le développement de jeux PC, Mac OS ou consoles n'est possible qu'en version payante. Il existe d'ailleurs deux versions payantes. L'une accessible aux particuliers ou développeurs indépendants, et l'autre, aux entreprises ou équipes de développeurs importantes. La première coûte 499$, et la deuxième n'affiche pas ses prix ; selon le site officiel, ses tarifs sont calculés en fonction du business-model de ceux qui veulent l'acheter.

Unreal Development Kit (UDK) est un moteur de jeu basé sur Unreal Engine 3 qui intègre plusieurs outils développés par Epic Game sous forme d'éditeurs (Unreal Kismet, Unreal PhAT etc...). UDK est entièrement gratuit pour du développement personnel mais si vous décidez de commercialiser un jeu de votre création avec UDK, il vous faudra débourser 99$ pour obtenir une licence et vous devrez payer 25% de royalties à partir de vos premiers 50.000$ gagnés. Le développement pour PC et iOS est gratuit, en revanche pour le développement sur PS3 et XBOX360, il vous faudra la licence. Il est aussi possible de développer pour le Web, notamment avec Adobe Flash.

Facilité d'utilisation / Première approche

Unity3D possède une interface très intuitive. On peut très vite faire ce que l'on souhaite : c'est clair, on retrouve rapidement ses objets, ses scripts. Quelqu'un qui n'a jamais programmé de sa vie arrivera à faire des scènes simples. On peut drag and dropper ses scripts, ses assets ou autres composents pour les associer avec des objets, ce qui est particulièrement pratique. Des packages de base permettent, en un clic, de créer un personnage contrôlable par exemple. La communauté d'Unity3D est très importante : on peut trouver des réponses facilement sur les forums du site d'Unity3D, sans parler des centaines de tutoriels sur le Net. Sachez qu'un système de versionning est possible avec mercurial ou svn.

L'interface d'Anarchy est un peu moins accueillante que celle d'Unity3D mais n'en reste pas moins logique. On peut créer des entités auxquelles on peut rattacher des components un peu comme sous Unity3D, mais en naviguant à travers une page "properties" au lieu d'un drag and drop. Comme il s'agit d'un moteur de jeux récent, il n'y a pas encore une communauté très active sur le Net et les tutoriels sont encore rares. Vous en trouverez de très intéressants pour débuter sur le site officiel du Project Anarchy qui a créé toute une collection de vidéos pour apprendre à utiliser ce moteur.

Quant à UDK, son interface est complète, il faut en revanche beaucoup naviguer pour trouver ce que l'on vient y chercher. Beaucoup d'icônes sont à disposition et elles sont intuitives. La communauté autour d'UDK est importante et active. UDK intègre une feature importante pour les Level Designers, le BSP Brushes (Binary Spacing Partitioning) servant à associer des formes géométriques directement à la map. On peut créer des formes simples facilement, et les modifier plus précisément que sur Unity3D. De très belles maps d'exemples sont disponibles avec le moteur, pour se faire la main dessus.

Graphique / Asset Management

Unity3D possède un asset store permettant d'obtenir des modèles gratuits et d'autres payants. Si vous voulez aller vite, il existe aussi des formes géométriques simples intégrées de base (cube, plan, sphère, cylindre). On peut importer des modèles 3D sous la plupart des formats classiques (FBX, 3DS, Maya) en glissant les modèles directement dans la fenêtre du projet. On peut aussi bien sûr importer des musiques au format mp3, wav, ogg ou encore les streamers au runtime. Beaucoup de plugins sont disponibles sur l'asset store.

Anarchy est plus difficile à utiliser pour importer des assets mais n'est pas hors de portée. Seulement quelques formats sont supportés pour le moment : 3DSMax, Maya, et FBX depuis peu. Pour importer des modèles 3D au format 3DS vous devrez installer 3DSMax ainsi qu'un plugin permettant d'importer les modèles depuis le logiciel. Un software pour importer ses FBX est disponible mais il est encore en version alpha. Le fait de devoir avoir une version de 3DSMax installée afin d'importer des assets fonctionne, mais le principe est plutôt étrange. Project Anarchy vise en effet un public de développeurs indépendants, or les suites Autodesk ne sont pas à la portée de tous.

Unreal Development Kit intègre de nombreux shaders de base pouvant rendre un jeu très beau, l'interface permet aussi de créer des shaders et des materials rapidement et facilement sans avoir à mettre une seule ligne de code.

Il est possible d'importer les modèles en FBX mais aussi d'importer des modèles faits avec 3DSMax, Maya, Zbrush et Photoshop. Les outils intégrés tels que Unreal Matinee (éditeur de cinématiques), SpeedTree (création d'arbres, feuilles, herbes...) et FaceFX (animation des visages) apportent une aide aux graphistes mais aussi aux non développeurs.

Scripting / Physique

Sous Unity3D il est possible de scripter en C#, Javascript, et Boo. Le moteur est accompagné de l'IDE Monodevelop. Il est possible de coder avec Visual Studio, mais cela est laborieux et laisse le mode debug de Visual en arrière-plan. La documentation est très bien faite et contient des exemples de code dans les trois langages.

Créer un jeu en réseau sera d'une facilité enfantine pour ceux qui ont des bases en réseau. Unity3D comporte des classes contenant toutes les fonctions nécessaires afin de créer un serveur en quelques lignes. Il faudra donc forcément apprendre à scripter pour se lancer dans un projet en réseau.

La physique quant à elle est facilement calibrable, mais son adaptabilité à différentes plateformes la rend moins performante que certains moteurs spécialisés dans un seul type de plateforme. Des colliders, et rigidbodies sont drag and droppable sur les objets d'une scène.

Anarchy permet de scripter en Lua et C++. L'IDE est directement intégré dans l'éditeur Vision Engine, mais vous pouvez utiliser Visual Studio. Le moteur est d'ailleurs open-source et codé en C++, alors si quelque chose ne vous plait pas, vous pouvez le changer ! La documentation est très complète et explicative. Il manque peut-être un glossaire des fonctions utilisables, mais l'auto-complétion dans l'IDE facilite la tâche du dévelop-

peur. Anarchy permettant uniquement de créer des jeux mobiles dans sa version gratuite, il ne semble pas y avoir d'élément permettant de créer un jeu en réseau au sein du moteur.

On peut dire que la physique d'Havok mérite sa réputation. Configurable à souhait, elle se dote d'un mode debug très détaillé montrant les vecteurs symbolisant les forces et vitesses appliquées aux objets de la scène en temps réel.

UDK utilise un langage propre : l'UnrealScript qui est similaire au Java. Rebutant au premier abord (si on a touché aux scripts, on doit relancer le moteur), il n'en reste pas moins efficace. Aucun IDE n'est fourni avec UDK, il vous faudra vous rabattre sur votre éditeur de texte préféré. Une gestion d'état de l'objet permet de surcharger des fonctions selon l'état. Il est d'ailleurs plus facile de modifier une classe préexistante que d'en recréer une de toute pièce.

Pour l'utilisation du réseau, il faut manipuler un objet qui est composé de plusieurs classes se surchargeant les unes les autres. Chaque classe possède donc des attributs qui la définissent.

Il est possible de "répliquer" la variable en l'insérant dans une liste afin que le serveur envoie automatiquement la nouvelle valeur au client lorsque cette variable change. Une fonction exécutée sur le serveur peut appeler une fonction sur le client, et réciproquement. C'est ainsi que les joueurs mettent à jour leur position sur le serveur par exemple : les joueurs exécutent une fonction qui met à jour la position en local, puis appelle une fonction sur le serveur (avec les paramètres) afin de se mettre à jour sur le serveur. UDK est un moteur de jeu très puissant qui permet de charger beaucoup d'objets dans une scène (plus de 5000 gérés en temps réel). Sa physique est aussi très efficace et propose de nombreux outils, notamment l'Unreal PhAT (Physics Asset Tool), outil qui gère les collisions et les ragdolls. Il existe aussi un système permettant de gérer la physique des objets rigides, élastiques, des tissus, ou encore de simuler des foules contenant des centaines de personnages.

Voici un exemple de code rédigé pour chaque moteur de jeu, permettant de faire tourner un objet sur lui-même et d'afficher ses informations. Les exemples diffèrent sur quelques points en fonction des particularités des moteurs de jeux. Nous ne montrerons pas d'exemple de GUI pour Anarchy, pour ne citer que lui, car il faut manipuler des fichiers XML afin de détailler l'arborescence des menus. Notre but est de vous montrer un rapide aperçu du code que vous pouvez lancer au sein de projets sous les différents moteurs, sans manipulation particulière dans l'interface.

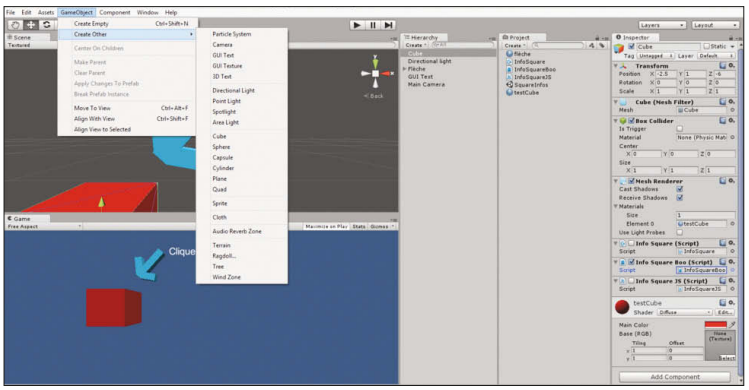
Unity3D

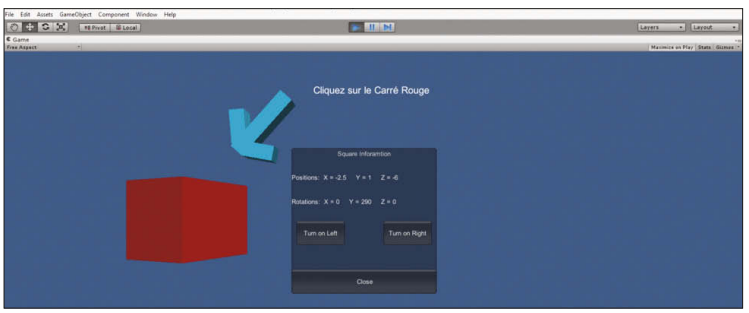
C#

```
using UnityEngine;
using System.Collections;

public class InfoSquare : MonoBehaviour {
  private bool infoObject;

  void Start ()
  {
    infoObject = false;
  }

  // Quand le clic gauche de la souris est fait sur l'objet
  void OnMouseDown()
  {
    infoObject = true;
  }

  // Affichage d'une GUI
  void OnGUI()
  {
    // On vérifie que l'on veut afficher les informations
    if (infoObject)
    {
      // Création d'une "fenêtre" d'information
      GUI.Box(new Rect(600, 200, 300, 300), "Square Inforamtion");

      // Listing des informations affichées
      GUI.Label(new Rect(600, 250, 300, 50), "Positions:  X =
" + transform.position.x.ToString() + "   Y = " + transform.
position.y.ToString() + "   Z = " + transform.position.z.To
String());
      GUI.Label(new Rect(600, 300, 300, 50), "Rotations:  X =
" + transform.rotation.eulerAngles.x.ToString() + "   Y = "
+ transform.rotation.eulerAngles.y.ToString() + "   Z = " +
transform.rotation.eulerAngles.z.ToString());


      if (GUI.Button(new Rect(610, 350, 100, 50), "Turn on Left"))
        transform.Rotate(new Vector3(0, 1, 0), 10);
      if (GUI.Button(new Rect(790, 350, 100, 50), "Turn on Right"))
        transform.Rotate(new Vector3(0, -1, 0), 10);
      if (GUI.Button(new Rect(600, 450, 300, 50), "Close"))
        infoObject = false;
    }
  }
}
```

Boo

```
import UnityEngine
import System.Collections

class InfoSquareBoo (MonoBehaviour):

  private infoObject as bool
```





UNITY

```
 def Start ():
   infoObject = false

 def Update ():
   pass

 // Quand le clic gauche de la souris est fait sur l'objet
 def OnMouseDown () as void :
   infoObject = true

// Affichage d'une GUI
 def OnGUI () as void :
// On vérifie que l'on veut afficher les informations
   if infoObject == true:
     // Création d'une "fenêtre" d'information
     GUI.Box(Rect(600, 200, 300, 300), "Square Inforamtion")
     GUI.Label(Rect(600, 250, 300, 50), "Positions:  X = "
+ transform.position.x.ToString() + "   Y = " + transform.
position.y.ToString() + "      Z = " + transform.position.
z.ToString())
     GUI.Label(Rect(600, 300, 300, 50), "Rotations:  X = "
+ transform.rotation.eulerAngles.x.ToString() + "   Y = " +
transform.rotation.eulerAngles.y.ToString() + "    Z = " +
transform.rotation.eulerAngles.z.ToString())
      // Création et action des boutons
     if GUI.Button(Rect(610, 350, 100, 50), "Turn on Left"):
       transform.Rotate(0,10,0,0)
     if GUI.Button(Rect(790, 350, 100, 50), "Turn on Right"):
         transform.Rotate(0,-10,0,0)
     if GUI.Button(Rect(600, 450, 300, 50), "Close"):
         infoObject = false
```

JavaScript

```
#pragma strict

private var infoObject : System.Boolean;

function Start () {
 infoObject = false;
}

  // Quand le clic gauche de la souris est fait sur l'objet
function OnMouseDown () : void
{
 infoObject = true;
}

 // Affichage d'une GUI
function OnGUI() : void {
 // On vérifie que l'on veut afficher les informations
 if (infoObject)
 {
     // Création d'une "fenêtre" d'information
     GUI.Box(new Rect(600, 200, 300, 300), "Square Information");

     // Listing des informations affichées
     GUI.Label(new Rect(600, 250, 300, 50), "Positions:  X =
" + transform.position.x.ToString() + "   Y = " + transform.
position.y.ToString() + "    Z = " + transform.position.z.
ToString());
     GUI.Label(new Rect(600, 300, 300, 50), "Rotations:  X =
" + transform.rotation.eulerAngles.x.ToString() + "    Y =
" + transform.rotation.eulerAngles.y.ToString() + "    Z =
" + transform.rotation.eulerAngles.z.ToString());

     // Création et action des boutons
     if (GUI.Button(new Rect(610, 350, 100, 50), "Turn on Left"))
       transform.Rotate(new Vector3(0, 1, 0), 10);
     if (GUI.Button(new Rect(790, 350, 100, 50), "Turn on Right"))
       transform.Rotate(new Vector3(0, -1, 0), 10);
     if (GUI.Button(new Rect(600, 450, 300, 50), "Close"))
       infoObject = false;
 }
}
```

Project Anarchy

Lua

```
function OnAfterSceneLoaded(self)

 --Crée une map regroupant toutes les touches que vous voulez
utiliser ainsi que leurs fonctions respectives

 self.map = Input:CreateMap("myInputMap")

 --Touches Fléchées
 self.map:MapTriggerAxis("MoveX", "KEYBOARD", "CT_KB_LEFT",
"CT_KB_RIGHT")

 --Touches ZQSD (gauche et droite)
 self.map:MapTriggerAxis("MoveX", "KEYBOARD", "CT_KB_Q", "CT_KB_D")

 --Joystick pour une manette, si elle est branchée
 --self.map:MapTriggerAxis("MoveX",  "PAD1",  "CT_PAD_RIGHT
_THUMB_STICK_LEFT", "CT_PAD_RIGHT_THUMB_STICK_RIGHT")

--Touche espace pour afficher des infos dans la console
 self.map:MapTrigger("ShowInfo", "KEYBOARD", "CT_KB_SPACE")

end

function OnThink(self)

--Fait tourner l'objet sur lui-même quand on appuie sur les
touches correspondantes.
  if self.map:GetTrigger("MoveX") < 0 then
 self:IncOrientation(5, 0, 0)
  elseif self.map:GetTrigger("MoveX") > 0 then
 self:IncOrientation(-5, 0, 0)
  elseif self.map:GetTrigger("ShowInfo") == 1 then
 Debug:Log(self:GetOrientation().x)
  end

end
```

UDK

UnrealScript

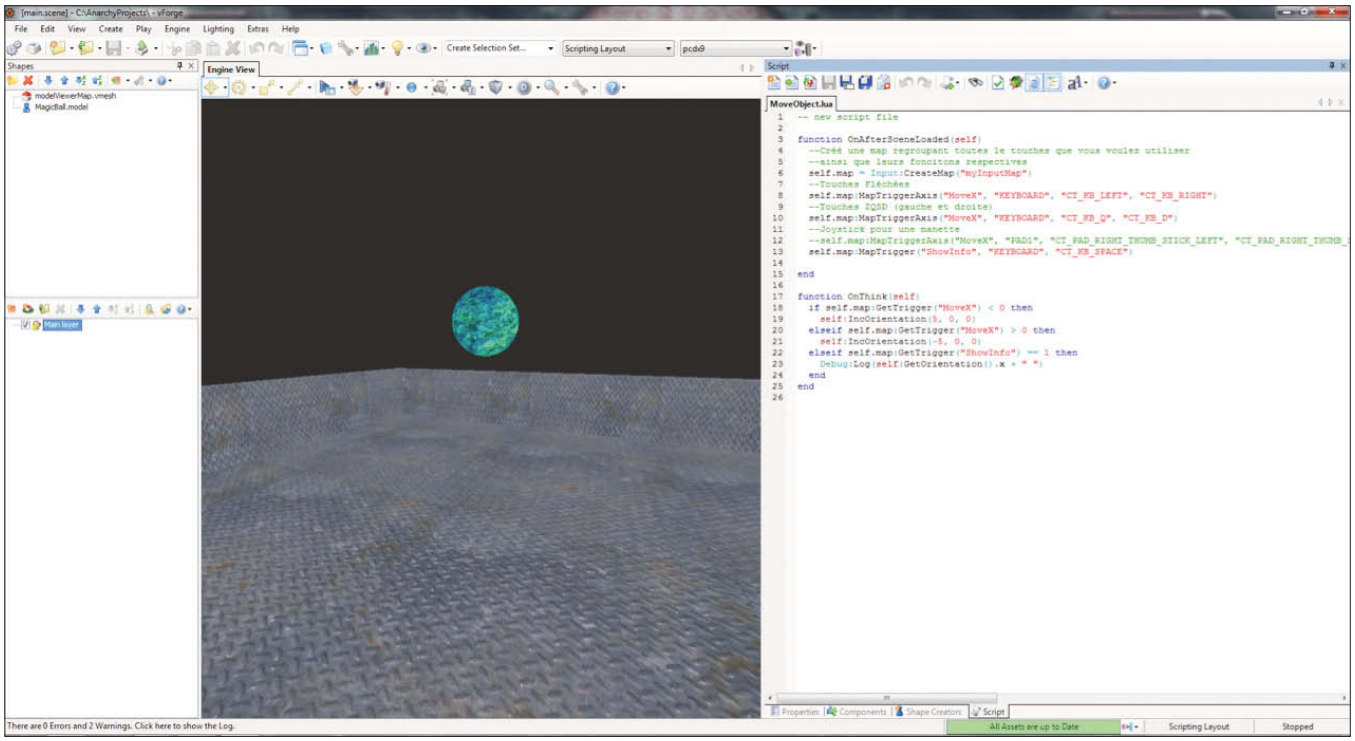
```
// Effectue une rotation à chaque frame
event Tick(float DeltaTime)
{
local float deltaRotation;
```

```
local Rotator newRotation;

deltaRotation = velRotation * DeltaTime;

newRotation = Rotation;

newRotation.Pitch += deltaRotation;
newRotation.Yaw += deltaRotation;
newRotation.Roll += deltaRotation;
SetRotation(newRotation);
}
// Affiche les informations de l'objet
function showTargetInfo()
{
 local vector loc, norm, end;
 local TraceHitInfo hitInfo;
 local Actor traceHit;

 end = Location + normal(vector(Rotation))*32768; // Raycast
vers "l'infini"
 traceHit = trace(loc, norm, end, Location, true,, hitInfo);

 ClientMessage("");

 if (traceHit == none)
 {
   ClientMessage("Nothing found, try again.");
   return;
 }

 // Joue un son pour confirmer qu'un objet a été touché
 ClientPlaySound(SoundCue'A_Vehicle_Cicada.SoundCues.A_Vehicle_
Cicada_TargetLock');

 // Affiche des messages d'information sur l'objet dans la chatbox.
 ClientMessage("Hit: "$traceHit$"  class: "$traceHit.class.
outer.name$"."$traceHit.class);
 ClientMessage("Location: "$loc.X$","$loc.Y$","$loc.Z);
 ClientMessage("Material: "$hitInfo.Material$"  PhysMaterial:
 "$hitInfo.PhysMaterial);
 ClientMessage("Component: "$hitInfo.HitComponent);
}
```

Conclusion

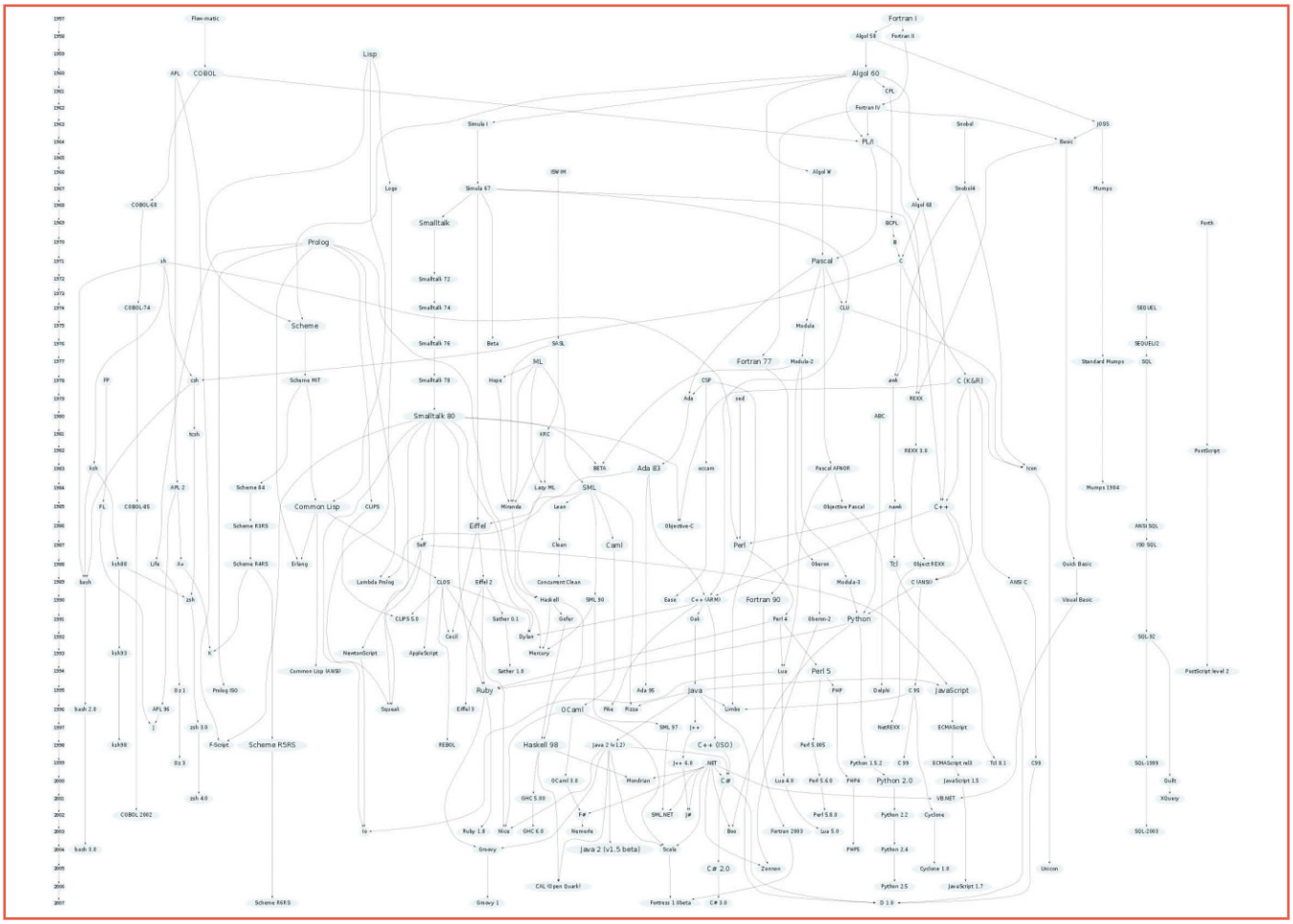
Unity3D est un moteur de jeux performant qui permet de développer sur de nombreux supports. Cela le rend moins puissant que ses concurrents qui se concentrent sur certaines plateformes en particulier. Il a comme atout d'être très facile d'utilisation et de bénéficier d'une importante communauté, ce qui le rend encore plus accessible. Cela est dû au fait que le moteur est disponible depuis 2007. Il a bénéficié de nombreuses améliorations ces derniers mois, notamment la sortie d'Unity2D ou encore la sortie en beta des nouveaux assets de base d'Unity3D : les Sample Assets.
Ceux qui veulent développer pour mobile pourront se tourner vers Project Anarchy. Le moteur est encore jeune, il est donc encore difficile de trouver des tutoriels en ligne comme pour Unity3D. Néanmoins la documentation est de bonne qualité et la communauté autour du moteur grandit jour après jour. Il possède certains outils qu'Unity3D n'a pas et inversement. C'est souvent ce qui fera pencher la balance vers un moteur ou l'autre en fonction de vos projets. Sachez que Project Anarchy 2D est en développement. Pour les plateformes fixes comme PC ou Mac, les développeurs pourront choisir UDK. Le développement mobile est tout de même possible mais n'est pas la spécialité du moteur. Cela tend cependant à s'améliorer au fil des versions, notamment sur iOS. UDK est très puissant et possède de nombreux outils. Moins facile d'accès qu'Unity3D, il reste accessible aux non développeurs. Ce moteur a déjà fait ses preuves avec de nombreux jeux à succès, on peut citer par exemple la série des Bioshock ou encore Batman Arkham City.

Florence **Noé**, Jean-Sébastien **Perrier** et Ulysse **Manceron**
Pandas du GameDev Lab d'Epitech



ANARCHY

Futurs et tendances
des langages de programmation



© Pixel

Dans le n°50 de Programmez !, nous avions représenté sous forme de schéma quelques familles de langages. Et encore, il en manquait beaucoup (langages dynamiques, langages fonctionnels, etc.). 123 numéros plus tard, soit 12 ans, notre schéma a-t-il beaucoup bougé ? Assez peu : des nouvelles versions, quelques dérivés de langages. D'un côté, il est normal que le langage de programmation évolue : nouvelles fonctions, nouvelles commandes. Le langage se complète peu à peu, au risque de l'alourdir. L'inflation des API, des SDK n'en est qu'un des symptômes. Prenons le langage Objective-C popularisé avec iOS, langage historique de NeXT et maintenant d'OS X et de iOS; il est à l'origine un dérivé du C, avec une forte influence de Smalltalk. Le langage DART de Google prend racine sur JavaScript, Smalltalk, Erlang et même C# ! Même le langage R, très utilisé dans les statistiques, hérite de plusieurs langages : S et Scheme. Smalltalk (lui-même influencé et inspiré par Lisp, Logo, Simula) est à la base de + 20 langages ! Bref, la consanguinité est très forte. Si on me demandait les langages de 2014, aucune surprise à prévoir : C++, C+, Objective-C, Javascript, Java, Python, C#. Nous sommes loin d'une rupture.

Mais est-elle possible ? Si nous observons bien, il y a clairement une nécessité à rompre : le massivement distribué (la capacité à monter en charge automatiquement) tel que l'on peut l'avoir avec Scala et Go, la multiplication des terminaux connectés pose la question d'un langage déterminé, fiable et robuste, l'explosion des données non-structurées (typiquement le big data)... Imaginons que l'informatique quantique existe réellement au quotidien, avec quoi pourrions-nous développer et quels paradigmes adopter ? Une pertinente analyse a été faite lors du colloque « the future of programning » sur comment le langage de programmation structure et influence la pensée informatique (attention : cela peut donner mal à la tête) :

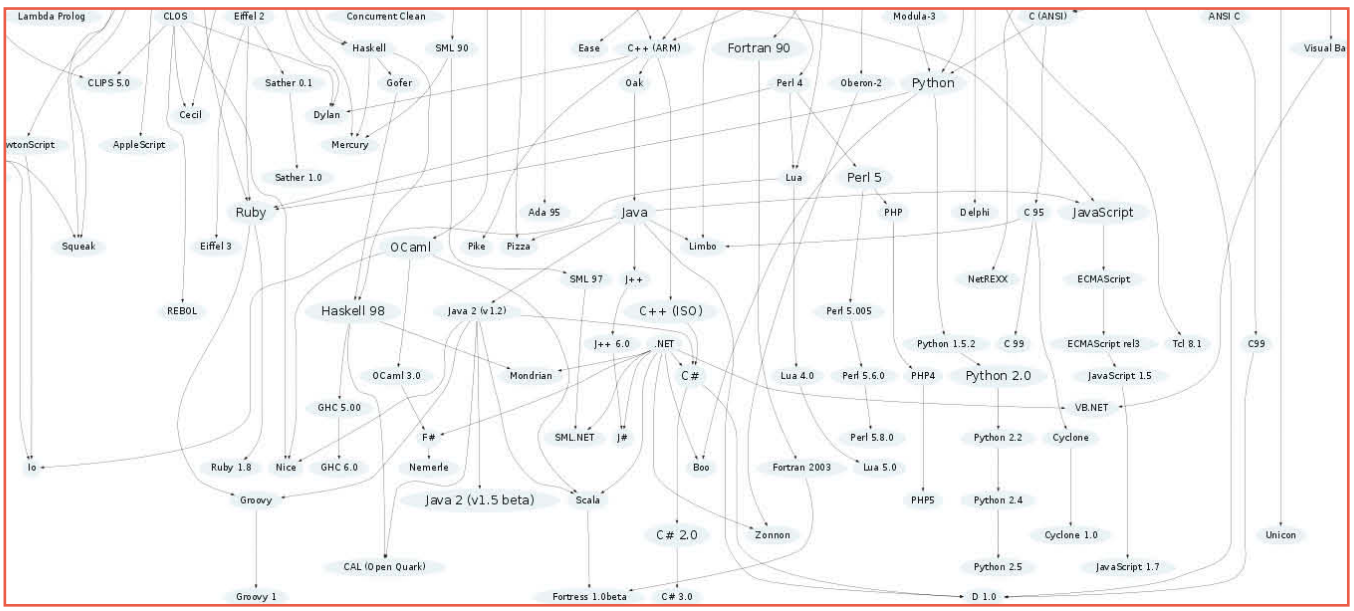
- les systèmes logiciels sont pour les personnes,
- les logiciels sont là « juste pour faire leur boulot »,
- les logiciels sont l'encodage de la pensée informatique,
- les ingénieurs logiciels sont les architectes et les mécaniciens de la société moderne de l'information,
- l'ingénierie informatique fournit les outils que les ingénieurs logiciels utilisent pour faire un bon travail,
- les langages de programmation sont les outils clés pour encoder les process,
- les langages de programmation sont aussi des systèmes logiciels,
- les langages de programmation devraient « juste travailler »,
- les designers de langage créent les langages de programmation,
- l'ingénierie des langages fournit les outils pour que les designers de langage fassent un bon travail,
- les meta-langages supportent l'encodage et le design des langages,
- les ingénieurs langages mangent leur propre nourriture (dogfood en Anglais)

Le schéma peut paraître abstrait, voire, totalement idiot, mais l'est-il réellement ? Pour aller au-delà, nous vous conseillons le colloque « the future of programming » de janvier 2014 : http://eelcovisser.org/wiki/future-of-programming
Et aussi : http://fr.slideshare.net/eelcovisser/programming-languagesshapecomputationalthinking
Une autre conférence à consulter sans modération : DBX Conference http://worrydream.com/dbx/
Et aussi : http://www.tele-task.de/archive/series/overview/844/#lecture5819

François **Tonic**

Débat : le développeur doit-il évoluer ? Faut-il jeter l'orienté objet ? Faut-il changer notre mode de penser l'informatique ?

Durant la journée développeur organisée à l'école ESGI, le 16 janvier 2014, un des sujets de discussion fut le langage orienté objet, s'il fallait continuer à l'enseigner et s'il était possible de savoir si ce paradigme était toujours pertinent; d'autres langages, comme Python, apportent des réponses claires à des problématiques métiers ou applicatives. Et l'objet n'est pas la réponse à tout.



© Pixel

Avez-vous un problème avec l'objet ? Pourquoi se focaliser dessus ? Ne faudrait-il pas élargir notre vision et nos compétences ? Le débat a été vif et passionnant à suivre. Nous avons décidé de poursuivre le débat, parallèlement au dossier « futur des langages ». La rédaction.

Les débatteurs sont :

- Thierry Joubert,
  directeur technique/cofondateur de Theoris, enseignant ESGI
- Frédéric Jadel,
  président fondateur d'Aspectize, enseignant ESGI
- Michael Thomas,
  président fondateur de Wehicles, enseignant ESGI

Thierry Joubert : J'ai trouvé récemment dans mes lectures la citation suivante : *« Pour l'exercice d'un métier, la seule vision d'outils est peu parlante, seul leur usage systématique permet à qui les manie d'en faire ce qu'il veut. Autrement dit pour comprendre les mathématiques il faut commencer par faire des mathématiques, de manière à instaurer les automatismes mentaux à l'instar de cet automatisme physique qu'est la marche ».*

À mon sens cette citation vaut également pour la programmation des ordinateurs. On peut m'opposer le fait que les langages de programmation ne sont que des cousins éloignés des formalismes mathématiques, et que ces derniers sont certainement plus utiles à l'effort de conceptualisation qu'à celui de programmation. Il n'en est pas moins que la conceptualisation est une étape préliminaire à la programmation. Et c'est parfois aussi en programmant un système que l'on précise sa conceptualisation. En bref, il apparaît que ces deux activités partagent non seulement un côté franchement cérébral, mais travaillent également main dans la main (si je puis oser une métaphore).

Le langage informatique, en tant qu'outil du programmeur, permet de rapprocher ces deux dimensions, en apportant une « puissance de feu » comme les notions de procédure, de paramètre ou d'exception, et, plus encore, des notions de classe et d'instance. Tout cela se construit aux dépens de la performance, et il existe de nombreux niveaux dans l'approche proposée par les langages informatiques qui se répartissent en bon ordre. Du moins au plus « conceptuel », je donnerais la liste ordonnée qui suit : langage machine et assembleur, langage procédural et enfin langage-objet.

Frédéric Fadel : Si je suis entièrement d'accord avec Thierry concernant le caractère intellectuel du métier d'informaticien au même titre que celui de mathématicien, le parallèle entre le formalisme mathématique et les langages de programmation n'est pas, à mon sens, très heureux.

Pour le mathématicien, le formalisme n'est qu'un moyen (relativement insignifiant) lui permettant d'échanger des idées et des propositions avec ses collègues, avec « précision », et dans une grande mesure, sans ambiguïté. Le mathématicien n'est pas épris de son formalisme.

Alors que pour l'informaticien, la situation est très différente, et cela, sur plusieurs plans. D'abord le langage informatique ne joue pas le même rôle : l'utilité des langages ou catégories de langages cités par Thierry n'est pas la communication entre informaticiens, mais la

communication entre un informaticien et une machine, entre une intelligence humaine et un « automate », qui va exécuter des milliards d'instructions par seconde avec une tolérance zéro concernant l'ordre des instructions et leur parallélisme. Cette situation exige de l'informaticien une précision inhumaine.
Ensuite, et c'est un constat culturel, contrairement aux mathématiciens, beaucoup d'informaticiens sont fascinés par leurs outils... Ils se divisent en clans, préférant tel langage ou telle marque de système d'exploitation, et ce faisant, ils oublient l'essentiel, c.-à-d. le métier qu'ils sont censés informatiser.

Thierry Joubert : Il faut bien distinguer conception et programmation, pour un problème donné dans un domaine métier, il existe virtuellement une infinité de solutions informatiques. Si beaucoup de ces solutions ne donnent que partiellement satisfaction, c'est certainement qu'il y a eu un manque dans la chaîne, et le maillon faible est souvent la conception. Dans les faits il est paradoxalement plus difficile de communiquer avec quelqu'un qu'avec un ordinateur. La difficulté de communication empire lorsque les interlocuteurs n'ont pas la même culture. A ma connaissance les mathématiciens communiquent principalement avec leurs collègues.
Du fait de la nature informelle du programme informatique, la seule façon pour un informaticien de communiquer avec le spécialiste métier, c'est de lui fournir des programmes « à essayer », un peu comme dans un magasin de vêtements. Pour mener cet exercice de manière optimale, l'informaticien doit maîtriser la programmation des automates depuis le niveau le plus bas. Dans ce domaine, la liste présentée est très incomplète, mais elle a le mérite de représenter un « tronc de base » permettant de bâtir tout le reste. En effet beaucoup des programmes écrits avec ces outils de base ne sont que des sortes d'interpréteurs d'autres langages métier de plus haut niveau, dont la syntaxe et/ou la sémantique appartiennent à la culture de l'utilisateur. Je citerais par exemple les langages du monde des automatismes programmables industriels (API pour faire court), qui n'entrent dans aucune des catégories de langages déjà citées, et qui intègrent d'emblée une notion comme le parallélisme dont on ne peut pas dire qu'elle soit naturelle en procédural ou en objet. Une question se pose à ce point de l'exposé : est-ce qu'à un moment de l'histoire on pourra se débarrasser du « tronc de base » pour ne plus utiliser que des outils ou langages métiers ? Autrement dit, est-ce que l'effort de programmation peut converger vers un avenir purement métier, et ce de manière irréversible ?

Frédéric Fadel : La question, pour moi, n'est pas tellement s'il est possible de se débarrasser du « tronc de base » des langages informatiques, ni depuis quand, parce que oui c'est possible et depuis fort longtemps. La question qui me préoccupe davantage est : est-on capable de changer le métier de l'informaticien tel qu'il se pratique aujourd'hui avec beaucoup d'inefficacité, de souffrances, de déchets, de code et d'échecs, en une informatique plus industrialisée, qui s'intéresse plus à l'immensité des choses possibles, qu'aux instructions et aux langages qui les produisent ?
Pour moi le rôle de l'informaticien, c'est d'organiser et d'orchestrer les échanges d'information entre les machines et les hommes (le rôle des machines étant très limité), et pour cela, pas besoin des langages du « tronc de base ». Avec un exemple ce sera plus clair. Un des bouts de code les plus utilisés sur la planète et ce depuis plus de 30 ans, c'est le code qui permet aux machines de communiquer entre elles dans un réseau mondial. Pour cela, les inventeurs d'Internet ont écrit relativement peu de code, mais ils ont imaginé le système dans son ensemble de telle manière que plusieurs décennies après, le réseau est debout, a évolué sans remettre en cause son passé, et s'est adapté à nombre de technologies nouvelles qui n'existaient pas au moment de sa création. C'est l'imagination d'un système adaptable qui est le cœur du travail de l'informaticien et ce n'est pas possible s'il s'imagine écrire beaucoup de code.
Autre exemple, ceux qui ont inventé le Web : ils ont écrit peu de code pour ajouter au-dessus d'Internet et ses protocoles, d'autres protocoles qui permettent aujourd'hui à une personne de déclencher des échanges d'informations entre des dizaines de machines, juste en cliquant sur un lien, sans écrire le moindre code, ou plutôt en utilisant d'une manière sous-jacente toujours le même bout de code écrit il y a longtemps. Pour échanger des données entre les hommes et les machines, point besoin de code, mais plutôt de l'imagination pour inventer un système adaptable.
Pour revenir à la question de la convergence de l'informatique vers un avenir purement métier, je dirais que c'est inévitable, c'est déjà en marche : elle ne converge pas aussi vite qu'elle pourrait, non pas pour des raisons techniques, mais pour des raisons d'habitudes. Pour « *instaurer les automatismes mentaux* » de cette nouvelle informatique, il faut cesser de penser code, mais penser métier, adaptable et dynamique.

Thierry Joubert : Le code qui permet aux machines de communiquer entre elles ou le protocole Web ne sont pas des fins en soi, ce ne sont que des « super-automates » qu'il faut encore une fois programmer. Il suffit de penser à la quantité de programmes qui utilisent l'API socket, ou encore à tous ces sites Web qui contiennent du code JavaScript et PHP pour s'en convaincre.
Je suis d'accord que des outils métier comme Matlab ou Maple permettent à des ingénieurs de créer des programmes informatiques en n'utilisant que des formalismes mathématiques. Quand on arrive ainsi à faire entrer un domaine métier dans un formalisme bien structuré, ou dans des formes prédéfinies, on peut se débarrasser du langage de programmation. Mais toutes les interfaces graphiques peuvent-elles avoir le même aspect ? Tous les sites Web peuvent-ils être faits avec un unique CMS ?
Je crains que l'élimination des langages de programmation ne nous mène à la fin de la créativité. Cette question a un impact essentiel sur la manière d'enseigner l'informatique, et donc sur le futur même de l'informatique. En tant que développeur, je ne peux qu'apprécier cette superbe mise en abîme. Mais elle me fait frémir, car si nous ne faisons pas les bons choix aujourd'hui, l'avenir ne nous appartient pas.

> *« Je crains que l'élimination des langages de programmation ne nous mène à la fin de la créativité »*
> (Thierry Joubert)

Frédéric Fadel :
Effectivement Internet et le Web ne représentent pas des fins en soi, mais comme ils ont été conçus pour croître organiquement comme une plante et non pas codés en dur comme la tour Eiffel, ce sont pour moi des exemples de systèmes bien conçus : ils évoluent et s'adaptent comme les systèmes biologiques, ce ne sont pas des suites presque infinies d'instructions qui flanchent au premier changement.
Quand je dis que l'important c'est le métier et

pas les techniques d'un quelconque langage de programmation, je ne parle pas de ceux qui utilisent Matlab ou Maple. Je parle par exemple d'un système de paye ayant coûté des centaines de millions d'euros, et qui ne donne pas satisfaction. Je parle d'un système d'accès à des assurances de santé publique ayant coûté des centaines de millions de dollars, et qui a flanché au 500e utilisateur. Je parle de plus de 60 % des projets informatiques qui sont en échec. Pour moi la raison de ces échecs, c'est le code, peu importe le langage. Aujourd'hui le processus de fabrication du logiciel est basé sur l'écriture de code. Nos dogmes comme l'orienté objet mettent l'informaticien dans un mode de pensée ni proche ni adapté à un quelconque métier : l'orienté objet invite à écrire du code, beaucoup de code, du code qui appelle du code, souvent selon des schémas répétitifs. Cela, pour moi, s'apparente à la violence : « Si elle n'a pas résolu votre problème c'est que vous n'en avez pas exercé assez ».

L'art de créer des logiciels qui fonctionnent dès le premier jour, évoluent et s'adaptent au changement inéluctable des métiers, des besoins, des circonstances ne passe pas par le déversement de code, le test de code, la correction du code, cercle vicieux qui induit des retards, des coûts élevés, une satisfaction faible.

Pour un métier donné, parmi toutes les solutions informatiques possibles, celles avec le moins de code technique seront les plus solides et robustes. En effet, comme le code technique est souvent le même, qu'il est prévisible, et qu'on a 40 ans d'expérience dans son écriture, on peut et on doit l'écrire une fois pour toutes.

Là où il y a besoin d'intelligence, c'est dans la conception d'une structure des données appropriée au métier, l'imagination d'une expérience utilisateur agréable, et, si besoin, dans l'écriture, d'un peu de code métier pour des calculs ou des validations. Tout le reste, lecture, écriture, transport, affichage, sécurité des informations, peuvent être automatisés, ce qui permettra de diminuer le volume de code d'un facteur 50 pour des logiciels moyennement sophistiqués, davantage pour des logiciels plus complexes. On maîtrise la complexité en écrivant peu de code, pas l'inverse.

Il n'y a pas à craindre pour la créativité parce qu'on en a rudement besoin pour concevoir les données, les échanges et les interactions utilisateurs : pour tout cela le code n'est pas d'un grand secours.

Nous sommes d'accord, la difficulté réside dans la communication avec le spécialiste métier et la compréhension de son besoin. Une façon pragmatique et efficace de le faire, c'est en effet « de lui fournir des programmes "à essayer" un peu comme dans un magasin de vêtements », et ce dès le premier jour. Ni nécessaire ni utile, le code pousse l'informaticien à chercher la solution là où il voit de la lumière (écrire du code pour parler à la machine) et pas là où réside la difficulté du métier.

> *« L'orienté objet invite à écrire du code, beaucoup de code »*
> (Frédéric Fadel)

Thierry Joubert : Pour être précis, les 60 % de projets qui ne donnent pas satisfaction sont ceux de plus d'un million de lignes de code (et on atteint 80 % d'insatisfaction pour ceux de plus de 10 millions de lignes). À l'opposé, les projets de 10 000 lignes de code donnent satisfaction à environ 80 %. Je suis d'accord avec Frédéric sur le fait qu'il y a un immense problème au niveau des « gros » projets informatiques, et je souscris à son argument sur le fait que le code nécessite un effort inhumain. Je constate toutefois que certains codes finissent par faire l'unanimité, il suffit d'observer les systèmes utilisés dans les smartphones pour constater que ce sont des codes très anciens (BSD base d'iOS) ou s'appuyant sur des principes très anciens (UNIX base de Android). Je suis d'accord que ce ne sont pas des codes métier, mais on peut certainement analyser les raisons de leurs qualités pour savoir comment produire une meilleure informatique. Dans le cas des deux systèmes que je viens de citer, BSD et Linux, leur histoire nous montre que pour obtenir un bon code il faut que de très nombreux développeurs travaillent dans la même direction, on sait depuis la construction des pyramides qu'un groupe d'humains peut réaliser une tâche inhumaine.

C'est une question de spécialisation et d'organisation et, à mon sens, l'erreur en informatique se situe dans un manque de qualité du travail de codage et dans un pilotage inadapté. Depuis de trop nombreuses années, on applique un modèle de type manufacturier ou BTP aux grands projets informatiques, alors que l'on a affaire à une production purement cérébrale comme la littérature (ne parle-t-on pas d'éditeurs ?). La qualité des développeurs est une des clés de la réussite et je suis convaincu que sur un projet informatique, il vaut mieux ne rien faire tant qu'on n'a pas la bonne personne. Mais cette approche de la gestion des projets va à l'encontre d'une certaine vision du respect des délais, et elle va aussi à l'encontre de la fluidité du business dans un domaine qui est structuré à 90 % en activité de service, et où la mobilité du développeur prime sur sa compétence. Que penser d'un secteur qui a réussi à se mettre dans une situation où le maître d'œuvre gagne de l'argent tant que son chantier n'aboutit pas ?

Il nous faut donc des développeurs mieux formés et mieux pilotés. Il nous faut aussi de véritables architectes, pas de ceux qui font des « copier-coller » depuis leurs ouvrages sur les « Patterns », ou qui bêlent à la suite de soi-disant gourous de l'objet ou d'autre chose. Les bons architectes sont ceux qui arrivent à infiltrer le besoin métier pour en déduire une construction technologique utile. Je suis convaincu que ces développeurs et ces architectes sont la même personne qui est d'abord un passionné à la fois de code et de contacts humains et qui a su rester en prise sur la chose technique, en toute humilité.

Frédéric Fadel : Ce sont des remarques très justes. Effectivement parmi les logiciels les plus utilisés et réutilisés, on peut nommer les systèmes d'exploitation comme ceux cités par Thierry, ou d'autres, le plus abouti à mon sens étant Windows dans la version notamment d'Azure. Et la raison est simple, ce sont ces systèmes d'exploitation (et pas langages de programmation) qui nous débarrassent de tout un ensemble de détails techniques et nous facilitent la vie pour développer des softs plus spécifiquement utiles pour un métier particulier. On peut dire que dans tout logiciel, il y a un noyau dur commun (l'OS) qui est réutilisé. La question pour moi est : peut-on réutiliser plus ? Je pense que la réponse est « ça dépend du domaine ».

Par exemple, dans le monde du développement de jeux, en dehors des langages il y a les moteurs 2D, 3D, Physique... qui font que les créateurs de jeux peuvent déployer leur talent et leur imagination dans autre chose que le calcul des équations de mouvement ou la portée des ombres. Le talent et l'imagination du créateur du moteur, pièce importante et réutilisable, étant d'un autre type. On peut, aujourd'hui et maintenant, dans le monde des SI, isoler un moteur technique, le même pour tout le monde (au même titre que l'OS) qui s'occupera des aspects techniques et prévisibles. Ce moteur permettra aux développeurs d'applications métier d'aller de l'idée au produit, de façon continue, dès le

premier jour, sans cahier des charges, avec des cycles de livraison quotidiens voire plus.
Thierry conclut qu'il nous faut des développeurs mieux formés et mieux pilotés, je ne peux qu'être d'accord. Mais que veut dire mieux formé ? Pour moi c'est d'abord être moins déformé : pour pouvoir voir une solution, il faut déjà être capable de voir le problème. Le rôle du développeur n'est pas d'écrire du code, mais de garantir que les flux d'informations entrants et sortants de son système restent sains et cohérents.
Pour moi, être mieux formé, c'est :
Etre conscient qu'un SI est un système dynamique qui devrait vivre sa vie de façon quasi autonome et organique.
Être conscient que toute complexité cérémoniale due à des détails ou dogmes techniques devrait être soustraite du système.
Prendre très au sérieux les acronymes DRY et YAGNI dans ses développements, de façon globale et pas projet par projet.
Avec la mobilité, le « Cloud », la connectivité et des capteurs partout, l'informatique va « manger le monde ». Ne faisons pas l'erreur de fabriquer des usines keynésiennes, avec un informaticien en train de faire et de refaire tous les jours la même chose : nous n'avons pas besoin d'informaticiens spécialisés, mais d'informaticiens capables de transformer industriellement leurs solutions intellectuelles en produits métier utiles et utilisés.

Thierry Joubert : J'ai l'impression que nous sommes décidément d'accord sur bien des points mon cher Fred, même si je pense que le travail répétitif doit plus à Taylor qu'à Keynes; les calculateurs omniprésents et en réseau vont effectivement reposer la problématique du SI tel qu'il a marqué les 30 dernières années. Si le métier doit être pris en compte de manière plus fiable et plus agile, l'avènement de cette nouvelle informatique totalement diffuse va aussi soulever de vastes défis au niveau de l'optimisation et de la sécurisation des systèmes. Dans mon expérience des systèmes embarqués, j'ai souvent été confronté à ces défis et j'ai constaté qu'un des meilleurs principes est celui de la simplicité: après DRY et YAGNI, c'est celui qu'on nomme KISS. Mais attention de ne pas retomber d'un dogmatisme dans un autre, il est essentiel à l'avenir d'accorder le maximum de confiance dans les individus et non pas dans les méthodes.

Les réponses de Michael Thomas
J'ai lu avec attention des points de vue développés par Thierry et Frédéric. Bien que je partage la vision générale, à quelques positions près, certains points n'ont pas encore été

développés et me semblent importants par rapport à la question posée. Nous vivons un virage technologique sans précédent. Si je voulais être polémiste, j'irai même à dire que tout ce que nous avons connu et conçu ces 20 dernières années pour essayer d'orchestrer cette mutation numérique du traitement de l'information a été une erreur historique.

Bien sûr ce point de vue est volontairement caricatural, puisque raisonnablement, nous ne pouvons contester que le socle qui constitue la révolution numérique qui s'annonce, s'est principalement construit sur les retours d'expériences inédites conduites au cours de ces deux dernières décennies.

Une erreur Classe

De mon point de vue l'erreur que nous avons tous faite ces 20 dernières années a été de penser qu'il était possible d'abstraire nos applications des langages de projection.
Nous avons vécu une course à l'abstraction largement soutenue par les défendeurs du paradigme Objet par Classe et notamment les vendeurs d'implémentation Java et C#.
L'erreur n'est pas due au paradigme Objet, que je défends par et pour de nombreux aspects, mais par la volonté de fusionner les paradigmes de conception de ceux d'implémentation via des artefacts communs.
Cette volonté effrénée est née au moment de la première vague Internet et a été portée par les acteurs de la finance cherchant à rationaliser et à donner des métriques aux lourds investissements de recherche et développement qu'ils supportaient.
La communauté des développeurs s'est alors retrouvée aspirée dans une spirale du tout objet, mais surtout du tout Objet par Classe, c'est-à-dire contraint et frustré par un formalisme lourd et rigide (protection des variables, héritages et interfaces, getter/setter, security manager...)
Le paradigme Objet par Classe possède de nombreuses qualités, dont principalement celle de permettre une forte assistance de code (autocompletion, documentation en ligne) et de debuggage au moment de la compilation (et non du run-time).
Par contre tout cela à des conséquences. Je pourrais en citer des dizaines et des dizaines. Mais en synthèse, ce sont principalement les contraintes d'interfaces et la multiplication des librairies et frameworks intermédiaires, qui imposent d'innombrables encapsulations, copies mémoires d'objets, traitements de flux IO de manière synchrone, etc. qui ont structurellement condamné le paradigme par Classe. Les objets sont lourds et peu malléables, et contrairement à la vision originelle du paradigme objet, supportent mal le changement. Prenons pour exemple le simple changement d'une interface... Tout l'arbre d'implémentation s'en retrouve invalidé. Bien sûr des classes abstraites intermédiaires permettent de pallier élégamment aux problèmes de compilation, mais objectivement, c'est panser un problème créé uniquement du fait du langage... et on entre ainsi dans le merveilleux monde de la verbosité....
Mais que l'on soit bien clair, ce n'est pas le paradigme objet que je remets en cause, un paradigme proche de la théorie des ensembles en mathématique (ex. construction des réels), mais sa projection dans une implémentation par classe, qu'elle soit à héritage simple ou multiple.
Ce que je retiens du paradigme orienté objet, c'est une approche de modélisation de l'espace du problème, plutôt que la modélisation de la réponse à un problème. La solution émanant d'une approche orientée objet étant une instance de collaboration entre les objets du problème.
Par ailleurs, si l'on abstrait l'accès à des données en informatique au simple couple clé/valeur (que ce soit adresse mémoire donnée, ou clé valeur d'une map, etc.), si l'on accepte également qu'une valeur puisse être un état statique ou une fonction, le problème devient principalement le concept de namespace, c'est à dire d'espace de nommage et d'accès aux valeurs pour s'y retrouver.
L'approche orientée objet ne fait que proposer un mécanisme qui permet de formaliser ce namespace, auquel elle apporte une notion de contexte (le this), qui simplifie les accès entre les valeurs partageant la même portée, le même namespace.
Le C par exemple possède de nombreux avantages, mais pas celui-ci; quiconque a écrit des milliers de lignes de code dans ce langage acceptera qu'il ne le compte pas à son crédit.
En cela l'approche orientée objet est une avancée en terme de rationalisation par rapport à une approche purement C.

Des joints à fumer les transistors

Parmi les erreurs de parcours, nous ne pouvons pas non plus oublier la persistance SQL, inadaptée aux structures et à l'objet, et qui pourtant, avec le langage de templating PHP ont trusté les devants de la scène pendant plus de 10 ans.
Alors que nous remettons en cause les

paradigmes de programmation, n'oublions pas que du côté de la persistance, les technologies font également leur révolution.
D'une modélisation aplatie et générique, le SQL, nous retournons à des stratégies de persistances proches des structures que nous manipulons dans nos programmes : Queue, Stack, Map, Tree, Graph... Nous parlons alors de sérialisation native.
C'est le sens du courant NoSQL (Not Only SQL), qui est en train de s'installer naturellement dans toutes les architectures des grands sites Internet et des applications Big Data.
La raison est en très simple, et toute aussi proche que l'erreur faite par l'approche Objet dans son implémentation par Classe : penser que l'on pouvait rendre générique le mécanisme de persistance, indépendamment de la structure de l'élément ainsi que de la structure et de la volumétrie de l'ensemble.
Prenons un exemple concret, celui de la modélisation d'une facture en SQL; nous aurons à modéliser dans le cas le plus simple la table Order, OrderLine, OrderItem, Customer et Supplier. Pour reconstruire une facture, il faudra alors exécuter autant de requêtes qu'il y a de tables. Pour chacune d'elle, il faudra interroger des indexes dans lesquels, peut-être, des milliers (voir millions) d'autres enregistrements seront présents.
Qui pourrait raisonnablement défendre ce modèle d'implémentation de persistance pour ce type de donnée (Document). C'est sans doute la pire implémentation qu'il puisse exister. Pour le stockage des Graphes et des structures arborescentes, c'est encore pire.
Le SQL n'est ni plus ni moins qu'un ensemble de fichiers CSV avec des indexes. Il est impossible de représenter des structures complexes (c'est à dire des structures clé-valeur d'un niveau de profondeur supérieur à 1) sans avoir recours au mécanisme de jointures.
Et tout le problème du SQL est justement là, au niveau des jointures : elles imposent de multiples requêtes pour reconstruire des objets, et interdisent le sharding de collection pour mettre en place des architectures distribuées efficientes.
Il faut comprendre qu'il n'y a aucun dogme dans ma présentation, j'ai adoré le SQL, mais après 20 ans de collaboration, je ne peux que reconnaître son rôle majeur dans le passé, mais en même temps avouer son obsolescence dans sa forme actuelle pour les nouveaux défis informatiques que nous allons relever ces 20 prochaines années.

Code Forever

Le métier de développeur est intrinsèquement lié à sa connaissance des langages, de l'algorithmique, de la machine, des interfaces et des protocoles réseaux. Il est, à mon sens, risqué de penser que le métier de développeur pourrait évoluer en dehors de ce cadre, c'est à dire en s'abstrayant des compétences de l'un de ces socles.
Je ne dis pas que d'autres métiers, relatifs à la gestion et la configuration de workflow via des Frameworks dédiés ne vont pas se créer ou se pérenniser. Mais il s'agit d'autres métiers et non pas celui de développeur, et peut-être même pas de métiers rattaché au domaine de l'ingénierie informatique. Il y a un lieu commun qui me dérange et qui pointe souvent son nez à ce moment, celui qui dit que ce n'est pas nécessaire de réinventer la roue, et qu'il vaut mieux utiliser des librairies existantes plutôt que de refaire.
Bien que d'un point de vue microscopique, on ne peut qu'être d'accord pour peu que l'implémentation technologique présente stabilité et performance, par contre le point de vue ne résiste pas à une vision macroscopique : depuis 20 ans, les frameworks qui ont trusté le top 100 n'ont cessé de « réinventer » la roue, pour souvent réduire les dépendances, mais surtout pour embarquer des concepts à l'intérieur de nouveaux concepts. Réinventer la roue pour agréger et fusionner les concepts, c'est le fondement même du métier de développeur.
Tant que nous continuerons à innover et à créer de nouveaux concepts, ce cycle de création, agrégation, fusion restera au cœur du savoir-faire du développeur.
Ce n'est pas un sentiment qui m'anime, mais un constat cinglant des pyramides technologiques qui se sont bâties, puis systématiquement effondrées, donnant naissance à de nouvelles approches, de nouveaux paradigmes, de nouvelles librairies, plus simples et fonctionnelles.
Équipe-t-on nos voitures actuelles des premières roues mésopotamiennes datant de 3000 av. J.-C., ou n'a-t-on pas cessé de les repenser, les renouveler...
Penser qu'il n'est pas nécessaire de réinventer la roue, c'est défendre l'idée selon laquelle une technologie est une somme des technologies primaires qui la constituent. De mon point de vue, en informatique cette arithmétique se vérifie rarement, au mieux elle permet de livrer des logiciels qui répondent aux contraintes fonctionnelles, mais se révèlent totalement inefficace au niveau des contraintes non fonctionnelles (temps de réponse, haute disponibilité, ergonomie)...
En synthèse : on peut obtenir des logiciels qui possèdent les fonctionnalités, mais que les utilisateurs refusent d'utiliser, parce qu'inadaptés à l'expérience spécifique qu'ils souhaitent vivre.

Vers une approche polyglotte

L'informatique, avec la musique, est sans doute la discipline qui rapproche le plus les sciences des arts. Elle est le liant entre la technique et l'expérience.
Je recommande à tous les lecteurs de prendre connaissance, si ce n'est déjà fait, de l'initiative code.org pour s'en convaincre.
Dans la vie courante, pour donner une vision poétique, scientifique, politique, économique d'un fait ou d'une expérience, on utilise un langage spécifique pour le matérialiser et le décrire. En Informatique, c'est un peu la même chose, il n'y a pas un style, mais des styles et des langages.
Les logiciels qui gagneront demain seront bâtis par des architectes, qui n'utiliseront non pas un langage de programmation et un système de persistance pour tout rationaliser, mais qui utiliseront chacun d'eux, soit pour leurs caractéristiques techniques, soit pour la formalisation de l'expérience utilisateur. La force de l'architecte sera de maîtriser parfaitement, et connecter chacun des langages pour un flux de données et des conditions d'usage optimum.
Pour moi, l'informatique s'inscrit dans la littérature, puisqu'elle possède une esthétique, des figures de style, des champs lexicaux, etc... Elle est le nouvel art, un art qui rend les coups ;-)
Il n'y a pas d'informaticien sans langage, le langage est le {code}.

C++ 11 : une version aboutie

Le C++ est un langage vivant. Pour le prouver, si le standard ISO C++11 est disponible depuis trois ans (mars 2011), la roadmap pour les futurs standards C++14 et C++17 a été annoncée récemment par Herb Sutter, le secrétaire de standardisation du langage C++ de l'ISO. Le comité ISO a en effet décidé de délivrer de nouvelles versions du standard plus rapidement, que ce soit au niveau des extensions du langage, que de la librairie STL (Standard Template Library).

Le support de C++11 dans les compilateurs

Le support du C++11 par les principaux compilateurs du marché - GCC, CLang et Visual C++ - est maintenant assuré en totalité. L'implémentation des fonctionnalités ne s'est pas faite en une seule fois; vérifiez bien le support de conformité de votre compilateur si vous n'utilisez pas la dernière version. C++11 regroupe à la fois le langage C++ et la librairie STL. La librairie STL exploite certaines nouvelles fonctionnalités du langage (la sémantique de déplacement par exemple).

C++11 et STL

En C++ moderne, l'utilisation de la librairie STL est implicite car elle fournit, par exemple, les classes string et les « containers » comme vector, list, set, map et le support de tables de hash. Il est admis que ces collections sont ce qui se fait de fait de plus efficace. La STL fournit aussi tout un ensemble d'algorithmes pour le parcours ou la transformation des ensembles de données.

Les lambdas en C++

Le C++ dispose maintenant de la fonctionnalité des lambdas. Une lambda est une fonction anonyme qui n'est pas nommée qui peut prendre des arguments et qui possède un corps de fonction et un type de retour. Le type de retour par défaut est void, mais si vous devez définir un autre type de retour, il faut préciser ce type avec la syntaxe de la flèche ->. En plus des arguments passés à la fonction, une lambda définit une liste de capture décrite entre []. Voici une fonction lambda simple utilisée avec une opération for each :

```
void Use_Lambda()
{
  vector<string> v;
  v.push_back(«Edith»);
  v.push_back(«Lisa»);
  v.push_back(«Audrey Maggy»);

  std::for_each(v.begin(), v.end(), [](string str) -> void {
    cout << str << endl;
  });
}
```

Dans cet exemple, la liste de capture est vide. Si la syntaxe [&] est utilisée, toutes les variables locales seront passées en référence. Techniquement, une fonction lambda est définie dans une classe avec une surcharge de l'opérateur ().

Références RValue et move constructor

Le C++11 introduit la notion de référence rvalue, identifiée par T&&. Cette notion permet de mettre en œuvre la sémantique de déplacement (move semantic). Le résultat c'est que nous pouvons librement déplacer les ressources d'une référence rvalue vers un autre objet. Cette mécanique de déplacement est particulièrement utile pour éviter la copie de variable passée par valeur qui peut être couteuse en termes de performance. Le meilleur exemple consiste à retourner un vecteur créé dans une fonction. En C++03, les éléments du vecteur sont copiés les uns derrière les autres. En C++11, le vecteur est déplacé. Cette opération est supportée parce que le vecteur possède un constructeur de déplacement (move constructor) qui prend en paramètre une référence rvalue à un vecteur.

```
vector<string> Use_VectorReturn()
{
  vector<string> local_vector;
  local_vector.push_back(«Didou»);
  local_vector.push_back(«Liz»);
  local_vector.push_back(«Maggy»);

  printf(«local_vector=0x%08x\n», local_vector);
  return local_vector;
}

int _tmain(int argc, _TCHAR* argv[])
{
  vector<string> v = Use_VectorReturn();
  printf(«v=0x%08x\n», v);
  return 0;
}
```

La sortie de ce programme prouve que l'adresse du vecteur a été « volée ». Dans main, le vecteur pointe sur la même adresse que la variable locale de la fonction Use_VectorReturn().

```
local_vector=0x00590a00
v=0x00590a00
```

Dans notre exemple, le vecteur ne contient que très peu d'éléments. Dans la vraie vie, un vecteur peut contenir une quantité importante d'éléments et cette mécanique qui évite les copies locales permet de gagner en performance. Il faut noter que cette sémantique de déplacement est transparente pour l'utilisateur car les containers STL du C++11 disposent tous d'un constructeur de déplacement, mais aussi d'un opérateur de copie de déplacement (move-assignement operator). Une opération de déplacement exécute un « vol » de ressources et n'alloue aucune ressource. Ces opérations de déplacement n'ont pas vocation à lever des exceptions. Pour cela, nous devons le préciser en utilisant le nouveau mot-clé noexcept à la fois dans le fichier h et le fichier cpp. La librairie STL du C++11 introduit la fonction std::move() définie dans le fichier d'entête utility, qui permet de retourner un référence rvalue.

Expression constante

Le nouveau mot-clé constexpr garantit qu'une fonction ou que la construction d'un objet soit une opération constante garantie à la compilation. Le corps de la fonction doit contenir exactement une instruction return. Le compilateur remplace l'appel à une fonction constexpr directement par sa valeur.

Template marqué extern

Dans les applications importantes, l'instanciation d'un même template dans plusieurs fichiers peut rendre le temps de compilation important. Pour améliorer cela, il est possible de faire référence à un template dans différents fichiers sans pour autant que le compilateur ne génère le code nécessaire à l'instanciation de celui-ci. Il suffit alors de préfixer la définition du template comme extern.

Initialisation avec {}

Il est possible d'initialiser une variable ou un container STL comme vector, list ou map avec la syntaxe {} comme pour les structures en C :

```
void Use_Initialisation()
{
  int i = {0};
  vector<int> v = { 10, 20, 30, 40, 50 };
  list<string> mes_chats = { «Tom», «Pti'Gris», «Romeo» };
}
```

L'initialisation de liste est implémentée via l'utilisation de la classe std::initializer_list comme constructeur.

Utilisation de auto

Le mot clé auto est certainement la fonctionnalité qui va avoir le plus de succès en termes d'adoption.
L'utilisation de auto permet de laisser le compilateur déduire le type de la variable déclarée.

```
void Use_auto()
{
  vector<int> v = { 10, 20, 30, 40, 50 };
  //for (vector<int>::iterator it = v.begin(); it != v.end(); it++)
  for (auto it = v.begin(); it != v.end(); it++)
  {
    int v = *it;
    cout << «value = « << v << endl;
  }
}
```

Le mot clé auto peut être utilisé avec const et &.

Evolution du for

Le parcours d'un ensemble de données avec for est disponible depuis que le C existe. C++11 introduit une variante pour le parcours des ensembles au travers de l'utilisation du mot clé for. La nouvelle syntaxe ressemble à cela :

```
void Use_RangeFor()
{
  vector<string> langs = { «VB6», «C#», «C++» };
  for (string s : langs )
  {
    cout << s << endl;
  }
}
```

Evolution du constructeur

C++11 introduit la notion de délégation de constructeur. En C++03, un constructeur non trivial affecte des valeurs à des variables membres. S'il existe plusieurs constructeurs, on se retrouve à dupliquer le code qui affecte les valeurs par défaut à certaines variables membres.

Classe marquée final

Si vous définissez une classe et que vous ne voulez pas qu'elle soit dérivée, il suffit d'ajouter le mot clé final après votre nom de classe. Si une classe essaie de dériver de celle-ci, le compilateur sortira en erreur.

```
class Bar final
{
};

// Illegal construction !!! Compilation error !!!
class DerivedBar : public Bar
{
};
```

Constante de pointeur nul

Par convention et par compatibilité avec le langage C, un pointeur nul vaut 0. La variable de préprocesseur NULL est utilisée et vaut 0. Le C++11 introduit le mot clé nullptr pour remplacer l'usage du NULL mais ne vaut pas 0 !

Template variadic

Un template variadic est une fonction ou classe template qui prend un nombre d'arguments variables. La syntaxe ... est utilisée pour définir les paramètres variables.

Séquences de chaines de caractères

La gestion de chaînes de caractères complexes pose un problème en C ou en C++. En effet, les séquences avec le caractère '\' posent problème. Le caractère '\' doit être doublé pour ne pas rentrer en conflit avec les séquences comme '\n', '\r', ou \'t'. De plus, on ne peut pas déclarer une chaîne de caractères sur plusieurs lignes. C++11 répond à ce problème en fournissant un support pour les « raw string literals ». Il est maintenant possible de déclarer le contenu d'un document XML directement dans le code. C++11 répond à cela avec une séquence qui ressemble à R»(« contenu »)»:

```
  string strXML1 = R»(<DataApollo>
  <Info>
    <PriceUnit>euro</PriceUnit>
    <VolumeUnit>TW0</VolumeUnit>
    <RunCode>2012-03_00_avantEBO_V1_20130110</RunCode>
    <RunOption>am</RunOption>
    <GenerationDate>24-02-2014</GenerationDate>
    <CoreVersion>2.3.3.894</CoreVersion>
    <ReturnCode>0</ReturnCode>
  </Info>
</DataApollo>
)»;
```

Les tuples

La librairie STL introduit la notion de tuple au travers la class std ::tuple. Cette implémentation est réalisée en utilisant les templates variadic. La classe std::tuple prend un nombre d'argument variables.

```
void Use_Tuple()
{
  tuple<string, string, int> maggy( «Maggy», «super coquine», 3);
  string str = get<1>(maggy);
  cout << str << endl;
}
```

Le type array

La classe array est définie dans le fichier d'entête array. Ce type permet de gérer une séquence d'éléments de taille fixe. C'est comme un tableau standard mais avec le support de la notion d'itérateur. Ce type est particulièrement adapté pour la programmation embarquée. Il supporte l'initialisation sous forme de liste :

```
void Use_Array()
{
  array<int, 10> ar = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
  for (auto it = ar.begin(); it != ar.end(); it++) {…}
  int a2 = ar[2]; int a3 = ar.at(3);}
```

Les hash tables

C++11 regroupe une seule et même notion de hash tables au travers des fichiers d'en-tête « unordered_set » et « unordered_map ». Les anciens fichiers « has_set » et « hash_map » sont obsolètes.

Les expressions régulières

Le support des expressions régulières est assuré au travers du fichier d'en-tête « regex ». Regex peut utiliser les grammaires des expressions régulières suivantes : ECMAScript, POSIX standard, étendu, awk, grep et egrep. Les fonctions regex_match() et regex_search() déterminent si une séquence de caractères correspond à une regex donnée. La fonction regex_replace() permet de trouver et remplacer une expression régulière dans une séquence. Regex utilise aussi la notion d'itérateur au travers la classe regex_iterator.

Les pointers intelligents (smart pointers)

Le support des « smart pointers » est assuré au travers du fichier d'en-tête « memory ». Les templates les plus utiles sont shared_ptr<T>, unique_ptr<T> et weak_ptr<T>. Le but avoué de cet effort de pointeurs intelligent est de supprimer l'usage du delete et de permettre la libération automatique de la mémoire. La gestion mémoire des objets peut être confiée aux pointeurs intelligents. Ils s'occuperont tout seul de la suppression des objets le temps venu. Le template shared_ptr<T> fonctionne avec l'ensemble des types C++ et ne nécessite aucune plomberie préalable dans le type qui souhaite en tirer parti. Le template shared_ptr<T> permet aussi, pour certains cas bien spécifiques, de spécifier la manière d'allouer ou de libérer le type T. Bref, il est possible d'allouer ou de libérer n'importe quel type de ressources. Pour gérer un pointeur intelligent, il suffit de d'encapsuler son type T avec le template shared_ptr<T> et d'utiliser la fonction make_shared<T>() à la place de new(). L'accès au type T se fait via l'opérateur -> donc le type T est utilisé comme un pointeur standard. Exemple :

```
void Use_SharedPtr()
{
  shared_ptr<MyClass> ptr = make_shared<MyClass>();
  ptr->data = «Maggy est une coquine !»;
  // no delete !!!
…
}
```

Dans cet exemple, la fonction delete n'est pas appelée, et le shared_ptr<T> s'en charge automatiquement. Il est possible d'affecter des shared_ptr et leur fonctionnement interne est basé sur le comptage de référence. Le compteur est automatiquement incrémenté ou décrémenté suivant les cas d'utilisation, et l'objet géré est libéré quand le compteur de référence tombe à 0.

Les threads et mutex

Le support des threads est assuré au travers du fichier d'entête thread. La classe thread encapsule le thread et sa routine de traitement associée en tirant parti du support des threads fournit par le système d'exploitation. Les API Linux et Windows sont différentes donc l'utilisation de std::thread permet d'uniformiser la programmation parallèle. La gestion des verrous est assurée au travers du fichier d'entête mutex. Un mutex est un objet qui représente un accès exclusif à une ressource. Il peut être utilisé pour protéger l'accès à une donnée et synchroniser l'accès aux données partagées entre plusieurs threads. Pour démarrer un thread, il suffit de lui passer le nom de la routine qui sera dédiée au traitement et éventuellement, une liste de paramètres à prendre en entrée de cette routine. La déclaration du thread provoque immédiatement son lancement et l'appel à la méthode join() permet d'attendre la fin du traitement.

```
void FnThread(vector<string> v)
{
  thread::id id = this_thread::get_id();
  for (string s : v) { cout << s << endl; }
}

void Use_Thread()
{
  vector<string> mes_filles = { «Edith», «Lisa», «Audrey» };
  thread t(FnThread, mes_filles);
  t.join();
  cout << «Thread finished !» << endl;
}
```

Il faut noter que l'implémentation des threads fournie par les dernières versions de Visual C++ utilise le Concurrency Runtime (ConCRT) alors que la librairie Boost tire partie des interfaces systèmes PThreads sous Linux et des API système Win32 sous Windows.

Méta programmation et type traits

Le fichier d'entête type_traits contient tout ce qu'il faut pour faire de la méta programmation. Ce sujet dépasse le cadre de cet article mais mérite d'être précisé.

C++14 et C++17

Le standard C++ évolue. C++14 sera une release mineure. C++17 va intégrer de nouvelles librairies dans la STL. Ces librairies sont inspirées de Boost comme FileSystem ou Network. Il y aura aussi un support des algorithmes parallèles.

Environnements de compilation C++11

Si vous êtes sous Linux ou Mac, vous avez la possibilité d'utiliser GCC ou CLang. Pour CLang, une version 3.x vous assure un maximum de confort. Pour GCC, une version 4.7 (sortie en mars 2012) est largement confortable. Il est possible de faire tourner GCC sous Windows via MinGW (http://mingw.org). Concernant Windows, Microsoft distribue Visual C++ dans la dernière version de Visual Studio 2013. Je vous conseille d'opter au minimum pour la version Visual Studio 2012 car la version Visual Studio 2010 n'offre qu'un support partiel. Il existe une version Visual Studio 2013 Express « for Desktop » qui est gratuite et qui tourne sous Windows 7 SP1 et plus.

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

Le futur de PHP

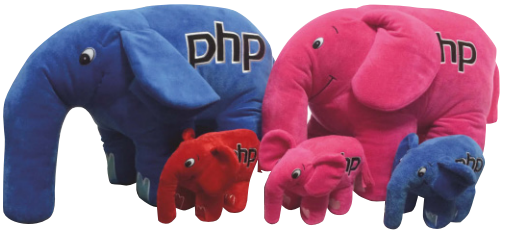


photo : © françois tonic

Avec 81 % de parts de marché (source : W3Techs), le langage PHP est le langage web le plus utilisé dans le monde. De plus, il est toujours bien placé dans l'index TIOBE.

Pour rester toujours réactif au web, le langage PHP a dû s'adapter à l'Internet moderne. Tout d'abord en réduisant le délai de développement de ses versions principales. Les versions mineures sont ensuite publiées mensuellement, sauf cas exceptionnel comme une faille de sécurité importante. Bien entendu, ces versions apportent principalement des correctifs liés au langage et aux fonctionnalités.
Au niveau des publications majeures, le cycle de publication est planifié tous les ans, voire 1 an et demi. Cependant ce qui ne change pas, concerne les versions importantes, car depuis l'année 2004, le langage PHP utilise le même que Zend Engine, c'est à dire PHP 5.

Pour 2014

En 2013, la communauté a publié la version PHP 5.5 et depuis le début 2014, la version 5.6 a été mise sur les rails avec un planning précis :

- Janvier 2014 : 1ere version alpha
- Mars 2014 : 1ere version beta
- Mai 2014 : 1ere version RC
- Mi-juin 2014 : version stable

La liste des fonctionnalités prévues est connue et disponible à l'adresse suivante : https://wiki.php.net/todo/php56

Débogueur

Le débogage est un des points sensibles au niveau des développements car cette fonction vous aide à trouver des erreurs lors de la réalisation de projets. PHP 5.6 se voit doter de nouvelles possibilités pour aider les développeurs et développeuses PHP :

phpdbg

Phpdbg est un débogueur PHP, qui s'exécute dans un terminal. Il s'agit d'un module qui pourra être utilisé à côté de CLI et des autres modules SAPI. Il offre différentes possibilités dont celle de déboguer en mode pas à pas. Bien entendu, l'utilisateur peut contrôler et inspecter l'exécution à travers de nombreuses fonctionnalités. Exemple :

```
phpdbg> break [file] test.php:1
phpdbg> b [F] test.php:1
```

L'exemple montre l'utilisation d'un point d'arrêt à la ligne 1 du fichier test.php. Source : https://wiki.php.net/rfc/phpdbg

debugInfo ( ) méthode magique

Il s'agit d'une méthode magique car elle va fournir des informations supplémentaires pour les fonctions suivantes : var_dump() ou print_r(). Les données affichées en plus que vous pouvez obtenir sont des données venant d'un objet CLASS.

```
$f = new File;
var_dump($f); // object(File)#1 { }
$f->open('http://php.net');
var_dump($f);

/*
object(File)#1 {
  [«wrapper_type»]=>
  string(4) «http»
  [«stream_type»]=>
  string(10) «tcp_socket»
  etc...
*/
```

Source : https://wiki.php.net/rfc/debug-info

La sécurité

Un problème récurrent touche les failles de sécurité ; le langage PHP propose régulièrement des améliorations sur ce point à travers de nouvelles fonctionnalités pour sécuriser votre code.

Crypt() function salt

Il s'agit d'une amélioration sur la fonction crypt() qui permet de créer un mot de passe faible si aucun paramètre d'option n'est utilisé. A partir de la prochaine version, une alerte sera affichée de niveau E_NOTICE si l'option SALT n'est pas renseignée. La fonction SALT qui se positionne dans les critères d'options va affiner le niveau de hachage et des chiffrements suivant le type choisi. Source : https://wiki.php.net/rfc/crypt_function_salt

TLS Peer Verification

Le cryptage de flux client PHP n'est pas sécurisé par défaut. Le RFC explore la nature problématique de son utilisation actuelle et propose des correctifs à travers de nouvelles fonctionnalités pour améliorer la sécurité de celle-ci. Le résultat final est une mise en œuvre plus sécurisée à travers la fonction 'TLS Peer Verification' sans que le développeur soit un expert de la sécurité et sans se préoccuper du niveau de cryptage de la valeur des données.
Pour cela, le développement s'effectue de la manière suivante :

```
<?php
$uri = 'https://www.votreSite.com/';
$html = file_get_contents($uri);
?>
```

A partir de PHP 5.6 :

```
<?php
$uri = 'https://www.votreSite.com/';
$ctx = stream_context_create(['ssl' => [
  'verify_peer' => true,
  'cafile' => '/emplacementDuFichier',
  'CN_match' => 'www.votreSite.com/'
]]);
$html = file_get_contents($uri, FALSE, $ctx);
?>
```

Source : https://wiki.php.net/rfc/tls-peer-verification
L'exemple montre que vous transférez le contenu complet d'un fichier (à partir d'un Upload) vers un autre site ou de l'affichage d'une page, son contenu va être mieux sécurisé grâce aux nouvelles options possibles de la fonction stream_context_create().

Improved TLS Defaults

La communication à travers les flux TLS peut s'utiliser en PHP. Cependant le cryptage des paramètres par défaut est potentiellement dangereux. Cette fonction est transparente pour l'utilisateur car vous ne la voyez pas. Son but est d'améliorer un peu plus la transmission des flux cryptés et de les mettre

en œuvre par défaut en toute transparence. Il s'agit d'une amélioration de la fonction 'TLS Peer Vérification RFC' pour éviter les pièges potentiels TLS. A partir de l'exemple précédent, vous pouvez ajouter l'option 'ciphers' dans la fonction stream_context_create(), comme ceci :

```
$context = stream_context_create(['ssl' => [
  'ciphers' => 'HIGH:MEDIUM:LOW@SPEED'
]]);
$html = file_get_contents('https://somesite.com', null, $context);
```

Source : https://wiki.php.net/rfc/improved-tls-defaults

Divers

Slim POST data

Il s'agit d'une nouvelle fonction PHP, car son but est d'améliorer l'utilisation de la mémoire au niveau de la réception des charges utiles HTTP. Le gain de performance que vous allez gagner est de 200 - 300 %, ce qui est très utile au niveau d'un flux temporaire. Elle pourrait s'utiliser de la manière suivante :

```
$GLOBALS[«HTTP_RAW_POST_DATA»] = file_get_contents(«php://input»);
```

Source : https://wiki.php.net/rfc/slim_post_data

RFC : Argument unpacking

Il s'agit d'une évolution de la fonction RFC, en proposant une syntaxe pour la manipulation des tableaux dans une liste d'arguments, sur le même principe que « l'opérateur de diffusion ». C'est à dire, jusqu'à maintenant la fonction user_func_array ne peut utiliser que ce type de méthode :

```
call_user_func_array([$db, 'query'], array_merge(array($query), $params));
```

C'est à dire que vous disposez d'une requête $query et un tableau de paramètres et vous voulez une méthode d'utilisation dans la variadics RFC. La solution actuelle se dirigerait sous cette forme

```
$db->query($query, ...$params);
```

Cette syntaxe permet de déballer ses arguments directement dans la syntaxe d'appel. Source : https://wiki.php.net/rfc/argument_unpacking

Importing namespaced functions

Les espaces de noms, aussi appelés Namespace, bénéficient d'une évolution intéressante pour cette nouvelle version à partir de la fonction 'importing namespaced functions'. Cette fonctionnalité offre jusqu'à présent la possibilité d'importer des NameSpace et les types (Classes, interfaces, traits). Maintenant vous pourrez importer les fonctions dans les espaces de noms au travers d'une nouvelle séquence.
Source : https://wiki.php.net/rfc/use_function

Default character encoding

La norme de codage PHP au niveau de la manipulation des caractères, est définie par défaut en UTF-8, depuis la version Php 5.4. La mise en place du choix par défaut était utile pour les fonctions htmlentities / htmlspecialchars. Cependant, vous pouvez configurer le codage dans le fichier php.ini. Toutefois certains utilisateurs peuvent quand même mélanger ISO-8859-1 et UTF-8 et indirectement provoquer des problèmes de sécurité. C'est pourquoi, le codage de caractère ne doit pas être pris à la légère. La solution proposée avec la prochaine version, est d'utiliser 'default character' au niveau du codage des caractères par défaut. Source : https://wiki.php.net/rfc/default_encoding

Constant Scalar Expressions (with constants)

Cette RFC propose d'ajouter le support des expressions scalaires constantes avec le soutien de constantes. Le code est plus facile à écrire et à comprendre, en permettant pour le code d'avoir un code beaucoup plus expressif. La différence par rapport au code existant, c'est que les constantes peuvent être impliquées dans les opérations scalaires. Cela se présente de la manière suivante :

```
<?php
const a = 1;

const b = a?2:100;

?>
```

L'exemple montre que la constante 'b' est dépendante de la constante 'a' source : https://wiki.php.net/rfc/const_scalar_exprs

Les attentes des développeurs

Les attentes des développeurs sont nombreuses et variées car le site PHP propose de recueillir les différentes demandes pour les soumettre à la validation de l'équipe Core du langage. Cette page est transparente et accessible à travers ce site internet http://php-vote.com/
Ce site se compose entre-autres d'une page de saisie des demandes, et d'une page qui liste les propositions avec les votes. Comme ceci, vous pouvez voir les orientations et connaître les attentes et les tendances du web.

Unicode

Le projet de l'unicode en natif est un énorme chantier puisqu'il a été annoncé pour PHP 6. Cependant, la Team de PHP a du provisoirement mettre ce chantier en sommeil pour laisser le temps aux équipes de se pencher sur cette demande. Pour rappel l'unicode est un standard informatique qui permet d'effectuer des échanges de textes dans différentes langues. Il existe différentes interrogations venant de l'équipe CORE pour intégrer cette fonctionnalité. Dans les pistes assez avancées :

- L'intégration d'une extension déjà disponible. C'est pourquoi actuellement différents membres effectuent des tests de faisabilité.
- La piste de tout reprogrammer est toujours d'actualité et revient de temps en temps dans les discussions.

Actuellement, le choix n'est pas défini, mais c'est un chantier toujours vivant.

Les attentes intéressantes

Les autres attentes sont :

- La méthode overloading existe déjà dans certains langages (Java, Ada, C++, c#). Elle permet de créer plusieurs méthodes avec le même nom, mais se différencie suivant le type d'entrée et de sortie de la fonction. Ainsi, elle se définit comme une capacité d'une fonction à effectuer des tâches différentes.
- Le typage est aussi une autre attente, car il est apparu avec PHP 5.1 et a évolué avec PHP 5.4. Cependant il a été remarqué qu'il manquait des types scalaires, comme le Type hints (des notes). Les autres types sont Int, string, Traits...
- Avec le web moderne, il est de plus en plus nécessaire d'utiliser une classe enfant qui hérite d'une classe mère. Cette méthode s'appelle 'Polymorphisme', ainsi les implémentations entre mère et enfants sont différentes.

Par ailleurs, l'évolution des arguments dans les fonctions, exit, utilisateur sont aussi des petits détails qui font la différence par rapport aux techniques de développement.

Conclusion

Le langage est en constante évolution comme le web avec des cycles de réalisation réguliers, le langage PHP montre son avancement et vous pouvez le constater au niveau de l'utilisation directe et indirecte avec les Frameworks et CMS.

Christophe **Villeneuve**
Consultant IT pour Neuros, auteur du livre « PHP & MySQL-MySQLi-PDO, Construisez votre application », aux Éditions ENI. Rédacteur pour WebRIVER, membre des Teams DrupalFR, AFUP, LeMug.fr, Drupagora, PHPTV..

Quels avenirs pour .NET 5.0, C# 6 et Visual Basic 13 ?

Cet article présente des propositions d'évolutions provenant de différentes sources Microsoft. Il est important de se rappeler que toutes les nouvelles fonctionnalités explicitées sont, à ce jour, seulement des propositions. Vous trouverez ici les probables évolutions des langages C# 6 et Visual Basic 13.

Proposition des futures fonctionnalités de C# 6

Version	Année	.NET Framework	Visual Studio	Fonctionnalités (liste non exhaustive)
C# 1	2002	1.0 / 1.1	VS2002 / VS2003	Code managé
C# 2	2005	2.0	VS2005	Génériques Méthodes anonymes Itérateurs Types nullables
C# 3	2007	2.0 / 3.0 / 3.5	VS2008	Expressions lambda Méthodes d'extensions Types anonymes Typage implicite
C# 4	2010	4.0	VS2010	Binding dynamique Arguments nommés et optionnels Covariance et contravariance
C# 5	2012	4.5 / 4.5.1	VS2012 / VS2013	Méthodes asynchrones Attributs d'informations de l'appelant
C# 6?	2014?	5.0?	VS2014?	?

La prochaine version 6 de C# devrait intégrer Roslyn, un nouveau compilateur complètement écrit en C# (anciennement écrit en C++). Roslyn ouvre de nombreuses nouvelles opportunités et est actuellement disponible en version preview CTP (http://msdn.microsoft.com/en-us/vstudio/roslyn.aspx). Les évolutions possibles liées à la prochaine version du langage C# (http://damieng.com/blog/2013/12/09/probable-c-6-0-features-illustrated) ont été évoquées lors de la Norwegian Developers Conference (NDC) en décembre 2013 à Londres par Mads Torgersen (Program Manager pour le langage C# chez Microsoft Corp.). Les paragraphes suivants font état des principaux changements qui devraient voir le jour.

Les constructeurs primaires

Cette modification permettrait de s'affranchir de la création explicite du constructeur. Le développeur pourra alors placer les paramètres directement après la définition du nom de la classe. Le code d'initialisation pour copier les paramètres du constructeur dans les champs privés associés devient alors inutile.

C# 5

```
public class Point
{
  private int x, y;

  public Point(int x, int y)
  {
    this.x = x;
    this.y = y;
  }
}
```

Possible implémentation en C# 6

```
public class Point(int x, int y)
{
  private int x, y;
}
```

Les propriétés en lecture seule

Pour définir des propriétés avec des champs en lecture seule, le développeur pourra déclarer les propriétés, et leur champ de stockage, dans une seule ligne de code grâce aux propriétés automatiquement implémentées en lecture seule.

C# 5

```
private readonly int value;
public int Value { get { return value;
} }
```

Possible implémentation en C# 6

```
public int Value { get; } = value;
```

L'import de types statiques

Visual Basic et JavaScript permettent déjà l'import des modules (classes statiques en C#) dans un espace de nom. C# 6 pourrait hériter de cette fonctionnalité. Le code répétitif lors de l'appel à une méthode statique est ainsi diminué. Le développeur pourrait dans ce cas supprimer le code redondant tel que le préfixe « Math. », comme dans l'exemple ci-dessous.

C# 5

```
public double Value { get { return Math.Sqrt(Math.Round(5.142)); } }
```

possible implémentation en C# 6

```
using System.Math;
public double Value { get { return Sqrt(Round(5.142)); } }
```

Les propriétés d'expression

Les expressions lambda appliquées aux propriétés devraient réduire et rendre plus lisible le code nécessaire lors de la définition d'une propriété calculée en lecture seule. Les propriétés paramétrées ne sont toutefois pas encore concernées, cela est disponible en Visual Basic mais non envisagé à court terme pour C#.

C# 5

```
public double Distance
{
  get { return Math.Sqrt((x * x) + (y * y));
}
```

Possible implémentation en C# 6

```
public double Distance => Math.Sqrt((x * x) + (y * y));
```

Les méthodes d'expression

Les expressions lambda appliquées aux méthodes devraient fournir les mêmes fonctionnalités que les propriétés d'expression et supporter le passage de paramètres.

C# 5

```
public Point Move(int dx, int dy)
{
  return new Point(x + dx1, y + dy1);
}
```

Possible implémentation en C# 6

```
public Point Move(int dx, int dy) => new Point(x + dx, y + dy);
```

IEnumerable<T> comme paramètres de fonction

L'interface IEnumerable devrait également être mise à jour afin de supporter

le passage de paramètres. Il devient alors possible de l'utiliser comme paramètre de fonction à la place des tableaux comprenant le mot-clé « params ». Néanmoins, les autres langages .NET tels que Visual Basic auraient aussi besoin de se mettre à jour pour supporter cette fonctionnalité.

C# 5

```
Do(someEnum.ToArray());
...
public void Do(params int[] values) { ... }
```

Possible implémentation en C# 6

```
  Do(someEnum);
...
public void Do(params IEnumerable<Point> points) { ... }
```

La vérification monadique (Safe Navigation Operator)

Lors de la programmation, il convient souvent d'analyser et de vérifier les données d'entrée avant de déclencher le traitement. La nouvelle syntaxe « ?. » simplifie et automatise la vérification de la nullité des données d'entrée. Plus besoin d'écrire une série de vérifications avant d'invoquer une méthode ou de lire une propriété. Dans son blog, Jerry Nixon (Developer Evangelist chez Microsoft Corp.) explique de manière plus détaillée le principe : http://blogs.msdn.com/b/jerrynixon/archive/2014/02/26/at-last-c-is-getting-sometimes-called-the-safe-navigation-operator.aspx.

Cette approche existe dans les langages Smalltalk et Objective-C et permet de réduire la quantité de code à écrire. Toutefois, la détection des exceptions du type « NullReferenceException » peut s'avérer plus compliquée. D'autre part, la lisibilité du code devient moins évidente.

Le 25 février 2014, Mads Torgesen a confirmé sur le UserVoice de Visual Studio que son équipe envisageait sérieusement l'intégration de cette fonctionnalité dans C# 6 et Visual Basic 13. Ce serait une excellente nouvelle, car cette fonctionnalité est réclamée depuis longtemps par la communauté. Dans l'exemple ci-dessous, si « points » est « null », ou si l'appel à la méthode « points.FirstOrDefault()» renvoie « null » et la valeur de la propriété « next.X » est « null », la valeur « -1 » est retournée. Sinon la valeur de la propriété « next.X » est retournée.

C# 5

```
if (points != null)
{
  var next = points.FirstOrDefault();
  if (next != null && next.X != null)
    return next.X;
}
return -1;
```

Possible implémentation en C# 6

```
  var value = points?.FirstOrDefault()?.X ?? -1;
```

Inférence de type sur les méthodes génériques statiques

Cette évolution permettrait de s'affranchir de la création des méthodes statiques de type « factory » pour en déduire les types génériques. Ceci sera applicable pour l'utilisation de tuples.

C# 5

```
var example = MyClass.Create(1, «Example»);
public MyClass<T1, T2> Create<T1, T2>(T1 a, T2 b)
{
  return new MyClass<T1, T2>(a, b);
}
```

Possible implémentation en C# 6

```
  var example = new MyClass(1, «Example»);
```

Déclaration implicite des paramètres utilisant le mot-clé « out »

Ici, les variables locales peuvent être déclarées implicitement lors de l'utilisation du mot-clé « out ».

C# 5

```
int value = 0;
int.TryParse(«123», out value);
```

Possible implémentation en C# 6

```
int.TryParse(«123», out int x);
```

Proposition des futures fonctionnalités de Visual Basic 13

L'homologue de Mads Torgersen, Lucian Wischik (VB Language Designer chez Microsoft Corp.) a évoqué les possibles évolutions de Visual Basic (http://blogs.msdn.com/b/lucian/archive/2013/12/13/some-potential-language-ideas-for-future-versions-of-vb.aspx). Elles tournent surtout autour de la simplification et de la réutilisabilité du code. Il n'y aura pas de modifications majeures comme lors du passage de la version 10 à la version 11.

Les constructeurs primaires

Le principe est le même que pour C# 6. Il s'agirait de ne plus avoir à écrire de constructeur mais de déclarer les paramètres directement après la définition du nom de la classe. Le code d'initialisation pour copier les paramètres du constructeur dans les champs privés associés devient alors inutile. Voici une proposition d'implémentation en VB 13 :

```
class  Point ( int  x,  int  y)
{
  public  double  X {  get ; } = x;
  public  double  Y {  get ; } = y;
  public  double  Z {  get ; } =  Math .Sqrt(X * X + Y * Y);
}
```

Les propriétés en lecture seule

Les propriétés implémentées automatiquement (ou auto-propriétés) et en lecture seule seront également implémentées dans la prochaine version de Visual Basic. Il suffira de faire précéder le mot clé « ReadOnly » à la déclaration d'une propriété afin de restreindre sa portée au moment de la création de l'objet (dans le constructeur). Actuellement, lorsqu'on définit une propriété implémentée automatiquement, elle est systématiquement en lecture / écriture (« ReadWrite »).

Cette évolution permettrait de déclarer cette propriété en lecture seule (« ReadOnly »). Dans ce cas, le compilateur déclarera implicitement une propriété privée (« backing field »). Il serait également possible d'assigner cette auto-propriété dans le constructeur, et la propriété privée serait alimentée implicitement. De plus, le développeur pourra déclarer les propriétés et leur champ de stockage dans une seule ligne de code grâce aux propriétés automatiquement implémentées en lecture seule.

Voici une proposition d'implémentation en VB 13 :

```
Class  Customer
  ReadOnly Property  Name  As  String
  ReadOnly Property  Address  As  String

  Sub  New (name  As  String , address  As  String )
    Me .Name = name
    Me .Address = address
```

```
  End  Sub
End  Class
```

IEnumerable<T> comme paramètres de fonctions

De même que pour C# 6, il s'agirait de pouvoir utiliser le mot clé « ParamArray » (« params » en C#) avec l'interface IEnumerable et pas seulement avec des tableaux.
Voici une proposition d'implémentation en VB 13 :

```
Sub  f( ParamArray  x  As  IEnumerable ( Of  String ))
```

Déclaration implicite des paramètres utilisant le mot-clé « out »

Comme pour C# 6, les variables locales peuvent être déclarées implicitement lors de l'utilisation du mot-clé « out ».
Voici une proposition d'implémentation en VB 13 :

```
Function  TryParse(s  As  String ,  Out  x  As  Integer )  As  Boolean
If  Integer .TryParse(s,  Out  x)  Then  ...
```

Visual Basic permettrait aussi la déclaration implicite de variables.
Voici une proposition d'implémentation en VB 13 :

```
If (Dim  x = GetValue()) > 0  Then  Console .WriteLine(x)
```

Propagation et absence de valeur (Safe Navigation Operator)

Comme déjà évoqué, il convient souvent d'analyser et de vérifier les données d'entrée avant de déclencher le traitement. Comme dans C# 6, la nouvelle syntaxe « ?. » simplifie et automatise la vérification de la nullité des données d'entrée. Il pourrait aussi exister le mot clé « ?(). ». Plus besoin d'écrire une série de vérifications avant d'invoquer une méthode ou de lire une propriété, car il y aurait un opérateur de non propagation.
Voici une proposition d'implémentation en VB 13 :

```
Dim  y  As  Integer  = x?.y?(3)?.z
Dim  y =  If (x.y(3).z, fallback)
```

Commentaires à la suite d'une continuation de ligne implicite

Actuellement, dans Visual Basic, il n'est pas possible de faire des commentaires qui s'imbriquent correctement avec la continuation de ligne implicite. Il s'agit de pouvoir autoriser un commentaire à la suite de chaque ligne implicite. Voici une proposition d'implémentation en VB 13 :

```
Dim  value =  From  i  In  list       ' itérer dans la liste
   Let  value = Lookup(i)    ' appel à la méthode Lookup()
   Select  i, value
```

Interfaces partielles et modules

Comme les classes partielles, on pourrait désormais écrire des interfaces partielles qui peuvent être définies à plusieurs endroits ou plusieurs fichiers. Ceci serait très utile pour rajouter de nouvelles méthodes ou propriétés. Cette fonctionnalité est surtout dévolue aux utilisateurs usant fortement de la génération de code.
Voici une proposition d'implémentation en VB 13 :

```
Partial  Interface  I
     Sub  Example()
End  Interface
```

Inférence de type dans le constructeur

Il est pénible d'avoir à fournir les types d'arguments de façon explicite lorsque l'on appelle le constructeur d'une classe. L'approche consisterait par exemple lorsque l'on instancie une nouvelle liste (« List ») à inférer le type afin de trouver le bon constructeur à appeler.

```
Dim  x =  New  List ({1, 2, 3})    ' devrait inférer New List
(Of Integer)({1,2,3})
```

TypeOf IsNot

Dans sa version courante, Visual Basic ne permet pas d'écrire des tests sur le type d'un objet de façon très élégante, car l'opérateur « IsNot » n'existe pas sur le type (« TypeOf »). La syntaxe suivante serait bien plus lisible.

```
If  Not  TypeOf  sender  Is  Button  Then  ...   ' peu explicite
If  TypeOf  sender  IsNot  Button  Then  ...   ' plus lisible
```

Interpolation de chaînes de caractères

A l'heure d'aujourd'hui, quand on utilise String.Format avec beaucoup d'arguments, il arrive de se tromper entre les occurrences {0}, {1}, {2}, ... Ce qui génère des erreurs parfois difficiles à détecter et des lignes de code difficiles à comprendre. La demande d'évolution consiste à proposer l'interpolation de chaînes de caractères, ce qui est déjà répandu dans de nombreux langages modernes. L'exemple ci-dessous montre le préfixe $ devant une chaîne interpolée et une expression entre accolades, ainsi que des spécifications de format. L'exemple montre également une idée de sémantique où l'on utilise l'InvariantCulture dans le String.Format tout en laissant le compilateur optimiser.

Les formats des dates

Les formats de dates et heures sont actuellement au format américain (US). Visual Basic devrait pouvoir supporter l'ensemble des formats ISO.
Voici une proposition d'implémentation en VB 13 :

```
Dim  us_format  = #2/10/2014 10:30:43#
Dim  iso_format = #2014-02-10 10:30:43#
```

Les formats binaires

L'utilisation des formats binaires serait désormais possible. Ils vont probablement utiliser le préfixe &B et ils seront applicables pour les énumérations du type bit-flag. Voici une proposition d'implémentation en VB 13 :

```
Dim  code = &B010101
```

Séparateur de digit groups

En écrivant des littéraux binaires de 32-bits, il arrive très souvent de faire des erreurs de saisie. L'évolution consisterait à autoriser un séparateur de digit groups (underscore, espace, tiret...).

Conclusion

Vous connaissez maintenant les différentes propositions d'évolution. Comme vous pouvez le constater, ces évolutions tendent à faire de C# un langage encore plus simple, plus fonctionnel et plus synthétique : le développeur s'attache de plus en plus aux fonctionnalités à développer et de moins en moins à la syntaxe qui, au final, n'apporte aucune valeur ajoutée. Certaines évolutions sont très intéressantes et vont beaucoup nous faciliter la vie. Comme à l'habitude, vous pouvez vous exprimer sur ces nouvelles fonctionnalités en votant sur le UserVoice (http://visualstudio.uservoice.com). N'hésitez pas à donner votre avis sur les évolutions que nous venons de présenter et à proposer de nouvelles idées. En cas de bug sur l'existant, vous pouvez les signaler sur Connect afin qu'ils soient résolus par Microsoft (https://connect.microsoft.com).

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

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

Meta Programmation : l'avenir du développeur ?

Qu'est-ce que la méta-programmation ? Il s'agit d'une vision de la programmation orientée « code ». En effet, lorsque nous programmons, dans la majorité des cas, nous cherchons à mettre en valeur des données (fichiers, bases de données, appareils de mesures, etc..).

La méta-programmation va nous permettre d'exploiter du code comme source de données, au même titre qu'un programme classique exploiterait une base de données. Nous allons donc tenter d'appréhender par la pratique ce vaste sujet, notamment à travers quelques situations que nous décrirons en C# et avec les outils de l'écosystème .NET. Le point de vue de cet article est résolument orienté productivité et outillage, mais la méta-programmation trouve son intérêt dans toutes sortes de scénarios.
Attention : code complet sur www.programmez.com

LA REFLECTION

Tout développeur pratiquant la *reflection* est un développeur ayant déjà fait de la méta-programmation. En effet, elle consiste à inspecter dynamiquement le contenu des assemblies. Tout le concept de *reflection* est basé sur une classe du framework .NET, la classe « Type ». Cette classe n'est ni plus ni moins que la description d'un type .NET. Dans la plupart des cas, le point d'entrée pour utiliser la *reflection* est la méthode « *GetType* » (disponible dans tout objet .NET) ou le mot clé « *typeof* ». Ces deux outils vont nous permettre d'extraire la description d'un type courant. Récupération et comparaison de deux objets de type « *String* ».

```
string meta = «programmation»;
Type typeOfString = typeof(string); // Description du type string
Type metaGetType = meta.GetType(); // Description du type de
la variable meta
Assert(typeOfString == metaGetType); // Les deux méthodes renvoient
Assert(ReferenceEquals(typeOfString, metaGetType)); // le
même objet
```

La *reflection* n'est pas seulement utile dans le cas d'une comparaison entre deux types. Comme dit plus haut, un objet « *Type* » étant la description d'un type .NET, nous allons pouvoir extraire un lot d'informations sur celui-ci tels que, ses méthodes et propriétés triées par critères de visibilité, ses attributs ou ses types de bases. On peut par exemple récupérer les méthodes (publiques ou non) de la classe String, ou récupérer la signature d'une méthode à partir de la classe MethodInfo (cf zip - Reflection samples.txt) .
Comme vu dans l'exemple précèdent, chaque objet « *Type* » donne accès à la liste des membres de la classe correspondante, sous la forme d'objet de type « *MemberInfo* », dérivés en « *EventInfo* », « *MethodInfo* », « *FieldInfo* » ou « *PropertyInfo* ». Ces classes exposent de plus des méthodes permettant d'accéder à la valeur correspondante du membre, pour un objet donné. Par exemple, accéder à une propriété se ferait ainsi (considérant la classe Person, ayant une propriété Name de type string) :

```
var person = new Person { Name = «Doe» };
var personType = typeof(Person);
var namePropertyInfo = personType.GetProperty(«Name»);
var value = (string)namePropertyInfo.GetValue(person);
Assert(value == «Doe»);
```

Enfin, il vous est possible à travers le *namespace* « *System.Reflection.Emit* » de définir de nouveaux types durant l'exécution du programme. Ce procédé est par exemple employé par *Entity Framework,* pour définir des *proxys* sur les entités de nos modèles. Plus précisément, si cela est autorisé par les configurations, lorsque l'ORM nous retourne des objets correspondant à nos entités, il utilise une classe dérivée de celle-ci qu'il aura créée à la volée, via « *Emit* ». Cela lui permet d'injecter du code MSIL dans les propriétés de la classe, ou même d'overrider des membres virtuels (pour les propriétés de navigation par exemple). Cela explique le fait qu'un code tel que « *myEntity.GetType()* ».Name retourne une chaîne comme « MyEntity_A30232F8-4A98-4461-865A-75C5D90EDFB9 », puisqu'il s'agit d'un nom généré dynamiquement et étant prévu pour être unique. Utiliser la *reflection* peut s'avérer bien pratique, mais il ne faut pas en abuser. En effet, utiliser massivement la *reflection* implique souvent un problème de conception, et cela aura également un coût sur les performances dont il faut avoir conscience.

T4

L'usage de la *reflection* est bien souvent le premier pas vers la méta-programmation. Elle est simple et facile d'accès pour peu que le langage cible offre une API valable. Néanmoins, si votre but est de générer du code (en analysant une portion d'un autre code par exemple), Visual Studio offre un outil intégré appelé « *template T4* » (prononcé généralement à l'anglaise). Il s'agit d'un fichier de code balisé utilisant C# (ou VB) (avec quelques syntaxes spécifiques que nous verrons) permettant de produire d'autres fichiers textes (C#, xaml, html ou n'importe quel autre type).
Un des intérêts de T4 vient du fait que le template est directement incorporé à la solution (sous la forme d'un fichier .tt), pouvant être régénéré à tout moment, à chaque compilation par exemple. Beaucoup de développeurs connaissent déjà cette technologie car elle est employée pour générer des classes à partir d'un model *Entity Framework* EDMX. Ici donc, le but recherché est de produire du code qui pourrait être vu comme « rébarbatif » ou source d'erreurs. Par défaut, sous Visual Studio 2013, lorsque nous créons un T4 (text template), voici ce que nous obtenons :

```
<#@ template debug=»false» hostspecific=»false» language=»C#» #>
<#@ assembly name=»System.Core» #>
<#@ import namespace=»System.Linq» #>
<#@ import namespace=»System.Text» #>
<#@ import namespace=»System.Collections.Generic» #>
<#@ output extension=».txt» #>
```

La syntaxe `<#@ #>` permet de définir des directives.
À la première ligne, les propriétés générales du template sont définies comme le langage utilisé pour l'écrire (C# ou VB).
L'assembly « *System.Core* » est ensuite chargée. La directive « `import` » permet de réaliser l'équivalent du mot clé « `using` ».
Enfin, « `output` » permet de spécifier l'extension du fichier à créer. Nous allons donc le passer en .cs pour générer un fichier de code C# : `<#@ output extension=».cs» #>`. Le principe de fonctionnement d'un template T4 est simple, tout caractère écrit en dehors des balises sera rendu tel quel dans notre fichier généré. Le « Main » du méta-code est quant à lui défini à l'intérieur des balises `<# #>`. Nous pouvons donc commencer par un commentaire prévenant d'éventuels lecteurs du code

qu'il s'agit d'un fichier généré automatiquement et que de ce fait il ne faut pas le modifier car il pourrait être écrasé par la suite :

```
// Ce fichier est généré automatiquement
// Date de dernière génération : <#= DateTime.Now #>
// Ne pas éditer !
```

À noter l'usage de la syntaxe <#= #>. Cette dernière nous permet d'insérer le résultat d'une expression dans le fichier généré. Pour cet exemple, nous pourrions créer un template générant un *proxy* pour une classe implémentant une interface donnée.
Tout d'abord, nous allons créer à la fin de notre Template une méthode que nous pourrons réutiliser. Pour cela, il faut placer notre méthode entre des balises <#+ #>. Le template n'est prévu pour produire qu'un seul fichier. Il faut ruser un peu si l'on en veut plus, et utiliser la library EnvDTE (cf zip - SaveInNewFile.txt). Pour utiliser EnvDTE, il faut le rajouter aux assembly / import du template T4 :

```
<#@ assembly name=»EnvDTE» #>
<#@ import namespace=»EnvDTE» #>
<#@ import namespace=»System.IO» #>
```

Et de remplacer la première ligne par :

```
<#@ template debug=»false» hostspecific=»true» language=»C#» #>
```

Ensuite, nous allons définir une nouvelle méthode permettant de construire le « proxy », en incorporant de la même façon nos helpers de la première partie, « *GetSignatureFromMethodInfo* » et « *ParameterAsText* » :

```
<#+
void GenerateWrapper(Type interfaceToWrap, string className,
 string classNamespace)
{
 // ici le code de génération de la classe portant le nom de className,
 // dans le namespace classNamespace implémentant interfaceToWrap
}
string GetSignatureFromMethodInfo(MethodInfo methodInfo)
{
 // même chose que dans la première partie
 // [...]
}
string ParameterAsText(ParameterInfo p)
{
 // même chose que dans la première partie
 // [...]
}
#>
```

Procédons par étape en complétant la méthode « *GenerateWrapper* ». Commençons par vérifier que la variable « *interfaceToWrap* » est bien un type correspondant à une interface (car dans le cas d'une classe, le travail à réaliser est un peu plus complexe) :

```
if (!interfaceToWrap.IsInterface) throw new Exception(«C'est un
peu plus compliqué pour une classe»);
```

Récupérons ensuite la liste des méthodes et propriétés de l'interface (en laissant de côté les évènements pour plus de simplicité).

```
var properties = interfaceToWrap.GetProperties();
// il faut filtrer pour éviter de récupérer les getters et
setters des propriétés
var methods = interfaceToWrap.GetMethods().Where(m => !m.Is
Constructor && !m.IsSpecialName);
```

Pour la suite, nous aurions besoin de générer les « `using` », et ce dès le début du template. Sauf qu'à ce stade de l'exécution du template, nous ne connaissons pas les noms des classes que nous aurons à résoudre. Une solution consiste à stocker en mémoire ces résolutions, et de les ajouter au template à la fin (nous verrons comment le moment venu).

```
var usings = new HashSet<string>();
if (interfaceToWrap.Namespace != null)
 usings.Add(interfaceToWrap.Namespace);
```

Nous continuerons en « écrivant » notre wrapper (ajouter le code qui suit à la suite, dans la définition de la méthode, avant l'accolade fermante) :

```
#>

namespace <#= classNamespace #>
{
 public class <#= className #> : <#= interfaceToWrap.Name #>
 {
   private readonly <#= interfaceToWrap.Name #> _wrap;
   public <#= className #>(<#= interfaceToWrap.Name #> wrap)
   {
     _wrap = wrap;
   }
/** La suite ici **/
 }
<#+
```

Ici nous utilisons les variables préalablement définies pour insérer du contenu dynamique dans le fichier avec les balises <#= #>. On remarque que le passage du code du template au code à insérer dans le fichier se fait en fermant la précédente balise <#+ avec #> et que l'on repasse au code du template avec un nouveau <#+. Nous allons ensuite itérer sur les propriétés de l'interface afin de produire une implémentation. C'est ici par exemple que nous pourrons injecter du code spécifique pour notre wrapper.

```
<#+
 foreach (var propertyInfo in properties)
 {
   if (propertyInfo.PropertyType.Namespace != null)
     usings.Add(propertyInfo.PropertyType.Namespace);
#>
   public <#= propertyInfo.PropertyType.Name #> <#= property
Info.Name #>
   {
<#+
   if (propertyInfo.CanRead)
   {
#>
     get { return _wrap.<#= propertyInfo.Name #>; }
<#+
   }
   if (propertyInfo.CanWrite)
   {
#>
     set { _wrap.<#= propertyInfo.Name #> = value; }
```

```
<#+
    }
#>
    }
<#+
 }
```

Et nous pouvons faire de même ensuite pour l'implémentation des méthodes (où nous réutiliserons nos deux helpers pour générer les signatures) :

```
 foreach (var methodInfo in methods)
 {
   if (methodInfo.ReturnType.Namespace != null)
     usings.Add(methodInfo.ReturnType.Namespace);
   foreach (var parameter in methodInfo.GetParameters())
     if (parameter.ParameterType.Namespace != null)
       usings.Add(parameter.ParameterType.Namespace);
#>
   <#= GetSignatureFromMethodInfo(methodInfo) #>
   {
     <#= methodInfo.ReturnType != typeof(void) ? «return « :
 «» #>_wrap.<#= methodInfo.Name #>(<#= string.Join(«, «, method
Info.GetParameters().Select(p => p.Name)) #>);
   }
<#+
 }
```

Ceci nous permettra de clôturer la classe et le *namespace* généré :

```
#>
 }
}
<#+
```

On remarque tout au long du code l'ajout des *namespace* des classes que nous utilisons dans la collection « *usings* ». Il est donc temps, maintenant que le code de la classe est généré, de les rajouter au début du fichier, comme suit (juste avant la fermeture de la méthode) :

```
foreach (var @using in usings)
 GenerationEnvironment.Insert(0, String.Format(«using {0};{1}»,
 @using, Environment.NewLine));
 SaveInNewFile(className + «.cs»);
```

La dernière ligne appelle la méthode nous permettant de sauvegarder l'état du « *writer* » du template dans un nouveau fichier.
Pour terminer, il suffit d'appeler notre méthode dans le template (avant la définition des méthodes, juste après le commentaire par exemple) :

```
<# GenerateWrapper(typeof(ISampleToWrap), «SampleWrapper1»,
«Articles.MetaProgrammation»); #>
<# GenerateWrapper(typeof(ISampleToWrap), «SampleWrapper2»,
«Articles.MetaProgrammation»); #>
```

Ici, le point intéressant est l'utilisation de l'interface « *ISampleToWrap* », définie dans un autre projet de la solution que celui dans lequel se trouve le template T4. Pour réaliser cela, il faut, d'une part avoir compilé le projet en question, et l'ajouter via une directive comme suit :

```
<#@ assembly name=»$(SolutionDir)Articles.MetaProgrammation.
Models\bin\Debug\Articles.MetaProgrammation.Models.dll» #>
```

Ici le `$(SolutionDir)` fait bien entendu référence au chemin complet du dossier dans lequel se trouve la solution.

ROSLYN ET NUGET

Découverte de Roslyn.

Jusqu'ici, nous nous sommes contentés d'analyser des types créés en mémoire via *reflection*.
Comme vu dans les exemples précédents, cela nécessite de compiler l'assembly dans laquelle se trouve le type cible. Avec *Roslyn*, Microsoft nous offre un outil nous permettant d'aller encore plus loin à travers une API exposant les services de compilation du framework .NET.
Plus concrètement, cela signifie qu'il nous est possible d'analyser du code, sans avoir à le compiler.
Nous intervenons directement à la place du compilateur, et pouvons donc analyser notre code directement sous sa forme textuelle. *Roslyn* va donc nous permettre de reconstruire l'arbre syntaxique de notre code textuel, afin de l'analyser. Cette analyse de l'arbre syntaxique pourra être complétée par l'analyse sémantique que *Roslyn* permet également.
Roslyn est installable à partir du package *NuGet* « *Roslyn* ».
Nous allons ici, réaliser un petit outil permettant de générer une couche domaine très simple. Cet outil sera disponible sous la forme d'un package *NuGet* permettant aux développeurs de générer les classes nécessaires à leur couche domaine via une commande *Powershell* (à exécuter dans la console *NuGet*)
L'analyse de l'arbre syntaxique se fait à travers un Visitor. *Roslyn* propose deux classes de base pour cela. La classe « *SyntaxVisitor* » et la classe « *SyntaxRewriter* ».
Le « *SyntaxVisitor* » ne sert qu'à analyser un arbre syntaxique alors que la classe « *SyntaxRewriter* » permet de le modifier. Il faut savoir que cette classe vous fournit la possibilité, lorsqu'elle est héritée, de surcharger des méthodes Visit spécifiques à l'ensemble des différents types de nœuds de l'arbre. Dans notre cas, nous nous intéresserons aux méthodes « *VisitXXXDeclaration* » ou « *VisitXXX* » où « XXX » est le nom de l'élément à visiter.
Cet élément peut être, entre-autres, *Property* ou *Method*. Cet arbre syntaxique va vous permettre de générer en sortie du code qui pourra être, soit la copie conforme du code analysé en entrée, soit une version de ce code modifié en ayant intercepté son parcours grâce aux méthodes « *VisitXXX* »
Mise en œuvre d'un « *SyntaxRewriter* » simple :

```
internal class SimpleRewriter : SyntaxRewriter
{
 public override SyntaxNode VisitMethodDeclaration(Method
DeclarationSyntax node)
 {
 return null;
 }
}
```

Nous pouvons voir ici qu'il nous suffit de surcharger la méthode « *VisitMethodDeclaration* » afin d'intervenir à chaque fois qu'une méthode est détectée dans notre code. Ici, notre rewriter nous permet de « supprimer » toutes les méthodes de notre code dans l'arbre syntaxique en renvoyant « *null* ». (Utile dans le cas où nous voudrions réaliser des entités de transfert de couche sans aucune logique à partir d'une couche métier). Il vous sera parfois nécessaire d'analyser le code plus en détails. Afin de pouvoir réaliser cela, nous allons utiliser le modèle sémantique de notre arbre syntaxique, qui vous fournira par exemple l'ensemble des symboles d'une déclaration de propriété.

```
public static void VisitFiles()
{
  ISolution solution = Solution.Load(«Chemin/Du/Fichier/.sln»);
  var projet = solution.Projects.First(p => p.Name == «MonProjet»);
  //On récupère tous les fichiers «Regular» (.cs ou .vb)
  var sourceFiles = projet.Documents.Where(d => d.SourceCode
Kind == Roslyn.Compilers.SourceCodeKind.Regular);

  //Pour chaque fichier source
  foreach (var document in sourceFiles)
  {
    //On récupère le type de compilation du projet
    CommonCompilation compilation = projet.GetCompilation();
    //A partir du type de compilation on récupère le model
  sémantique du code source
    ISemanticModel model = compilation.GetSemanticModel(document.
GetSyntaxTree());
    //On passe le model sémantique au rewriter dédié au fichier
  source courant
    DomainRewriter rewriter = new DomainRewriter(model);
    //On lance l'analyse syntaxique de notre fichier source
    var syntaxTree = rewriter.Visit(document.GetSyntaxRoot() as
  CompilationUnitSyntax);
    //On récupère sous forme de string le code résultant
    string code = syntaxTree.NormalizeWhitespace().ToFullString();
  }
}
```

Voyons désormais comment utiliser notre « *ISemanticModel* » au sein de notre Rewriter.

```
internal class DomainRewriter : SyntaxRewriter
{
  private readonly ISemanticModel model;
  public DomainRewriter(ISemanticModel model)
  {
    this.model = model;
  }

  public override SyntaxNode VisitPropertyDeclaration(Property
DeclarationSyntax node)
  {
    var symbol = model.GetDeclaredSymbol(node) as PropertySymbol;

    if (symbol.Type.TypeKind == TypeKind.Interface)
      return null;
    else
      return node;
  }
}
```

Nous nous servons ici de la méthode « *GetDeclaredSymbol* » de notre « *ISemanticModel* » afin d'extraire des informations sur la propriété en train d'être analysée. Il suffit ensuite de tester le type de notre propriété afin d'ignorer toute propriété dont le type est une interface.

Roslyn nous offre également d'autres services comme l'accès à des API de Visual Studio tel que le Refactoring par exemple.
Vous l'aurez compris, le framework *Roslyn* est très puissant dans de nombreux cas d'utilisations (pure analyse de code, génération ou modification de code). Cela donnera lieu à un prochain article où nous aborderons ce sujet de façon plus approfondie.

Création de notre package NuGet.

Maintenant que nous avons vu *Roslyn*, nous allons voir comment packager cet outil afin de permettre aux développeurs de générer leur couche domaine très simplement via un package *NuGet*.
Nous allons donc encapsuler notre code *Roslyn* dans une application console, qui sera elle-même appelée par une commande *Powershell* définie dans notre package *NuGet*.
Afin de réaliser ce package *NuGet*, plusieurs choses sont à faire. Tout d'abord il faut installer « *NuGet* Package Explorer » (http://npe.codeplex.com/). Créez un nouveau package (File > New). Passez en mode édition et changez l'id du package par « *DomainLayerGenerator* », vous pouvez aussi changer la version si vous le souhaitez et toutes les propriétés de votre package.
Sauvegardez votre package à l'endroit souhaité (et notez bien le chemin de ce dossier). Allons maintenant configurer Visual Studio afin d'utiliser les packages locaux.
Dans Visual Studio, allez dans « *Outils* > *Options* > *Package Manager* > *Package Sources* », il vous suffit ensuite de rajouter une entrée pointant vers le chemin du dossier précédemment noté afin que votre console *NuGet* installe les packages à partir de cette source supplémentaire.
Vous pouvez désormais installer votre package en tapant dans la console *NuGet* : « *Install-Package DomainLayerGenerator* ».
Nous allons maintenant améliorer notre package en y ajoutant les fonctionnalités précédemment développées avec *Roslyn*.
Créons une application console et référençons l'assembly contenant notre « *DomainRewriter* ».
Modifions un peu le code de notre méthode « *VisitFiles* » et renommons-la « *GenerateDomainLayer* » et définissons-y une méthode GenerateDomainLayer (cf zip - GenerateDomainLayer.txt).
Il nous suffit maintenant d'appeler la méthode « *GenerateDomainLayer* » dans notre classe « *Program* » qui est le point d'entrée de notre application console (cf zip - GenerateDomainLayerMain.txt). Comme vous pouvez le voir, notre application console utilise 5 paramètres :

- Le nom de la commande (si nous souhaitons que l'application console puisse réaliser plusieurs opérations),
- Le chemin de la solution contenant les projets source et cible,
- Le nom du projet source,
- Le nom du projet cible,
- Le chemin du fichier qui contiendra le chemin de chaque fichier généré.

Comme vous pouvez le voir, nous passons en dernier paramètre un chemin vers un fichier tampon. En effet, il est possible avec *Roslyn* de modifier un projet afin d'y ajouter des éléments. Mais cette manière de procéder n'est pas pratique, due au fait que Visual Studio détecte ce changement comme étant un changement externe au projet et vous propose de recharger complètement ledit projet à chaque ajout.
Nous allons donc utiliser la commande *Powershell* « *AddFromFiles* » fournie par *NuGet* afin d'ajouter chaque fichier au projet cible de façon totalement transparente.
Pour ce faire, nous allons ajouter un dossier « *tools* » à notre package *NuGet* (clique droit dans la partie droite de « *NuGet Package Explorer* », puis « *Add Tools Folder* »).
Ajoutez-y les fichiers « init.ps1 install.ps1 et uninstall.ps1 » (clic droit sur « *Tools* » puis « *Add init.ps1* »). Nous allons modifier quelque peu le fichier « *init.ps1* » afin d'y ajouter notre commande « *New-DomainLayer* ». Quant aux fichiers « *install* » et « *uninstall* », ils sont exécutés lors de l'installation et la désinstallation de notre package. Nous n'aurons pas à les modifier.

```
 param($installPath, $toolsPath, $package)

function New-DomainLayer($source, $target)
{
 #Récupération du chemin de la solution courante.
 $solutionPath = $DTE.Solution.FullName;
 #Création du chemin complet vers notre fichier tampon
 $outputFilePath = (Join-Path $toolsPath GenerateDomain
Layer_FilesToAdd.txt);
 #Création du chemin complet vers notre application console
 $exePath = Join-Path $toolsPath DomainLayerGenerator.exe
 #Création des paramètres à passer à notre application console.
 $exeArgs = @('«GenerateDomainLayer»', '«' + $solutionPath
+ '«', '«' + $source + '«', '«' + $target + '«', '«' + $out
putFilePath + '«');
 #Execution de l'application console et attente jusqu'à son
 exécution complète.
 start-process -filepath $exePath -ArgumentList $exeArgs -Wait

 #On récupère le projet cible
 $targetProject = $DTE.Solution.Projects | where { $_.Project
Name -eq $target }
 #Pour chaque chemin dans le fichier tampon, on l'ajoute au
 projet cible
 Get-Content $outputFilePath | foreach{ $targetProject.Project
Items.AddFromFile($_); }
}

#On active l'autocomplétion pour la commande 'New-DomainLayer'
Register-TabExpansion 'New-DomainLayer' @{
#Pour les paramètres source et target, on récupère le nom de
 chaque projet de la solution
#afin de l'afficher en option au développeur
'source' = { $DTE.Solution.Projects | foreach { $_.Project
Name } };
'target' = { $DTE.Solution.Projects | foreach { $_.Project
Name } };
}
```

Il ne nous reste plus qu'à ajouter au dossier « *Tools* » de notre package :

- Le fichier « *GenerateDomainLayer_FilesToAdd.txt* » qui fait office de fichier tampon,
- Notre application console « *DomainLayerGenerator.exe* »,
- Les DLLs nécessaires à l'exécution de notre application console.

Il ne nous reste plus qu'à installer notre package : taper la commande *NuGet* « *New-DomainLayer* » en se laissant guider par l'auto complétion afin de générer notre couche domaine.

CONCLUSION

La meta-programmation comprend plusieurs sous-domaines.
Dans le cadre de cet article, nous avons commencé par voir comment manipuler du code à travers la *reflection,* et nous avons très brièvement expliqué qu'il était également possible d'analyser notre code avec *Roslyn*. Dans le même registre, on trouve aussi d'autres technologies telles que les Expressions LINQ par exemple.
Avec la meta-programmation, il est également possible de remplacer le process de compilation avec *Roslyn,* ou de procéder à une modification de l'IL en post compilation avec Mono.Cecil par exemple.
Cependant, si cette approche est utilisée avec l'AOP,
elle n'est pas sans risque.
Enfin la meta-programmation, c'est aussi la génération de code. Celle-ci peut être faite de manière dynamique à travers des technologies comme *T4*, *Roslyn*, les *Expressions LINQ*, *CodeDOM*, *Reflection.Emit*, etc. ou de manière statique avec T4 ou avec *NuGet* par exemple.
En somme, la méta-programmation est l'outil par excellence pour tout développeur souhaitant passer à la vitesse supérieure : générer du code plus efficacement, plus rapidement, sans erreurs. Les possibilités en termes d'outillage sont infinies, pour peu que des API comme *Roslyn* soient accessibles. Elle permet de s'affranchir des contraintes d'un langage, afin de rendre le code générique et/ou portable (puisqu'il suffit de produire une version spécifique du code selon le contexte). Enfin, il est à noter que la méta-programmation est un sujet très vaste, et que cet article est trop court pour tout aborder. Cependant, nous aurons l'occasion d'y revenir dans un futur proche.

Pierre-Alexandre *Gury*, Thomas **Ouvre**
Infinite Square

INFINITE SQUARE

Que faut-il attendre du futur de Java ?

Il est déjà temps de se pencher sur le futur de Java, et notamment la prochaine version Java 9. Attendue pour début 2016, cette future mouture doit poursuivre le travail de modernisation de la plateforme entamé par Oracle pour répondre aux défis des applications d'entreprise modernes, et rassurer sur l'avenir du langage que certains polémistes qualifient déjà de nouveau Cobol.
Tour d'horizon et prospectives sur le futur de Java.

Avec la sortie de Java SE 8, Oracle met enfin à portée de main des développeurs Java la programmation fonctionnelle grâce aux Lambdas. Cette fonctionnalité tant attendue promet déjà de modifier profondément le quotidien des développeurs de la plateforme Java en apportant de nouveaux paradigmes pour la réalisation d'applications d'entreprise. Cette évolution s'inscrit dans la volonté d'Oracle de garder la plateforme au coeur des entreprises et d'éviter qu'elle ne devienne le futur Cobol.

Le futur de Java va donc s'écrire en privilégiant les technologies d'avenir qui sont d'ores et déjà au coeur des préoccupations en entreprise aujourd'hui. Les axes forts de Java SE 9 seront donc le Cloud, domaine où Oracle doit imposer Java comme la solution idoine, le mobile, où Google avec sa plateforme Android a déjà pris un avantage quasiment décisif, et la modernisation du JDK et de la JVM, pour répondre aux enjeux de parallélisation des traitements et de performances sous contraintes.

Premier gros chantier d'Oracle pour Java SE 9, la modularisation du JDK vise à moderniser le bloc monolithique de plus de 4000 classes qui le composent, afin de le rendre plus souple et plus léger. En effet, quel est l'intérêt pour une simple application client lourd d'utiliser un JDK alourdi des classes liées à JMX ou à la gestion LDAP ? La prise en compte de ce problème ne date pas d'hier puisque dès 2005, des premières ébauches de solution on été amorcées pour une mise à disposition au sein de Java 7 ! Dix ans plus tard, la JSR 277 et ses fichiers jam n'a pas résisté à la vive opposition des développeurs, et les superpackages de la JSR 294 initiés en 2007 ont, eux, été remplacés par le projet Jigsaw initialement prévu pour Java SE 8.

Projet Jigsaw

Devant l'ampleur de la tâche, le projet Jigsaw a été reporté pour une prise en compte dans Java SE 9. Jigsaw est un système de module pour les applications Java et la plateforme en général. Appliqué au JDK, Jigsaw permet un découpage en plusieurs petits blocs, ce qui augmentera nettement les performances au démarrage des applications, puisqu'il sera possible de définir des configurations adaptées au contexte d'exécution (serveur d'applications, desktop ou smartphone). Au niveau applicatif, il facilitera la construction, la maintenance et la distribution d'applications ayant de plus en plus de dépendances. Enfin, il doit résoudre le fameux problème du JAR hell qui se produit lorsque différentes version, d'une même classe se retrouvent au sein du classpath d'un projet.

Techniquement, Jigsaw se traduit par la présence de fichiers modules contenant les dépendances d'une application. Si le prototype de Jigsaw déjà réalisé permet le téléchargement des dépendances, Mark Reinhold, architecte en chef de Java SE, ne s'interdit pas de repartir de zéro sur certains points du projet dont celui-ci. En effet, cette fonctionnalité ferait doublon avec ce que proposent des outils de build comme Maven ou Gradle. De fait, du travail est à prévoir pour définir réellement le périmètre de Jigsaw et l'intégrer au mieux au JDK, ainsi qu'aux solutions de build qui font référence. Petite lueur d'espoir dans ce méandre d'incertitudes, l'introduction des Profiles dans Java SE 8 devrait faciliter le travail de modularisation pour Java SE 9.

Cloud

Autre axe majeur de travail d'Oracle pour Java SE 9, un meilleur support du Cloud et de ses impératifs. En ce sens, il est primordial d'améliorer le partage de JVMs au sein d'un même OS tout en optimisant sa gestion mémoire, tant au niveau des threads, qu'au niveau des ressources allouées. Pour ce faire, Java SE 9 devrait apporter certaines modifications du côté de la JVM, afin qu'elle puisse optimiser son paramétrage interne, et s'adapter au mieux aux contraintes de son environnement sous-jacent. Enfin, un gros travail d'optimisation mémoire sur les structures de données de la plateforme est attendu pour améliorer la scalabilité des applications Java dans le Cloud.

Mobile

Largement distancé dans la bataille de l'embarqué, et plus encore du mobile avec Android, Oracle souhaite faire de la mobilité un sujet majeur pour la version 9 en proposant d'unifier Java SE et Java ME. L'introduction des Profiles et notamment le profil compact1 s'y prête particulièrement bien. Autre cheval de bataille dans le domaine, l'investissement d'Oracle dans sa technologie Java FX qui doit redonner ses lettres de noblesse au développement d'interfaces graphiques en Java pour desktop et pour smartphones. En ce sens, la modularisation du JDK pour accélérer les temps de chargement s'inscrit parfaitement dans cette volonté d'imposer Java FX sur ces 2 plateformes. Néanmoins, Java SE 9 devra absolument permettre une meilleure intégration entre Java FX et le monde Java.

Parallélisme

Dernier axe d'importance pour conserver le marché des applications d'entreprise, un meilleur support de la parallélisation des traitements tant au niveau applicatif qu'au niveau machine virtuelle. Ainsi, plusieurs travaux en cours sur ces thématiques pourraient faire leur apparition dans la JVM de Java SE 9. Le premier concerne la parallélisation automatique des traitements directement dans la JVM via l'emploi d'OpenCL.

Déjà incorporé au sein d'OpenJDK, le projet Sumatra doit permettre aux applications Java et à la JVM de tirer partie des GPUs. Mettre à disposition une API pour tirer pleinement partie du GPU dans Java SE 9 semble être d'actualité selon les dires de Mark Reinhold. Ce projet ambitieux est également suivi de près par les langages basés sur la plateforme Java puisque cela leur ouvrirait de facto l'accès à la puissance de calculs des GPU.

Au niveau API

On le voit clairement Java SE 9 aura un impact fort sur la JVM et la plateforme Java. Néanmoins, les APIs côté développeur ne seront pas oubliées, même si pour le moment, fort logiquement, les informations de ce côté-là restent encore à l'état embryonnaire. On peut cependant d'ores et déjà avancer que comme de coutume, l'API java.util.concurrent portée par la JSR 166 subira son petit lifting pour lequel son créateur Doug Lea vient de lancer

les premières propositions. Au niveau de l'utilisation du code natif, l'architecte de Java SE 9 assure que le nécessaire sera fait afin de simplifier l'utilisation de JNI. Le but étant in fine de pouvoir appeler du code natif en Java sans écrire de code boilerplate supplémentaire.
Si la mise à disposition de l'API Date and Time dans Java SE 8 apporte une réponse au problème de gestion des dates sur la plateforme, la problématique reste entière concernant la gestion monétaire. Placé sous l'égide de la JSR 354, l'API Currency and Money possède déjà une implémentation de référence disponible sur GitHub. Basée sur un noyau permettant la représentation de montants et devises, elle facilite les opérations monétaires, le formatage, mais également les conversions tout en étant extensible. Bref, son intégration dans Java SE 9 ne devrait être qu'une formalité et son application fera le bonheur des développeurs du monde bancaire et de la finance abonnés jusqu'alors à l'indigeste BigDecimal.
Pour terminer, un petit mot sur le langage qui ne devrait pas connaître de grands chamboulements, comme avec Java 8 et les Lambdas. Néanmoins, on parle d'une suppression des types primitifs pour passer au tout objet, ainsi que d'une vraie implémentation des generics avec également l'introduction de tableaux extra larges via le support du 64 bits. Mark Reinhold a d'ores et déjà annoncé qu'il était envisagé de limiter la compatibilité ascendante de Java SE 9 à Java 8 !

Conclusion

Oracle est confronté aux réalités du terrain et a dû revoir à plusieurs reprises sa roadmap qui planifiait une sortie majeure de Java tous les 2 ans. Ainsi, Java SE 7 n'est sorti que 5 ans après la version 6, et Java SE 8 sort cette année 3 ans après la version 7. Néanmoins, ces retards auront permis d'assurer la pérennité de la plateforme au niveau sécurité, tout en aboutissant à l'introduction de la programmation fonctionnelle. Un mal pour un bien en somme qui doit permettre à Oracle de trouver son rythme de croisière désormais avec une version majeure tous les 2 ans.

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

L'agenda 2014-2015 des langages

Comme pour les applications, les langages évoluent version après version. Nous y trouvons les versions majeures, les versions mineures et les versions de maintenance. Souvent, on les reconnaît à la numérotation : x, x.x., x.x.x. Le « core language » est l'élément qui évolue le moins vite. Il ne faut pas casser le paradigme, la syntaxe. Les cycles de développement sont différents selon le type de version.
Par exemple, Python reçoit des correctifs tous les 4-6 mois durant 18 mois. Quand une version majeure du langage sort, la branche précédente passe généralement en mode maintenance (essentiellement patch de sécurité) pour une certaine durée (variable selon le langage), avant d'être déclarée obsolète.

Python

La communauté Python est très active. Python 3.3 continue à être maintenu avec les versions 3.3.x, la dernière actuellement référencée sera la 3.3.6 (mai prochain). La branche 3.4 est disponible depuis la mi-mars. Cette version incorpore une nouvelle API C pour l'allocation mémoire, un nouveau module tracemalloc, des améliorations sur l'introspection, l'introduction d'un framework asynchrone pour les E/S.

Ruby

Depuis 2 ans, Ruby se fait assez discret. Actuellement ce langage est en version 2.1.x. Le développement de la version 2.2.0 a réellement débuté en décembre dernier. La 2.2.0 devrait être disponible en décembre 2014.

HTML

Le travail autour de HTML 5 sera entièrement terminé au 4e trimestre 2014, avec la disponibilité de la recommandation officielle. Les éditeurs et développeurs disposeront d'une référence sèche, même si nous n'avons pas attendu le W3C pour utiliser partout HTML 5. HTML 5 introduit de nombreux changements, souvent très profonds. Avec cette v5, le développement web dispose d'une nouvelle base technologique pour les prochaines années.
Le travail sur HTML 5.1 a démarré dès 2012 mais c'est véritablement en 2014 que le travail va se structurer. Le W3C annonce une disponibilité de la recommandation fin 2016, si tout va bien. Cependant, il faudra continuer à utiliser la version stable de HTML 5 tant que la 5.1 ne sera pas fiable. La 5.2 sera en gestation d'ici un an.
Une des critiques est le manque de modularité de HTML 5. Mais en réalité, HTML 5 se divise en de nombreuses spécifications séparées du cœur de HTML 5, et d'autres sont vues comme des extensions.

JavaScript

JavaScript est devenu un des langages web clés. Souvent mal aimé, de nombreux frameworks sont nés pour simplifier la vie du développeur. Actuellement, la version 1.8.5 est la dernière version officielle. L'évolution de JavaScript n'est pas simple car ses fondations dépendent d'ECMAScript. ECMAScript est une norme internationale gérée et définie par l'ECMA. ECMAScript 5.1 est la dernière version majeure normalisée (2011). JavaScript est une implémentation d'ECMAScript (ou ES). Actuellement, la prochaine version sera ECMAScript 6, alias Harmony. Harmony a pour objectifs : de faciliter le développement d'applications complexes, de mieux partager les librairies (qui se multiplient), d'améliorer l'interopérabilité et le support de notions « orientées objet », et d'améliorer les mécanismes de sécurité. Harmony a aussi l'ambition de supporter des fonctions apparues dans JavaScript. ECMAScript 6 sera totalement compatible avec ECMAScript 5. Mais ES 6 veut aller bien plus loin : en simplifier la syntaxe et en améliorer l'écriture du code. Les éditeurs suivent les évolutions du groupe de travail et intègrent plus ou moins des fonctions d'ES 6. ES 6 n'est pas attendu avant fin 2014 ou courant 2015. Tableau de compatibilité : http://kangax.github.io/es5-compat-table/es6/

Dart

Google ne fournit pas de roadmap officielle à son langage Dart. Actuellement disponible en version 1.2, Dart évolue assez rapidement : mode debug, performances, WebSocket, éditeur, etc. La 1.3 a débuté son développement. Parmi les nouveautés attendues : amélioration des assistants, correction des bugs, auto-complétion sur le DartDoc, les annotations, amélioration du support Angular.

Objective-C

Ce langage évolue très lentement. Apple le contrôle totalement. La version 2.0 est sortie en 2007. Depuis, aucune évolution importante n'a été dévoilée même avec l'explosion d'iOS. Apple étend les possibilités via les nombreux frameworks et API. A chaque évolution système et des outils de développement, le langage propose des évolutions, des corrections. Le langage se modernise de cette manière : par petite itération. En 2012, par exemple, Apple ajouta les objets littéraux.

Soyez votre propre patron !

Vous en avez assez de votre patron ? Vous voulez être indépendant ? Faire ce que vous voulez ? La solution est de devenir son propre patron, de créer sa société de développement. L'enthousiasme de départ peut rapidement s'évanouir si vous n'êtes pas un minimum préparé. Car cela ne s'improvise pas. Que ce soit une société unipersonnelle ou une SARL, vous devez impérativement comprendre le fonctionnement d'une société.

Avant la moindre démarche, vous devez savoir pourquoi vous souhaitez créer une société et être votre propre patron. Avez-vous un projet bien défini (par exemple : développer des logiciels, des apps mobiles) ou des clients récurrents (pour faire du développement) ? De là, vous pourrez définir votre business plan, le plan de financement, la trésorerie nécessaire, etc.

Vous ne savez pas faire ? Demandez à des spécialistes !

Pour vous accompagner dans vos démarches et les aides, vous pouvez contacter la CCI de votre département, notamment pour ce qui concerne les prêts (prêt d'honneur, NACRE, etc.). Pour la constitution de la société (statuts, enregistrements, publicité légale), nous vous conseillons de contacter un avocat spécialisé en droit des sociétés. Comptez environ 4 à 5 000 € pour une prestation complète. Selon la complexité du montage, ce forfait pourra augmenter. Un expert-comptable sera utile pour la tenue de la comptabilité, effectuer l'ensemble des déclarations et démarches administratives (URSSAF, Trésor public, caisse de retraites, etc.). Sauf si vous aimez la comptabilité, nous vous conseillons de passer par un cabinet comptable. Vous vous limiterez à la tenue des tableaux analytiques et à préparer les documents. Prévoyez tout de même quelques heures par semaine à l'administratif.

Soignez votre relationnel et vos (futurs) clients

Le business plan est vital. Il varie selon la nature de votre activité : éditeur de logiciels / apps, société de développement, ou les deux. Pour être viable, votre société doit générer du chiffre d'affaires et des clients. Ayez une approche prudente quand vous concevez votre dossier, notamment envers les banques. Ne pas hésiter à être très pédagogique lors de vos entretiens. Vos interlocuteurs connaissent souvent mal le monde informatique et encore moins l'univers du développement. A vous aussi de savoir comment vous faire connaître. Dans le monde des apps, la concurrence est très forte. Avec un peu de chance, une app originale ou apportant un « + » par rapport à des apps existantes peut vous aider à sortir de l'anonymat. Les contacts avec les équipes des constructeurs et éditeurs des plateformes mobiles seront précieux.

Maîtriser les dépenses

La maîtrise des dépenses et des comptes est une des clés de la réussite. Les frais généraux peuvent lourdement peser sur la trésorerie. Pensez aussi à inclure dès le départ votre salaire, même si celui-ci pourra être peu élevé durant les premiers mois d'activité, surtout, si vous ne bénéficiez pas de l'assurance chômage.

A vous de jouer !

François **Tonic**

L'avis de Christophe Peugnet, développeur

On dit qu'une entreprise commence par une idée... Oui mais en étant développeur ce ne sont pas les idées qui manquent. Actuellement, la tendance est dans l'application mobile; on peut maintenant rapidement gagner de l'argent pour peu qu'on décroche des idées d'applications qui vont toucher du coup un très large public. Il va donc falloir réfléchir quelle structure construire autour de ces revenus. Je ne vais pas orienter ce point de vue vers ce que sont business plan, domaines commerciaux et stratégies marketing, il y a tous les ans de bons articles à ce propos dans les presses spécialisées. Je vais plutôt vous parler des à-côtés.

Lorsque l'on veut se lancer dans l'aventure de l'indépendance, il faut se trouver un local, bureau propre à l'activité professionnelle, afin de bien séparer la famille de l'entreprise ! Pareil pour ce qui est de vous joindre, je pense à votre téléphone et votre email. Au début, tout est rose et merveilleux, mais quand vous aurez des centaines de clients, imaginez-vous en vacances en famille avec votre téléphone portable n'arrêtant pas de sonner, ou avec des dizaines d'emails arrivant...

Bref, il faut apprendre dès le départ à bien cloisonner son activité. Attention, il ne faut pas croire que sur une journée vous allez coder 100% de votre temps ! Divisez ça au moins par deux, car vous allez comprendre ce que c'est que le terme « paperasserie »; vous allez passer du temps à réaliser vos devis, contrats et factures sans parler de la partie administrative qui est parfois lourde et complexe. L'avis d'un expert-comptable est souvent très utile.

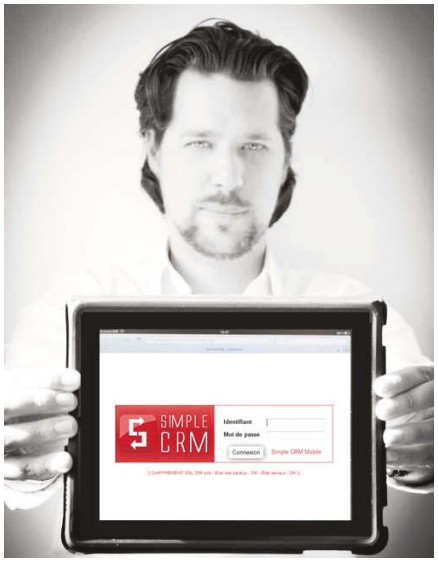
C'est un fait, on ne compte plus ses heures. En réalité, on ne les voit pas passer car on vit une passion. Et ce n'est pas forcément celle de votre famille; il va vraiment falloir apprendre à s'arrêter et s'imposer des limites horaires. Ne construisez pas votre entreprise au détriment de votre famille. Pas besoin de vous faire un dessin pour voir où je veux en venir ! Restez constamment en veille technologique, le développement est un métier superbe mais qui demande de toujours rester au courant. L´intérêt premier est bien évidement de rester en contact avec le marché, la concurrence... Surtout ne pas s'isoler!

Le temps passe tellement vite que j'en ai fait personnellement les frais lorsque j'ai vu arriver le .NET 1.0 et que j'ai seulement commencé à m'y mettre sur Windows Phone, soit dix an plus tard. Rejoindre une communauté de développement vous permettra de bien suivre ce qui se passe et de créer des relations qui sont toujours utiles.

Je pense aussi à une chose : protégez-vous et respectez les propriétés intellectuelles. Sachez que tout ce qu'il y a sur le net appartient en gros à quelqu'un. Combien je vois d'étudiants utiliser des images, logos, sons, flux de données sans faire attention aux droits. Conseil d'ami : respectez tout ça plus que jamais pour vos applications, votre futur site internet etc. Il existe de nombreux sites où certaines images/logo ne coûtent pas très cher pour être utilisés par vos soins. Un petit détail pour la fin : Faites des économies ! Les charges que vous devrez payer (comme l'URSSAF) ne vous arrivent que quelques mois après le début de votre activité et cela peut vous faire très mal si vous avez tout flambé. Dans le même registre, notez que si vous avez créé une EURL ou SARL (...), Vous ne cotisez que très peu pour votre retraite... Lorsqu'on est jeune on peut rigoler de ça, mais croyez-moi, pensez à mettre de côté dès le départ (le temps passe si vite).

Pour créer son entreprise, il faut être fou et créatif, et si c'était à refaire, et bien je le referais sans hésitation.

Christophe est gérant et développeur des logiciels SodeaSoft de la société EBLM
http://www.sodeasoft.com
Blog : http://www.peug.net
Twitter : @tossnet1



La vie d'un geek codeur, devenu CEO d'une startup

Brice Cornet est le CEO de Simple CRM, CRM français en ligne (http://crm-pour-pme.fr) qui compte des bureaux et filiales en Belgique, Roumanie, Inde, Etats-Unis et Tunisie. Se présentant de lui-même comme un pure geek, Programmez ! a voulu savoir comment cohabitent sa passion pour le code et ses responsabilités de CEO.

Quelle est votre formation ?

Je suis analyste programmeur. J'ai suivi un cursus orienté sur la programmation de gestion. Sachant depuis toujours que ce je voulais créer une entreprise de logiciel, j'ai suivi en cours du soir une formation complémentaire en gestion d'entreprise, ainsi que des cours de fiscalité et de droit commercial.

Quelle est votre technologie préférée ?

Je suis un amoureux du langage C. Je trouve que c'est le langage le plus puissant et le plus élégant jamais conçu. Mais je ne l'ai pratiqué que très peu dans ma vie professionnelle. Simple CRM est un logiciel LAMP (Linux – Apache – PHP – MySql) avec beaucoup de surcouches Jquery. Professionnellement, je dirai donc que ma préférence va au JS qui a réellement permis de bouleverser l'approche du logiciel en ligne.

Comment êtes-vous devenu CEO ?

Je l'ai tout simplement toujours été. J'ai fondé ma première entreprise alors que j'étais encore étudiant. Nous proposions en 2001 un Linux bootable sur CD-ROM qui copiait automatiquement tous les documents sur un disque dur externe, après par exemple le crash d'un Windows. Avec l'arrivée du C# en 2002, je me suis ensuite dirigé vers les applications en ligne, désirant créer une CRM dédié aux médecins et vétérinaires. Environs deux ans plus tard, j'ai abandonné ce langage pour m'orienter vers le PHP et la création d'une plateforme de travail collaboratif qui, quelques années plus tard, donna naissance à Simple CRM.

A partir de 2006, j'ai dû accepter l'évidence : si je voulais que mon entreprise évolue, je devais arrêter de coder et me concentrer sur le développement de l'entreprise. J'ai donc vécu le passage « douloureux » entre un travail orienté à 100% sur mes passions, pour un travail dont la ligne de conduite est composée du triptyque : « Rentabilité – Stabilité – Croissance ».

Sur votre blog personnel (http://www.ihaveto.be) vous publiez pas mal d'astuces orientées code ou réseau / système. Malgré votre position de CEO, vous semblez vouloir rester dans le bain. Pourquoi ?

Parce que l'on ne peut pas prétendre diriger quelque chose que l'on ne comprend pas.

Le développement de Simple CRM induit un coût de recherche et développement. Pour être à même de poser les bons choix, ceux qui permettront d'assurer la pérennité de l'entreprise, il est obligatoire de comprendre les impacts des choix techniques. Si maintenant je vais implémenter la fonctionnalité A, je sais que dans 3 ans, je serai à la fonctionnalité F. La question est donc de savoir quels choix technologiques me permettront d'effectuer un développement linéaire efficace, performant et qui respecteront nos contraintes tant fonctionnelles que budgétaires. Dès lors, si vous n'êtes pas capable d'à la fois lire du code, de gérer des performances serveurs et de comprendre vers où tendent ces technos : vous poserez les mauvais choix.

Et vous vous êtes déjà trompé ?

Oh oui ! L'an passé, nous avons jeté 9 mois de dev à la poubelle, car mon côté geek avait pris le dessus ! J'avais lancé mon équipe dans une conception très élégante techniquement mais qui ne prenait pas du tout en compte les besoins utilisateurs et la rentabilité financière. Au bout de 9 mois, j'ai réuni mon équipe et je leur ai présenté mes excuses. Je leur ai expliqué l'incohérence stratégique de ce choix. Pour dédramatiser, nous avons symboliquement gravé le code et l'avons enterré dans le jardin qui jouxte les bureaux ! Cette petite cérémonie a permis d'enterrer aussi les frustrations et les tristesses nées de cette erreur.

En pratique, comment faites-vous pour rester dans le coup ?

C'est simple mais très chronophage. D'abord, j'assiste à toutes les réunions de projet. Je suis également en copie de toutes les questions et échanges concernant les soucis de code ou de choix de dev. Chaque jour, je consacre une heure de mon temps à la lecture de flux RSS. Ces lectures concernent principalement l'actualité ICT, la maintenance R/S, la sécurité Linux, le CSS, JQuery, Ajax, PHP, ainsi que des infos relatives au design. Je survole également pas mal de code open source afin de rester à jour. Le week-end, je passe en revue les projets et les tickets de support.

Enfin, deux fois par mois, j'organise une réunion informelle avec un confrère ou un concurrent, afin d'échanger nos points de vue sur les technos, les tendances, etc.

Deux à trois fois par an, j'organise également d'autres réunions avec des designers, des spécialistes IHM (Interface Homme Machine) et également des ethnologues ou des psys, car la compréhension des utilisateurs finaux est également très importante.

Quel est l'impact de cette implication dans la vie de l'entreprise ?

Je pense que l'impact le plus évident est que mes collaborateurs savent qu'ils peuvent se tourner vers moi quand ils ont une question. Le week-end, je centralise les questions et les soucis rencontrés dans les développements en cours. Je potasse également les cahiers des charges. Ensuite, j'effectue des recherches, voire des prototypes. Le lundi matin, à la réunion de début de semaine, je viens avec des pistes de réflexion. Cela induit une dynamique positive : le fait que le « boss » soit motivé et cherche des solutions pousse toute l'équipe à se dépasser. Cela crée une ambiance très saine, sans échelon hiérarchique. Le mot d'ordre est simple : nous sommes là pour nous entraider et avancer ensemble. Cela sous-entend aussi que j'attends beaucoup de mes collaborateurs et que je refuse les fausses réponses du style : je n'ai pas eu le temps, c'est trop compliqué, etc.

Enfin, nous travaillons en démocratie. Suite à mes pistes, chacun a 15 jours pour creuser ces pistes ou en dégager de nouvelles. Ensuite, chacun présente ses choix, on échange et on finit par voter. Jusqu'à présent, je n'ai jamais imposé une idée sauf lorsque j'ai dû choisir de stopper des devs comme l'année passée.

Est-ce que vous codez encore ?

Oui mais très peu, puisque je reste en charge de l'écriture des IHM. Ceci représente donc entre un et deux projets par an, réalisés le plus souvent sous CSS3 et JQuery. J'effectue également deux fois par an un audit presque complet du code de Simple CRM, afin d'ensuite lancer des projets d'amélioration et d'optimisation des performances.

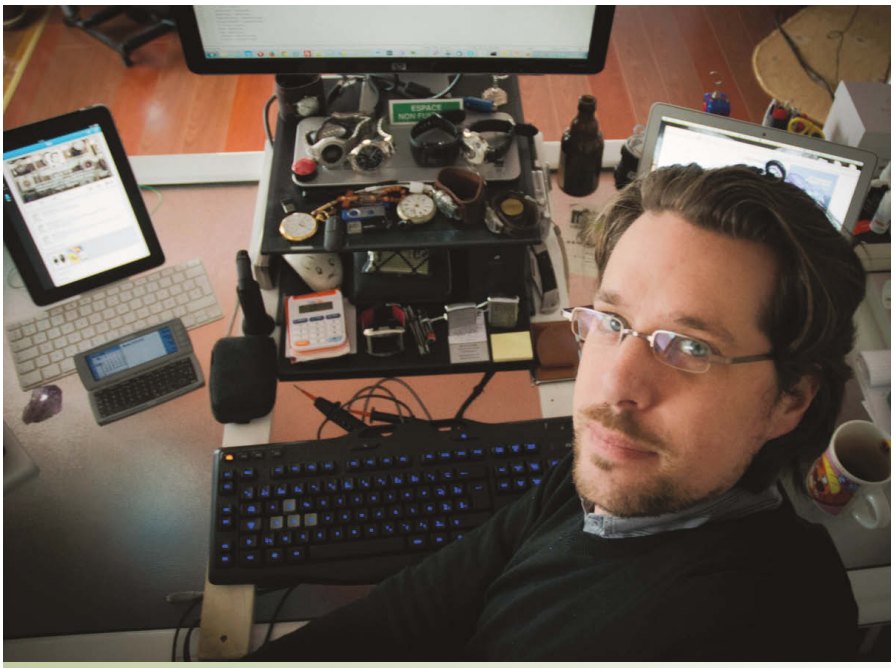
Quels conseils donner aux développeurs qui veulent se lancer ?

Le premier est de ne pas être naïf. Ce sera dur, très dur, et il faut en être conscient. Ils vont traverser de grosses galères et parfois, voire souvent, ils ne joindront pas les deux bouts financièrement parlant. Il faut être prêt mentalement et physiquement à vivre cette expérience.
Ensuite, d'enfermer leur côté geek dans une armoire, de suivre des cours de gestion d'entreprise, de finance et de marketing. Beaucoup de développeurs se plantent car ils ne savent pas comment gérer une entreprise. Il existe des cours gratuit en ligne sur https://www.coursera.org qui sont très bien faits.
Attention aussi de ne pas proposer un produit de programmeur, au lieu d'un produit de consommateur. Vous pouvez créer la meilleure techno du monde : si personne ne comprend à quoi elle sert ou si personne ne sait s'en servir : cela ne sert à rien.

Des écueils à éviter ?

Vous aurez peut-être un coup de chance au départ : sachez que la chance ne dure jamais et qu'il vous faudra tenir sur la longueur. Dès lors gérez très intelligemment vos finances, quitte à devenir radin. Petit bureau dans un grenier, frais minimum, une planche pour faire une table et du matériel de récupération suffisent à lancer une entreprise. ! Un Pentium 4 fera une très bonne machine de monitoring ou un formidable firewall. Récupérez, hackez, recyclez : ce sera autant d'argent épargné. N'oubliez pas que beaucoup de personnes qui se lancent, abandonnent leur projet de départ par obligation financière. Au lieu de développer leur projet génial, ils se retrouvent à faire des sites web bon marché pour payer les factures ; j'espère de tout cœur que cela n'arrivera pas aux personnes qui liront cette interview.

La rédaction.



L'astuce de Brice

Dopez vos performances serveurs avec S3 (et PHP) sur Amazon

Stockage des fichiers sur S3

Lorsqu'on développe un intranet documentaire, on s'aperçoit rapidement que les uploads et dowloads de fichiers monopolisent une part importante des ressources serveur.
La solution la plus fréquemment utilisée est de passer à un serveur de dimension supérieure, monopolisant au passage plusieurs heures du travail de l'administrateur.
Le service de stockage de fichiers en ligne Amazon S3 est une alternative souple au changement de serveur. L'idée est d'externaliser complètement le stockage des fichiers : le navigateur du client envoie le fichier au serveur qui, au lieu de le stocker sur un disque local, l'envoie directement sur l'espace S3 (le "bucket").
Le téléchargement se passe de la même façon : le serveur récupère le fichier et l'envoie au navigateur client sans l'avoir stocké localement.

ASTUCE : Sécuriser les accès aux fichiers

Pour sécuriser les documents, il faut restreindre l'accès aux fichiers sur S3 (via l'ACL S3 en PHP), et effectuer la correspondance entre une URL du type "download.php?fichier=fd54ed144c5d12a4b48" et le fichier stocké sur S3 via une table dans la base de données.

Pensez à l'expérience utilisateur

Utilisez une console Google Analytics dédiée pour monitorer votre serveur

Souvent le monitoring se passe uniquement côté performance serveur.
On mesure la présence en ligne, le CPU, la RAM, etc., sans se soucier du résultat final.
A mon sens, il est essentiel de prendre en compte ce pourquoi nous travaillons : la satisfaction client. Pour ce faire, une astuce simple est de créer un dashboard de monitoring serveur via Google Analytics. Ce tableau de mesure permet de visualiser la vitesse de rendu d'un logiciel en ligne, les pages les plus rapides, les plus lentes, etc. et ce côté utilisateur.
C'est à mon sens, un outil indispensable, car il permet de mieux comprendre nos clients, leurs technos et leurs habitudes d'utilisation.
Vous pouvez télécharger un tableau d'analyse Google gratuitement depuis mon blog :
http://www.ihaveto.be/2013/04/monitoring-server-via-google-analytics.html

Utiliser Node.js avec le Framework Express

Qu'est-ce que Node.js ? Node.js est une plateforme basée sur le moteur javascript V8, créée par Google pour Chrome. Celui-ci nous permet ainsi de créer différentes applications réseaux et donc des applications web. Après la création de node.js est venue celle d'Express. Express est un Framework MVC basé sur node.js que je vous présente ici. Nous allons créer une application web permettant d'afficher et d'insérer du contenu depuis une base de données de type mongodb.

Installer Express

Une fois node.js installé, il vous faudra installer tous les packages nécessaires à l'utilisation d'Express. Pour effectuer l'installation du Framework, le fonctionnement est similaire à tout package prévu pour node.js. Nous allons ainsi utiliser la commande Node Package Manager (npm) afin de télécharger l'ensemble des modules nécessaires.

Installation du Framework Express

npm install -g express

```
c:\node>
c:\node>npm install -g express
npm http GET https://registry.npmjs.org/express
npm http 200 https://registry.npmjs.org/express
npm http GET https://registry.npmjs.org/express/-/express-3.4.4.tgz
npm http 200 https://registry.npmjs.org/express/-/express-3.4.4.tgz
npm http GET https://registry.npmjs.org/connect/2.11.0
npm http GET https://registry.npmjs.org/commander/1.3.2
npm http GET https://registry.npmjs.org/fresh/0.2.0
npm http GET https://registry.npmjs.org/methods/0.1.0
npm http GET https://registry.npmjs.org/send/0.1.4
npm http GET https://registry.npmjs.org/mkdirp/0.3.5
npm http GET https://registry.npmjs.org/cookie/0.1.0
npm http GET https://registry.npmjs.org/buffer-crc32/0.2.1
npm http GET https://registry.npmjs.org/cookie-signature/1.0.1
```

L'ensemble du Framework Express est désormais installé sur votre ordinateur. Par le biais de npm, une nouvelle commande est installée. Express fournit une commande permettant de générer le squelette de l'application en renseignant différents paramètres : express(1).

On peut ainsi choisir si l'on désire activer ou non les sessions, utiliser le css classique ou less ou encore choisir le moteur de template. Cela vous permet de générer très rapidement l'architecture de l'application, la configuration de base, les dépendances ainsi que quelques exemples.

```
c:\node>express --sessions --css stylus --hogan monsite

   create : monsite
   create : monsite/package.json
   create : monsite/app.js
   create : monsite/public
   create : monsite/public/javascripts
   create : monsite/public/images
   create : monsite/public/stylesheets
   create : monsite/public/stylesheets/style.styl
   create : monsite/routes
   create : monsite/routes/index.js
   create : monsite/routes/user.js
   create : monsite/views
   create : monsite/views/index.hjs

   install dependencies:
     $ cd monsite && npm install

   run the app:
     $ node app
```

Dans cet exemple, j'ai utilisé « stylus » qui correspond au style css classique. J'utilise également hogan.js en tant que moteur de template. Celui-ci est très simple d'utilisation et offre de très bonnes performances. Hogan.js est en réalité basé sur le moteur Mustache.js. Les fonctionnalités offertes par ce moteur sont basiques mais offrent des options de compilation et d'excellentes performances. Ensuite, il nous suffit d'installer les dépendances après la création de l'application.

```
c:\node>cd monsite

c:\node\monsite>npm install
npm WARN package.json application-name@0.0.1 No README.md file found!
npm http GET https://registry.npmjs.org/express/3.4.4
npm http GET https://registry.npmjs.org/hjs
npm http GET https://registry.npmjs.org/stylus
npm http 200 https://registry.npmjs.org/express/3.4.4
npm http 200 https://registry.npmjs.org/hjs
npm http GET https://registry.npmjs.org/express/-/express-3.4.4.tgz
npm http GET https://registry.npmjs.org/hjs/-/hjs-0.0.6.tgz
npm http 200 https://registry.npmjs.org/express/-/express-3.4.4.tgz
npm http 200 https://registry.npmjs.org/hjs/-/hjs-0.0.6.tgz
npm http 200 https://registry.npmjs.org/stylus
npm http GET https://registry.npmjs.org/stylus/-/stylus-0.40.0.tgz
npm http 200 https://registry.npmjs.org/stylus/-/stylus-0.40.0.tgz
npm http GET https://registry.npmjs.org/hogan.js
npm http GET https://registry.npmjs.org/connect/2.11.0
npm http GET https://registry.npmjs.org/commander/1.3.2
npm http GET https://registry.npmjs.org/range-parser/0.0.4
npm http GET https://registry.npmjs.org/mkdirp/0.3.5
```

Vous pouvez maintenant lancer votre première application via la commande : node app.js

Nodemon

Afin de faciliter vos développements, je vous conseille également d'installer le package nodemon. Celui-ci permet de ne plus avoir besoin de redémarrer votre application à chaque modification. Il incorpore un système de tracking, permettant de détecter les changements dans votre application. Installation :

npm install –g nodemon

Utilisation :

nodemon [nom de l'application]

Notre première application

Nous allons regarder plus en détail notre application. SI nous regardons le contenu du fichier app.js, nous pouvons voir que le cœur de l'application est déjà généré grâce à la commande exécutée précédemment.

Nous avons tout d'abord l'instanciation du Framework Express :

```
var app = express();
```

Express offre la possibilité d'utiliser différentes propriétés propres à l'application et nécessaires au bon fonctionnement de celle-ci. Voici quelques exemples des propriétés renseignées :

```
// port de l'application
app.set('port', process.env.PORT || 3000);
// chemin vers le dossier contenant les vues
app.set('views', path.join(__dirname, 'views'));
// Choix du moteur de template, ici hogan
app.set('view engine', 'hjs');
```

Contrairement à d'autres langages, ici tout est fait en javascript. Ainsi, le serveur web est instancié grâce au module http de node.js. Il nous suffit tout simplement de donner en paramètre à la méthode http.createServer l'objet instancié précédemment correspondant à Express :

```
http.createServer(app).listen(app.get('port'), function () {
  console.log('Express server listening on port ' + app.get('port'));
});
```

Les middlewares incontournables

Javascript et donc node.js sont des langages basés sur les évènements. Express utilise le module « Connect ». Celui-ci fonctionne comme un module classique. Il nous permet de gérer une pile de méthodes associées aux évènements dans le cycle de vie de l'application. C'est comme cela que fonctionne un middleware. Lorsque l'utilisateur effectue une requête sur notre serveur web, le Framework Express réalisera plusieurs actions par le biais de plusieurs middlewares. Cela permet par exemple d'initialiser les sessions, de mettre en place un système de logs ou encore d'analyser la requête envoyée au serveur. Nous pouvons également créer nos propres middlewares afin d'apporter à notre application de nouveaux traitements. Un exemple sera montré plus tard dans cet article.

Dans le fichier app.js, nous avons par défaut les principaux middlewares qui sont utilisés. Afin d'activer ceux-ci, la fonction app.use est mise à disposition.

```
var app = express();

app.use(express.logger('dev'));
app.use(express.bodyParser());
app.use(express.methodOverride());
app.use(express.cookieParser(mywebsite));
app.use(express.session());
```

Il existe une grande quantité de middlewares permettant de simplifier les développements et d'implémenter les fonctionnalités les plus courantes. BasicAuth, bodyParser, responseTime, favicon ou encore query sont des exemples de middlewares couramment utilisés.

Le routing

Nous allons maintenant commencer à créer nos propres pages. Le point d'entrée de l'utilisateur sur la page de notre application web sera l'url. Il nous faut donc établir les différentes routes que notre application souhaite offrir. Par défaut, Express crée plusieurs routes.

```
// On inclut le module à l'emplacement ./routes/index.js
var routes = require('./routes');
//  …
// route correspondant à la page d'accueil
app.get('/', routes.index);
```

Dans le dossier "routes" vous trouverez ainsi le fichier index.js qui correspond à la partie Controller de notre modèle MVC.
La méthode app.get est utilisée dans cet exemple. La méthode get peut être remplacée par le verbe http voulu, cela est notamment très utilisé pour les applications REST. Le Framework fournit également la méthode app.all qui renseignera la même route pour tous les verbes, que celui-ci soit get, post ou autre. Ces méthodes prennent plusieurs paramètres :
Req : requête http
Res : réponse http
En ouvrant le fichier routes/index.js

```
exports.index = function (req, res) {
  res.render('index', { title: 'Express' });
};
```

On voit ici le contenu de la route « index ». Comment cela fonctionne ? La route est tout simplement associée à une ou plusieurs url permettant ensuite d'exécuter la fonction correspondante.
La route, sous forme de string, peut également utiliser des paramètres. Si nous prenons le contexte d'un blog, nous sommes souvent amenés à sélectionner un article ou une catégorie en fonction d'un paramètre dans l'url, par exemple : http://monsite.com/articles/2
Afin de renseigner un paramètre sur une route, je vais utiliser le caractère « : ». Ainsi, ma route sera de la forme suivante :

```
app.get('/articles/:id', routes.article);
```

Un paramètre peut être mis en optionnel en le suffixant du caractère « ? », ce qui permettra d'utiliser une valeur par défaut ou de prévoir un comportement différent.

```
app.get('/articles/:id?', routes.article);
```

Ceux-ci sont ensuite accessibles dans le code correspondant à ma route par le biais de l'objet req.params. En suivant l'exemple précédent, j'aurai donc une propriété req.params.id

Il est également possible d'utiliser des paramètres de type GET classique de la forme ?monparam=value. On pourrait ainsi avoir une url du type http://localhost:3000/articles?id=2.
Dans le cas présent, j'utiliserai l'objet req.query afin de récupérer la valeur de mon id. Ce qui me donnera donc req.query.id.
D'une façon plus générique, la méthode req.param('monparam') permet de ne pas se soucier de la provenance de mon paramètre. Cependant il est toujours plus sûr de situer l'endroit où notre paramètre est attendu.
Afin de prévoir la suite des différents exemples, je vais ajouter deux routes à mon fichier app.js

```
    app.post('/', routes.addPost);
      app.get('/', routes.index);
```

Il est possible de réaliser plusieurs actions pour la même route. En effet, il suffit pour cela d'ajouter en paramètres plusieurs callbacks afin de réaliser différentes actions. Il est par contre nécessaire dans le cas présent de spécifier le paramètre next à nos méthodes. Celui-ci fait référence à la prochaine action à exécuter. Par exemple :

```
app.post('/', function (req, res, next) {
  console.log('handle request');
  // call next method
  next();
}, routes.addPost);
```

Dans le cadre du développement web, nous sommes très rapidement amenés à utiliser une base de données. Ici je vais vous présenter MongoDB, une base de données de type NoSQL qui sera très performante et qui fonctionne avec node.js. Pour cela, il vous faudra donc installer une base de données de type mongodb sur votre environnement en fonction de votre système d'exploitation. (http://www.mongodb.org/downloads)
Une fois la base de données installée, je vais devoir installer le module node.js me permettant d'accéder à celle-ci. Pour cela, je vais installer le module « mongodb» grâce à NPM.

```
c:\node\monsite>npm install mongodb
npm WARN package.json application-name@0.0.1 No README.md file found!
npm http GET https://registry.npmjs.org/mongodb
npm http GET https://registry.npmjs.org/mongodb
npm http 200 https://registry.npmjs.org/mongodb
npm http GET https://registry.npmjs.org/mongodb/-/mongodb-1.3.19.tgz
npm http 200 https://registry.npmjs.org/mongodb/-/mongodb-1.3.19.tgz
npm http GET https://registry.npmjs.org/bson/0.2.2
npm http GET https://registry.npmjs.org/kerberos/0.0.3
npm http 200 https://registry.npmjs.org/bson/0.2.2
npm http GET https://registry.npmjs.org/bson/-/bson-0.2.2.tgz
npm http 200 https://registry.npmjs.org/kerberos/0.0.3
npm http GET https://registry.npmjs.org/kerberos/-/kerberos-0.0.3.tgz
npm http 200 https://registry.npmjs.org/bson/-/bson-0.2.2.tgz
npm http 200 https://registry.npmjs.org/kerberos/-/kerberos-0.0.3.tgz
```

Un premier objet Modèle

J'ai créé un premier objet en me servant de modèle. Celui-ci correspond à un post sur un blog ou autre avec un nom et un contenu. Je crée un fichier ./models.PostModel.js avec les méthodes suivantes :

```
module.exports = function (db) {
    this.db = db;
 };

 module.exports.prototype = {
    setDB: function (db) {
      this.db = db;
    },
    db: null,
    insert: function (data, callback) {
      this.db.collection('posts').insert(data, {}, callback);
    },
```

```
    getPosts: function (callback) {
      this.db.collection('posts').find().toArray(callback);
    },

  };
```

Il est possible de mettre en place des conditions. Pour cela, je vais ajouter un paramètre à la méthode find() afin que mongodb réalise la requête adéquate. La méthode suivante permettra de récupérer les posts en comparant la propriété name de l'objet avec la valeur du paramètre searchName.

```
getPostsByName: function (searchName, callback) {
this.db.collection('posts').find({ name: searchName }).toArray
(callback);
}
```

La connexion à la base de données

Après avoir créé un premier modèle, je souhaite maintenant l'utiliser dans mon application. Je vais tout d'abord devoir me connecter sur la base de données et la rendre accessible depuis mes différentes routes.
Il existe différents modules pour gérer facilement la connexion à notre base de données. Je vais utiliser ici l'objet MongoClient et la méthode connect. Cette fonction prend en paramètre un callback permettant d'utiliser la connexion une fois celle-ci instanciée. Ainsi dans le fichier app.js :

```
var MongoClient = require('mongodb').MongoClient;
MongoClient.connect('mongodb://' + config.mongo.host + ':'
+ config.mongo.port + '/posts', function (err, db) {
  if (!err) {
    var attachDB = function (req, res) {
      req.db = db;
    };
  app.post('/', attachDB, routes.addPost);
    app.get('/', attachDB, routes.index);

    http.createServer(app).listen(3000, function () {
      console.log('success');
    });
  }
});
```

Afin de pouvoir accéder à ma base de données depuis les routes, j'ai ajouté la propriété « db » correspondant à ma connexion à mongodb à mon objet request par une fonction que j'ai appelée « attachDB ». Cette méthode fonctionnera comme tout middleware. Les routes peuvent ainsi prendre plusieurs callbacks en paramètres qui seront exécutés dans l'ordre de passage défini.

Le controller

Je vais maintenant modifier mes routes afin de pouvoir utiliser la connexion à ma base de données. Ainsi, le fichier ./routes/index.js devient :

```
var postModel = require(«../models/PostModel»);
module.exports = {
 name: «Accueil»,
 posts: null,
 index: function(req, res, next) {
   var self = this;
// Appel de la fonction getPosts
   this.getPosts(function() {
     // On affiche la vue correspondante
     res.render('index', {posts : self.posts, title : self.name});
   })
 },
,
 getPosts: function(callback) {
   var self = this;
   self.posts = {};
var model = new postModel();
   model.setDB(req.db);
   model.getPosts(function(err, records) {
       self.posts = records;
       callback();
   });
 }
}
```

On remarque la méthode suivante :

```
res.render('index', { posts: self.posts, title: self.name });
```

Ici je sélectionne la vue "./views/index.hjs" et lui fournis plusieurs paramètres. Tout d'abord les posts en provenance de ma base de données ainsi que le titre de ma page que j'utiliserai par la suite dans la vue.
Pour gérer la connexion à ma base de données, j'ai instancié le module de mon modèle. Sur celui-ci, j'avais défini une fonction setDb. Grâce au middleware ajouté lors de la définition de la route, je peux désormais obtenir la connexion à ma base de données depuis l'objet req.
J'ajoute également une méthode addPost afin de pouvoir ajouter des entrées à ma base de données par le biais d'un formulaire :

```
addPost: function(req, res){
   var self = this;
   model.setDB(req.db);
   if(req.body) {
     var data = {
       name: req.body.name,
       content: req.body.content
     };
       model.insert(data, function(){
       self.index(req, res);
     });
   }
 }
```

Afin de récupérer les paramètres en provenance d'un formulaire, le middleware bodyParser va effectuer une grande partie du travail. Il nous suffit simplement d'utiliser l'objet req.body afin d'obtenir la valeur des paramètres de type POST envoyés par le formulaire. Nous appelons ensuite la méthode insert de notre model afin d'ajouter le post à notre base de données.

La vue

Nous allons maintenant nous intéresser à la partie « View » de notre modèle MVC. La méthode res.render nous permet de sélectionner la vue à retourner, ici « ./views/index.hjs ».
Dans le fichier view/index.js je vais renseigner la structure html de ma page et également utiliser les paramètres que le controller me fournit.

```
<!DOCTYPE html>
<html>
<head>
 <title>{{ title }}</title>
```

```
  <link rel='stylesheet' href='/stylesheets/style.css' />
</head>
<body>
  {{#posts}}
  <div class=»"item"«>
    {{name}} : {{content}}
  </div>
  {{/posts}}
  <br />
  <form method=»post»>
  Name :
  <input type=»text» name=»name» /><br />
  Content :
  <input type=»text» name=»content» />
  <input type=»submit» value=»send» />
  </form>
  </div>
</body>
</html>
```

Afin d'utiliser les paramètres, je dois utiliser la syntaxe Mustache. Pour afficher la valeur de mon paramètre, j'utilise des accolades : {{title}}. Cette syntaxe permet de dire au view-engine qu'il s'agit d'un paramètre et qu'il faut donc afficher la valeur de celui-ci et non du texte classique.
Sur cette vue, je souhaite afficher l'ensemble des posts de ma base de données. Afin d'afficher chaque post, je vais utiliser une boucle. Afin de réaliser cette boucle, similaire à un foreach, j'utilise la syntaxe suivante :

```
{{# posts }}
{{/#posts}}
```

Je peux ensuite utiliser les propriétés de mon objet. « Posts » est une collection en provenance de mongodb et a donc la propriété « name » et « content ».
Afin de permettre l'ajout de données dans ma base, j'utilise également un formulaire. Ce formulaire appellera ensuite la route addPost que nous avions définie sur la page d'accueil pour les requêtes de type POST.

Travailler sur plusieurs environnements

Dans ces exemples, j'ai travaillé avec un environnement local et le port par défaut de notre serveur web (3000). Cependant, dans un contexte de projet client par exemple, plusieurs environnements sont nécessaires. En fonction de l'architecture voulue, un serveur de recette, de pré-production ou encore de production peuvent être nécessaires.
Pour des raisons de sécurité, les chaines de connexion, les ports utilisés peuvent varier d'un environnement à l'autre. Pour permettre cela, plusieurs méthodes existent. La première méthode consiste à utiliser une variable d'environnement. Dans le fichier app.js, nous pouvons trouver la ligne suivante :

```
app.set('port', process.env.PORT || 3000);
```

Ainsi il est possible de stocker les informations nécessaires par le biais de variables d'environnement.
Cependant, rien ne vous empêche de créer votre propre système de configuration en créant un objet stockant l'ensemble de vos données. Cela peut être utile principalement dans le cas où vous utilisez des éléments de configuration propres au déroulement et aux fonctionnalités de votre application.
Voici un exemple de fichier qui pourrait être créé afin de gérer la configuration. Ce fichier se trouve à l'emplacement ./config/index.js

```
var config = {
  local: {
    port: 3000,
    mongo: {
      host: '127.0.0.1',
      port: 27017
    }
  },
  production: {
    port: 3002,
    mongo: {
      host: '127.0.0.1',
      port: 27019
    }
  }
}
module.exports = function (mode) {
  return config[mode || process.argv[2] || 'local'] || config.local;
}
```

Dans le fichier app.js, il me suffira d'utiliser cette configuration comme tout autre objet javascript.

```
var config = require('./config')();
// …
http.createServer(app).listen(config.port, function () {

});
```

Il me suffit ensuite de lancer mon serveur avec le paramètre correspondant à mon environnement : node app.js production

Les Tests

On parle de plus en plus aujourd'hui de la nécessité de prendre en compte les tests dans nos développements.
Pour ma part, j'utilise Jasmine afin de réaliser mes tests. Il existe un module pour node correspondant à jasmine. npm install –g jasmine-node
Pour cela je crée un fichier .spec à l'emplacement suivant : ./tests/test-mongodb.spec

```
describe(«MongoDB», function () {
  it(«is there a MongoDb server running», function (next) {
    var MongoClient = require('mongodb').MongoClient;
    MongoClient.connect('mongodb://127.0.0.1:27017/posts',
  function (err, db) {
          expect(db).toBeDefined();
          expect(err).toBe(null);
           next();
    });
  });
});
```

Il nous suffit ensuite d'exécuter les tests :
jasmine-node ./tests

Matthieu **Bosi**
Ingénieur de développement
matthieu.bosi@itelios.com
Itelios

Ecrivez vos librairies de composants ASP.Net MVC

L'ASP.NET MVC est un Framework complémentaire à ASP.NET et permettant de développer des sites web en appliquant le Pattern MVC. Il vient en complément des WebForms. Il est ainsi possible de choisir entre les deux en fonction des besoins, l'approche MVC étant conseillée pour la mise en place de sites complexes.
Il permet d'avoir une maîtrise très précise du code HTML généré. En contrepartie, le temps de développement induit est important. Ce temps peut être réduit en utilisant les Helpers HTML, et encore plus en mettant en œuvre des composants prenant en entrée le modèle et générant l'élément HTML correspondant, tel un tableau, un formulaire, etc.

Qu'est ce qu'un « Helper » MVC ?

De base, le framework MVC contient un certain nombre de méthodes permettant de simplifier l'écriture des pages en leur déléguant l'écriture du code HTML. Ces méthodes permettent, entre-autres, de définir facilement les contrôles de base d'une page tout en spécifiant les éléments du modèle qui y sont rattachés.
Ainsi il est possible d'ajouter la ligne suivante dans une vue :

```
@Html.ActionLink(«Sample Grid», «Grid», «Sample»)
```

Cette ligne permet d'ajouter un lien hypertexte en indiquant le contrôleur à appeler. Le code HTML généré est le suivant :

```
<a href=»/Sample/Grid»>Sample Grid</a>
```

Pour la mise en place d'une zone de saisie d'un élément du modèle :

```
@Html.EditorFor(model => model.Value)
```

Qui donne le code HTML suivant :

```
<input class=»text-box single-line» id=»Value» name=»Value» type=»text» value=»» />
```

Description du composant

Le composant que nous allons définir est une grille simple permettant d'afficher le contenu d'une liste d'éléments. Une des implémentations possibles pour mettre en œuvre une telle grille sans composant serait :

```
<table class=»table table-striped»>
  <tr>
    <th>@Html.DisplayNameFor(m => m.ProductList[0].ProductID)</th>
    <th>@Html.DisplayNameFor(m => m.ProductList[0].Name)</th>
    <th>@Html.DisplayNameFor(m => m.ProductList[0].ProductNumber)</th>
  </tr>
  @foreach (var item in Model.ProductList)
  {
    <tr>
      <td>
        @item.ProductID
      </td>
      <td>
        @item.Name
      </td>
      <td>
        @item.ProductNumber
      </td>
    </tr>
  }
</table>
```

Ce qui donne le tableau (Figure 1) sur la page générée :
Nous allons donc mettre en place un composant permettant de générer ce tableau automatiquement en analysant les propriétés du type des éléments qui seront passés en paramètres.

Mise en place du composant

La première étape consiste à générer une nouvelle « Application web ASP.NET MVC 4 » pour initialiser les vues et les modèles nécessaires au fonctionnement du composant. Ainsi qu'un projet « bibliothèque de classe » qui abritera le composant et l'ensemble des sous-éléments nécessaires à son fonctionnement. Voici l'arborescence de la solution : (figure 2)
Nous devons ensuite référencer « SampleControls » dans « SampleSite » et enregistrer le « namespace » dans le fichier de configuration du dossier Views (figure 3). Ceci rend accessible les classes du namespace « SampleControls » dans toutes les vues.
Dans la foulée, nous pouvons utiliser une classe statique pour exposer nos composants pour les utiliser comme les HTML Helpers du namespace « System.Web.Mvc.Html ». La classe statique contient le code suivant :

```
namespace SampleControls
{
  public static class CCPoc
  {
    public static MvcHtmlString Grid<T>(IEnumerable<T> val) where T : class
    {
```

ProductID	Name	ProductNumber
1	Adjustable Race	AR-5381
2	Bearing Ball	BA-8327
3	BB Ball Bearing	BE-2349
4	Headset Ball Bearings	BE-2908
316	Blade	BL-2036
317	LL Crankarm	CA-5965
318	ML Crankarm	CA-6738
319	HL Crankarm	CA-7457
320	Chainring Bolts	CB-2903
321	Chainring Nut	CN-6137

Fig.1

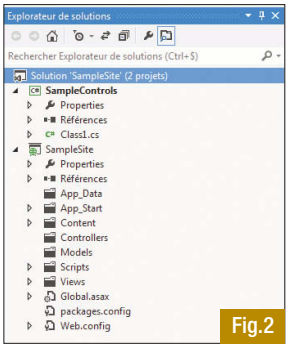


Fig.2

Arborescence de la solution

Fig.3

Enregistrement du namespace contenant le composant

```
            return new Grid().Generate(val);
        }
    }
}
```

L'utilisation des génériques va nous permettre de prendre n'importe quel type de classe en entrée. La méthode statique de la classe va lancer la génération de la grille avec en entrée la liste d'éléments à afficher.
Ajoutons la classe Grid qui abritera la logique de création du code HTML de génération du tableau :

```
internal class Grid
{
    public Grid()
    {
    }

    public MvcHtmlString Generate<T>(IEnumerable<T> model) where T : class
    {
        return MvcHtmlString.Create(«<div>Test</div>»);
    }
}
```

Pour le moment nous allons juste renvoyer une balise « div » contenant du texte. Créons une vue pour utiliser ce composant, celle-ci contient le code suivant :

```
@model SampleSite.Models.ProductModel

@{
    ViewBag.Title = «Sample Grid»;
}

@CCPoc.Grid(Model.Products)
```

Le contrôleur associé ne fait que fournir le modèle « ProductModel », celui-ci contenant une liste « Products » de « ProductElement » ainsi définie :

```
public class ProductElement
{
    public int Id { get; set; }

    public string Name { get; set; }

    public string Number { get; set; }
}
```

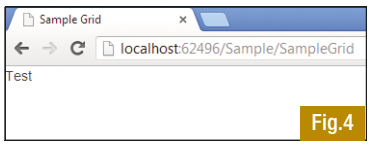


Fig.4

L'appel de la vue donne le résultat suivant (Figure 4)
Du côté du contrôleur il suffit d'instancier le modèle et le transmettre à la vue :

```
public ActionResult SampleGrid()
    {
        return View(new ProductModel());
    }
```

De l'objet au tableau

Maintenant que nous avons le composant fonctionnel, il nous reste à ajouter la logique de création du tableau.

Nous avons en entrée du composant une liste d'objets, il est donc possible d'analyser la structure de la classe en utilisant la réflexion pour déduire la structure du tableau.
Mais comment faire passer de la logique d'affichage au composant pour définir les propriétés qui doivent être affichées, le libellé des en-têtes, et autres éléments non décrits par la structure de la classe ?
Pour cela nous pouvons ajouter un attribut personnalisé qui sera utilisé dans le modèle et interprété par le composant. Ajoutons donc la classe « GridColumnAttribute » dans le projet « SampleControls » et contenant le code suivant :

```
namespace SampleControls
{
    public class GridColumnAttribute : Attribute
    {
        public GridColumnAttribute()
        {
            this.Display = true;
        }

        public string HeaderText { get; set; }

        public bool Display { get; set; }
    }
}
```

Nous pouvons alors décorer les propriétés de notre modèle avec cet attribut :

```
    public class ProductElement
    {
        [GridColumn(HeaderText=»Product Id»)]
        public int Id { get; set; }

        [GridColumn(HeaderText = «Product Name», Display = true)]
        public string Name { get; set; }

        [GridColumn(HeaderText = «Product Number», Display = true)]
        public string Number { get; set; }
    }
```

Il reste encore à analyser la classe dans le composant pour constituer le tableau. Pour ce faire, nous devons utiliser la réflexion pour récupérer la liste des propriétés de la classe et pour chacune vérifier si elle est décorée avec notre attribut personnalisé :

```
PropertyInfo[] properties = type.GetProperties();

foreach (PropertyInfo property in properties)
{
    var attributes = property.GetCustomAttributes(typeof(GridColumnAttribute), true);
    var attribute = new GridColumnAttribute();
    if (attributes != null && attributes.Length > 0)
    {
        attribute = (GridColumnAttribute)attributes[0];
    }
```

Une fois ces informations obtenues, il faut ensuite parcourir la liste des éléments passés au composant pour construire le code HTML contenant les données.
Pour la génération du code HTML nous pouvons utiliser la classe « Tag-

Builder » qui est contenue dans l'assembly « System.Web.WebPages ». Celle-ci s'utilise de la manière suivante :

```
TagBuilder table = new TagBuilder(«table»);
table.AddCssClass(«table»);
table.AddCssClass(«table-striped»);
TagBuilder thLine = new TagBuilder(«tr»);

foreach (GridColumnInfo colInfo in gridStruct.ColumnInfos)
{
    if (!colInfo.Display)
    {
        continue;
    }

    TagBuilder currentCell = new TagBuilder(«th»);
    currentCell.SetInnerText(colInfo.Header);

    thLine.InnerHtml += currentCell.ToString();
}

table.InnerHtml += thLine.ToString();

foreach (List<string> lineValues in gridStruct.CellValues)
{
    TagBuilder tr = new TagBuilder(«tr»);

    foreach (string cellValue in lineValues)
    {
        TagBuilder currentCell = new TagBuilder(«td»);
        currentCell.SetInnerText(cellValue);

        tr.InnerHtml += currentCell.ToString();
    }

    table.InnerHtml += tr.ToString();
}

return table.ToString();
```

Le tableau généré est le suivant (figure 5).

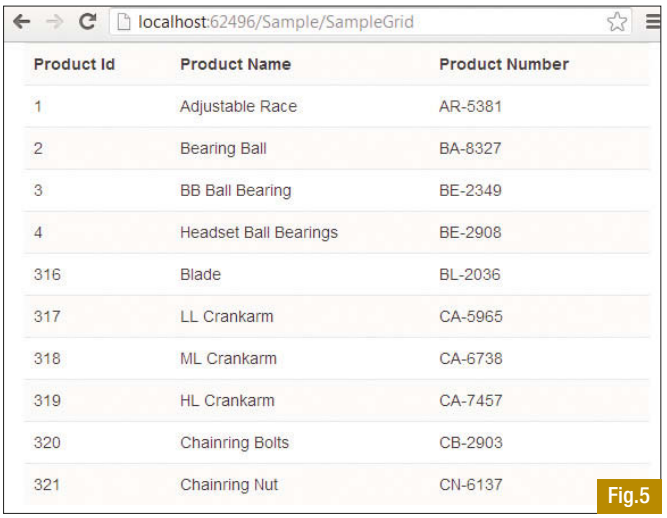


Fig.5

Conclusion

L'écriture de tels composants a plusieurs avantages comme les possibilités de réutilisations, l'accélération du développement, la mise en commun de l'apparence des éléments mis en place, et surtout la simplification de la maintenance. Mais un composant présente aussi des limites notamment sur son degré de personnalisation qui ne peut pas forcément couvrir tous les cas de présentation et d'organisation des données. L'intérêt principal du développement de tels composants réside dans l'économie d'échelle en termes de temps de développement qui sera réalisée par leur réutilisation systématique dans différents projets.

Eric **Dumortier**, *ingénieur expert .Net, Osaxis*

Développer un gestionnaire de contact couplé à Google Maps avec le mkFramework

Vous avez pu découvrir dans les numéros 167 et 170 de votre magazine Programmez! un framework qui se veut simple d'utilisation: le mkFramework. En effet pour utiliser celui-ci, pas besoin de ligne de commande, une interface web "le builder" vous permet facilement de créer de nouveaux projets web et de les administrer.
Dans ce tutoriel, nous allons créer une application web de gestion de contact couplée à une géolocalisation grâce à Google Maps.

Télécharger et installer le framework

Rendez-vous à l'adresse http://mkframework.com
Copiez ce répertoire zip dans votre répertoire web, et désarchivez-le
Ouvrez votre navigateur sur http://localhost/mkframework_v4_XX_YY_rZZZ en fonction de la version téléchargée.

Créer la base de données

Dans votre base de données mysql, nous allons créer une table "contact".
Exécutez la requête SQL suivante :

```
CREATE TABLE `contact` (
`id` int(11) NOT NULL auto_increment,
`nom` varchar(50) NOT NULL,
`prenom` varchar(50) NOT NULL,
`adresse` varchar(50) NOT NULL,
`cp` varchar(50) NOT NULL,
`ville` varchar(50) NOT NULL,

PRIMARY KEY  (`id`)
);
```

Utilisons le générateur web ou "builder"

Le mkframework vous propose via son générateur de gagner beaucoup de temps.
Créez votre projet: renseignez "gestionDeContact",
puis cliquez sur le bouton "Créer"
Le générateur va créer votre projet dans le répertoire data/genere du framework
Cliquez sur le lien "Explorer le projet", dépliez le répertoire "conf" et cliquez sur le fichier connexion.ini.php
Renseignez une connexion comme suit :

```
contactBase.dsn=»mysql:dbname=contactBase;host=localhost»
contactBase.sgbd=pdo_mysql
contactBase.username=root
contactBase.password=votreMotDePasse
```

Cliquez ensuite sur le second onglet (pour voir la liste des projets) et cliquez sur le lien "éditer" sur la ligne de votre projet "gestionDeContact",
Cliquez sur "générer la couche modèle", ensuite sur le profil "contactBase"
Cliquez enfin sur le bouton "générer",
Le générateur va générer un fichier "modèle" model/model_contact.php.

Créons un module CRUD

Dans cette application, nous souhaitons créer, modifier et supprimer nos contacts, c'est la définition du CRUD, heureusement le générateur web va vous aider.
Cliquez sur le lien "générer un CRUD"
Sélectionnez la classe model_contact.php
Cliquez sur "générer"
Le générateur va créer un module CRUD "contact" dans le répertoire module/contact/ Fig.1.
Créez quelques contacts avec des coordonnées, afin de remplir la liste du tableau :

nom	prenom	adresse	cp	ville	
Hugo	Victor	4 rue montmartre	75002	Paris	Edit | Delete | Show
Asimov	Isaac	7 rue monge	75005	paris	Edit | Delete | Show

New

Ajoutons une vérification des données

Le mkframework propose nativement un mécanisme de vérification des données.
Celui-ci est situé dans la classe modèle de chaque table, plus précisément dans la méthode getCheck(); vous pouvez vérifier que le champ n'est pas vide, qu'il respecte une expression régulière...
Editez la méthode getCheck() de la classe modèle
model/model_contact.php

```
private function getCheck(){
    $oPluginValid=new plugin_valid($this->getTab());
```

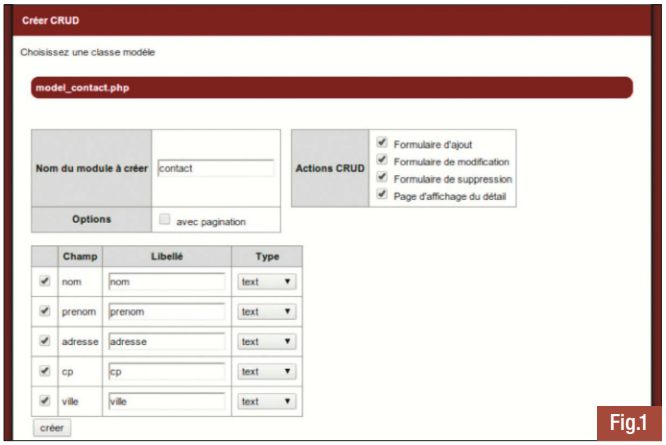


Fig.1

```
    $oPluginValid->isNotEmpty('nom','Le champ ne doit pas
&ecirc;tre vide');
    $oPluginValid->isNotEmpty('prenom','Le champ ne doit pas
&ecirc;tre vide');
    $oPluginValid->isNotEmpty('adresse','Le champ ne doit
pas &ecirc;tre vide');
    $oPluginValid->isNotEmpty('cp','Le champ ne doit pas
&ecirc;tre vide');
    $oPluginValid->isNotEmpty('ville','Le champ ne doit pas
&ecirc;tre vide');

    return $oPluginValid;
  }
```

Importer le module google Maps

La structure modulaire du mkframework (depuis 2009) permet facilement d'ajouter à vos projets des modules intégrables.
Vous pouvez ainsi développer des modules et les réutiliser dans d'autres projets.
Rendez-vous sur la page télécharger du mkFramework, et cliquez sur le module Google Map
http://mkframework.com/telechargerModule_googleMap.html
Téléchargez l'archive zip et désarchivez-la dans le répertoire module de votre projet (dans gestionDeContact/module/)
Comme vous pouvez le voir, il y a des instructions pour utiliser le module.

Intégrons notre carte Google Map à l'application

Nous allons, sous le tableau des contacts, afficher la liste de ceux-ci sur une carte google map, avec leurs noms ainsi qu'un lien menant au formulaire d'édition du contact.
Pour cela nous ajouterons à la suite du code chargeant les contacts, le code permettant de créer la carte ainsi qu'une boucle pour ajouter ces contacts sur la carte.

Editez la méthode _list() du fichier module/contact/main.php

```
public function _list(){

    $tContact=model_contact::getInstance()->findAll();

    $oView=new _view('contact::list');
    $oView->tContact=$tContact;

    $this->oLayout->add('main',$oView);

    $oModuleGoogleMap=new module_googleMap();
    $oModuleGoogleMap->setWidth(500);
    $oModuleGoogleMap->setHeight(400);
```

Fig.2


```
    $oModuleGoogleMap->setZoom(12);

    foreach($tContact as $oContact){
      $oModuleGoogleMap->addPositionWithContent(
          $oContact->adresse.','.$oContact->cp,
          $oContact->nom.' '.$oContact->prenom,
          array(
          '<h1>'.$oContact->prenom.' '.$oContact->nom.'</h1>',
          '<p>'.$oContact->adresse.','.$oContact->cp.'</p>',
          '<a href=»'._root::getLink('contact::edit',array('id'
=>$oContact->id),false).'»>Edit</a>'
          )

          );
    }

    $this->oLayout->add('main',$oModuleGoogleMap->getMap());

  }
```

Conclusion

Vous venez de voir dans ce tutoriel qu'il était simple d'intégrer une carte google Map dans vos applications en utilisant le mkFramework.
Voici à quoi ressemble votre application conçue en quelques minutes, vous pouvez facilement l'améliorer et la compléter Fig.2.

Michaël **Bertocchi**
Ingénieur développement
@dupot_org

Les API publiques et leur sécurité en question

Depuis peu, les « API » semblent être la réponse idéale à tous les enjeux d'interconnexion de systèmes et de présentations de services externes. Les départements métiers encouragent leur usage mais la DSI doit faire preuve d'acuité, notamment en termes de sécurité.

Qu'appelle-t-on précisément API ?

Le concept d'Interface de Programmation d'Application (Application Programming Interface) caractérise un mode d'exposition des fonctionnalités d'un applicatif via une bibliothèque logicielle ou un protocole réseau à d'autres applications. Ces dernières, dites « programmes consommateurs », peuvent ainsi utiliser des données ou des fonctionnalités du « programme fournisseur ». Notons que s'il existe de très nombreuses formes d'API, lorsqu'on parle aujourd'hui d'APIs, il s'agit d'APIs basées sur les architectures REST (Representational State Transfer), qui standardisent l'utilisation des verbes HTTP GET, POST, PUT et DELETE pour manipuler les « objets » métiers fournis et consommés par le service exposé. Ces dernières années, avec la mise à disposition des développeurs de leurs principales fonctionnalités telles que cartes géographiques et autres *timelines* par exemple, au travers d'APIs SOAP puis REST par les géants du Web tels que Twitter, Facebook ou Google, la notion d'API a fortement gagné en notoriété .

Quelle est la généalogie des API ?

Les APIs sont les héritières de deux grandes familles de technologies de l'information :

- Les applications d'intégration d'applications d'entreprise (EAI – *Entreprise Application Integration)*, qui répondent au besoin de faire communiquer entre eux des systèmes hétérogènes,
- Les technologies permettant la mise en place de l'informatique distribuée, avec toutes les contraintes liées aux protections périmétriques des systèmes d'information des entreprises que cela comporte (*firewall* notamment).

A la fin des années 90 les Web Services portaient la promesse d'une réponse globale à cette double problématique : utilisant XML comme format universel et interopérable de représentation des données et le protocole HTTP (le « protocole universel »), en raison de l'ouverture permanente du port 80.

Pourquoi les API REST remplacent-elles peu à peu SOAP, JSON et le XML ?

Avec les APIs SOAP, clients et serveurs échangeaient donc des messages SOAP. Ce mode de fonctionnement posait trois problèmes majeurs :

- La complexité et le coût CPU de la prise en charge du langage XML,
- La verbosité du XML, induisant une consommation excessive de la bande passante réseau,
- La complexité programmatique de la génération des messages clients (difficulté à consommer les services),

Ces problèmes sont devenus particulièrement patents avec l'apparition des terminaux mobiles comme les téléphones et les tablettes. Depuis quelques années, on assiste donc au remplacement de XML par le format JSON (JavaScript Object Notation) pour la représentation des données, ainsi que de celui de SOAP par REST pour la méthode de consommation des services exposés.

API Based Architecture *vs* Service Oriented Architecture ?

Dans l'esprit, le principe est le même : proposer la réutilisation de données et fonctionnalités préexistantes au sein du système d'information, au travers de services unitaires. Mais le concept SOA (Service Oriented Architecture) a très vite trouvé ses limites en raison d'une mauvaise gestion de son concept. En effet, dans la grande majorité des cas, la granularité et la redistribution des services unitaires ont été mal maîtrisées, ce qui en rendait la réutilisation particulièrement difficile ; Sans compter une courbe d'apprentissage très lente et peu de spécialistes présents sur le marché.

Aujourd'hui, les technologies sont maîtrisées mais la mise en œuvre autant que la consommation des services restent complexes et très chronophages. Pragmatique, le marché s'est donc naturellement tourné vers les API REST, plus simples et rapides à déployer et à utiliser, résolvant naturellement la problématique de la granularité des services, dans la mesure où il s'agit de manipulation d'objets.

Quelles précautions en matière de sécurité ?

Les APIs basées sur les technologies REST et JSON, simples et rapides à mettre en œuvre, répondent parfaitement à l'impatience des métiers en matière d'exposition publique de services. Mais selon la nature des données exposées, la sécurité peut devenir un enjeu important voire primordial. En effet, si l'intégration d'une Google Maps ou d'une *timeline* Twitter sur un site Web n'est pas à proprement parler problématique, ce n'est pas le cas, par exemple, de l'accès à ses points de retraite, à son dossier médical ou à la gestion de son compte bancaire. Il y a donc un distinguo crucial à effectuer entre APIs grand public (Open APIs) et APIs privées (Enterprise APIs), qu'il convient de sécuriser au mieux. Or à ce jour, les standards de sécurité sont encore à peine émergeants et les attaques se multiplient. Dans tous les cas, il est indispensable de sécuriser à la fois le transport (par l'utilisation de SSL, par exemple) et le message, en signant et chiffrant ce dernier. Autre point essentiel, dé-corréler la plateforme applicative de celle exposant l'API au monde extérieur, qui doit nécessairement être positionnée en DMZ, afin de maintenir un niveau élevé de sécurité sur le système d'information interne.

Pourquoi de telles précautions ?

En matière d'API, les plus gros risques sont le vol de données et l'intrusion. D'autant que les attaques visent aujourd'hui plus les applications que les machines elles-mêmes. D'où une vraie nécessité de s'appuyer sur des spécialistes de la sécurité applicative quand il s'agit d'exposer les données d'une application, critiques ou non, afin de limiter tant le vol de celles-ci que les risques d'intrusion par des personnes mal intentionnées. Plus globalement, il convient de rester encore très prudent. Et savoir parfois calmer les ardeurs des services marketing qui souhaiteraient tout mettre à disposition des clients. Car encore une fois les standards en matière de sécurité des API n'en sont encore qu'à leurs balbutiements, malgré les travaux communs initiés ça et là, notamment au sein de la Fondation OWASP.

Au sein de la DSI, à qui confier la mise en œuvre des API publiques ?

Deux services de la DSI seront principalement concernés par la gestion des API. Le pôle Etudes et Développements sera naturellement en charge de la création des API : les développeurs connaissent leurs applications, ils sont donc tout désignés pour créer les API correspondantes. En revanche, la publication (ou l'exposition) de ces API est à confier impérativement au pôle Sécurité, Systèmes et Réseaux : lui seul connaît parfaitement ses architectures et la meilleure façon de les sécuriser, afin d'éviter tout risque d'intrusion dans le système d'information ou de vols de données sensibles à l'occasion des échanges d'informations via les API proposées par l'entreprise.

Philippe **Léothaud**,
API Gateway Solution Leader, Axway

Complétez votre collection

PROGRAMMEZ! le magazine du développeur www.programmez.com

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

nouveau

Tout Programmez! sur une clé USB

29,90 €*

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

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

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

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

☐ ***168*** : |__| exemplaire(s)
☐ ***169*** : |__| exemplaire(s)
☐ ***170*** : |__| exemplaire(s)
☐ ***171*** : |__| exemplaire(s)
☐ ***172*** : |__| exemplaire(s)

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

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

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

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

PROG 173

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

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

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

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

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

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

Règlement par chèque à l'ordre de Programmez !

Offre limitée, valable jusqu'au 30 avril 2014

Réaliser une carte interactive en HTML 5

Tournée vers l'amélioration de l'expérience utilisateur au travers d'applications web toujours plus riches, la spécification HTML 5 propose 2 fonctionnalités pour réaliser des rendus graphiques au sein du navigateur. Grande nouveauté, le composant Canvas s'amène avec un large éventail d'applications possibles. Moins relayée, l'arrivée de la balise svg au sein du code HTML devrait pourtant avoir un retentissement tout aussi important. Afin de comparer ces technologies, nous nous proposons de les mettre en concurrence dans un cas d'utilisation de complexité moyenne à savoir la réalisation d'une carte interactive en HTML 5.

Attendue pour fin 2014, la spécification HTML 5 se veut une évolution majeure offrant des applications web plus riches aux utilisateurs. Le champ des possibilités de cette nouvelle mouture est vaste et vise avant tout à proposer des solutions performantes et standardisées. Cette dernière notion est essentielle puisqu'elle doit permettre de s'affranchir d'initiatives propriétaires basées sur des plugins. Dans le domaine du rendu graphique, HTML 5 s'amène avec 2 standards permettant de réaliser in fine le même travail avec Canvas et SVG. Abondance de bien ne nuit pas mais force cependant à se poser la question : quel est le meilleur choix entre Canvas et SVG ? Comme souvent, il n'y a pas de réponse à sens unique, et chaque technologie apporte ses avantages et ses inconvénients. Du contexte d'utilisation dépendra donc le choix réalisé. Au fond, tout se résume à un choix à faire entre 2 formats graphiques : le bitmap et le vectoriel. Dans le cadre de cet article, nous nous intéressons à une problématique bien connue, à savoir la création d'une carte interactive sur une page web. Bien longtemps, ce type de cartes est demeuré chasse gardée de Flash mais aujourd'hui, la donne est différente.

Composant Canvas

Initialement introduit par Apple pour le moteur de rendu WebKit, l'élément Canvas a ensuite été adopté par la plupart des autres navigateurs avant d'être standardisé par le W3C pour HTML 5. Le Canvas est une zone de rendu graphique dont les dimensions sont définies via des attributs en HTML et accessible via la balise canvas. Il propose une API programmable en Javascript mettant en avant des fonctions classiques de dessin 2D. Au niveau de son implémentation, il s'appuie sur un format bitmap au sein duquel les tracés sont réalisés sur des pixels anonymes. On parle ainsi de rendu immédiat ce qui implique un retraçage de l'ensemble du contexte graphique pour le mettre à jour. Ce format induit une perte de qualité lors d'éventuelles opérations de redimensionnement. En outre, les dessins sont réalisés exclusivement de manière programmatique ce qui s'avère un avantage pour des traitements complexes exigeant une maîtrise fine du rendu. Supporté par la majorité des navigateurs desktop et mobiles, le Canvas est déjà utilisé massivement pour réaliser des jeux 2D à base de technologies web standard tournant autour d'HTML 5. On le retrouve aussi souvent sur des sites boursiers ou bien dans d'autres contextes temps réel où une maîtrise fine du traçage s'avère cruciale. Enfin, la création d'images et notamment de cartes interactives rentre dans le périmètre du Canvas et c'est ce vers quoi nous nous dirigeons ici.

Mise en pratique de Canvas

Le Canvas s'ajoute simplement au sein d'une page HTML à l'aide de la balise éponyme :

```
<canvas id=»myCanvas» width=»500px» height=»500px»></canvas>
```

L'interaction avec le Canvas se fait ensuite en Javascript en récupérant dans un premier temps l'élément canvas via son id puis en accédant à son contexte 2 D :

```
var canvas = document.getElementById(«myCanvas»);
var ctx = canvas.getContext(«2d») ;
```

Le contexte expose plusieurs fonctions offrant au développeur les primitives nécessaires à la création de rendus graphiques de qualité. Ainsi, on peut définir les couleurs utilisées pour dessiner une ligne (strokeStyle) ou remplir une zone (fillStyle), dessiner un rectangle rempli (fillRect) ou vide (strokeRect), dessiner des suites de points à l'aide des fonctions moveTo et lineTo, des courbes de Bézier, ou encore écrire du texte via fillText. L'exemple suivant met en oeuvre ces fonctions en dessinant un carré et un triangle via une suite de points (figure 1) :

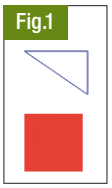


Fig.1
Premier rendu via Canvas

```
var canvas = document.getElementById(«myCanvas»);
var ctx = canvas.getContext(«2d»);
ctx.beginPath();
ctx.moveTo(15,15);
ctx.lineTo(80,15);
ctx.lineTo(80,60);
ctx.closePath();
ctx.fillStyle = «white»;
ctx.fill();
ctx.strokeStyle = «blue»;
ctx.stroke();
ctx.fillStyle = «red»;
ctx.fillRect(15,80,60,60);
```

Carte avec Canvas

La réalisation d'une carte passe par la récupération des données définissant les différentes limites considérées. Elles sont facilement accessibles de par le Web. Une fois l'élément canvas créé et ajouté au DOM, le traçage est effectué sur la zone de dessin. L'opération consiste à se positionner à un point donné via la fonction moveTo du contexte 2D puis à tracer des lignes vers les autres points de la carte définis par leurs coordonnées (x,y). Ce tracé s'effectuant via la fonction lineTo. La personnalisation du tracé se fait en jouant sur des propriétés du contexte telles que fillStyle et strokeStyle par exemple :

```
function draw(ctx) {
 ctx.fillStyle = '#dedede';
 ctx.strokeStyle= 'red';
 ctx.beginPath();
 ctx.moveTo(445.33847,488.9562);
 ctx.lineTo(445.33847,491.11245);
 ctx.lineTo(447.30722,492.48745);
 ctx.lineTo(450.61972,494.42495);
 // ...
```

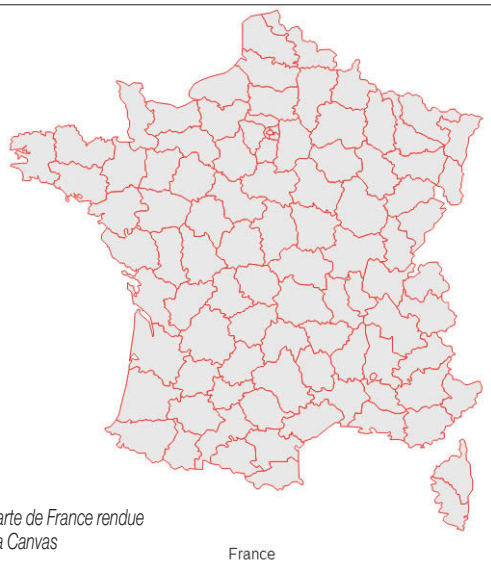
```
 ctx.closePath();
 ctx.fill();
 ctx.stroke();
 ctx.beginPath();
 ctx.moveTo(451.4375,186.03125);
 ctx.lineTo(449.625,186.96875);
 // ...
 ctx.closePath();
 ctx.fill();
 ctx.stroke();
 ctx.fillStyle= 'black';
 ctx.font = '16px Arial';
 ctx.fillText('France', 235, 575);
}
```

On commence la première suite de points via un appel à beginPath() puis on se positionne sur le premier point du tracé avant de tracer les lignes de points en points. L'appel à closePath() termine la suite de points d'une région. C'est alors qu'il faut appeler les fonctions fill() et stroke() sur le contexte afin de dessiner à proprement dit la région. Le résultat présenté à la figure 2 montre une carte des régions de France :

Malgré une perte de qualité au redimensionnement, la carte rendue via Canvas est convenable. Il reste maintenant à passer au côté interactif en rajoutant l'affichage du nom d'une région lors de son survol à la souris. Chaque région étant une simple suite de points dessinée en mode immédiat, il n'y a pas d'autre moyen pour y accéder que de récupérer la position de la souris et de vérifier si la position courante appartient à l'une ou l'autre des régions. Pour cela, on utilise un listener d'évènement mousemove sur le Canvas en convertissant les coordonnées renvoyées par l'évènement en coordonnées (x,y) dans le Canvas :

```
function getMousePos(canvas, evt) {
 var rect = canvas.getBoundingClientRect();
 return {
  x: evt.clientX - rect.left,
  y: evt.clientY - rect.top
 };
}

function init() {
 canvas = document.getElementById(«myCanvas»);
 canvas.addEventListener('mousemove', function (evt) {
  var mousePos = getMousePos(canvas, evt);
  // …
 }, false);

 // ...
}
```

Fig.2



Carte de France rendue via Canvas

Le gros du travail reste à faire puisqu'il faut vérifier à quelle région appartient la position récupérée. Rien de standard pour ce type de job. Une solution consiste à construire chacune des formes et à utiliser une bibliothèque 2D spécialisée permettant de vérifier l'appartenance d'un point à une forme. La partie interactivité de la carte met en lumière certains manques de Canvas qui à lui seul ne peut suffire dans le cas présent.

SVG

Développé à partir de 1999 au sein du W3C pour proposer un standard indépendant aux initiatives d'Adobe (PGML) et Microsoft (VML), le format SVG (Scalable Vector Graphics) est une spécification depuis fin 2001. Comme son nom l'indique, il s'agit d'un format vectoriel en mode retenu. Ainsi, tous les graphiques définis sont stockés en mémoire autorisant leur manipulation ce qui constitue la première différence avec Canvas. Format déclaratif basé sur XML, SVG manipule des formes définies à l'aide de points et courbes permettant un redimensionnement sans perte de qualité. C'est le navigateur qui prend en charge le rendu des objets au mieux de ses capacités ce qui peut poser des problèmes dans des scénarios à fortes contraintes de performance.

De par son degré d'abstraction plus élevé, SVG se révèle plus adapté à la définition de formes complexes. En outre, l'interprétation des objets SVG conduit à la construction d'un arbre DOM spécifique ce qui rend ces objets accessibles à tout moment et permet d'écouter les évènements qu'ils génèrent ou d'y appliquer des styles CSS. Enfin, SVG offre l'avantage d'être manipulable facilement par les graphistes via des outils dédiés comme Inkscape. Face à tant de qualités, on peut se demander pourquoi SVG n'a pas réellement percé depuis sa standardisation. En effet, malgré un support de qualité sur mobiles, une présence forte dans la cartographie avec un acteur comme OpenStreetMap qui stocke ses cartes en SVG, preuve de sa pérennité, le format n'a pas conquis les développeurs web. La réponse peut surprendre mais SVG était sans doute trop en avance sur son époque avec ses possibilités impressionnantes qui ont pu effrayer les graphistes. En sus, l'absence de support par Internet Explorer a clairement joué en sa défaveur. Alors pourquoi aujourd'hui avec l'arrivée d'HTML 5 et son composant Canvas, SVG revient-il sur le devant de la scène ?

Jusqu'à présent, l'interaction entre SVG et HTML ne pouvait se faire que via un fichier SVG externe embarqué via des éléments tels que embed, object, iframe, img ou background-image en CSS. Ces solutions posaient des problèmes pour interagir de manière dynamique avec le contenu SVG importé obligeant bien souvent à réaliser ces opérations directement au sein du fichier externe. L'arrivée de la balise svg en HTML 5 permet de s'affranchir de ces contraintes en définissant directement au sein d'une page web un contenu SVG. Cette nouveauté ouvre la voie à une utilisation massive du format SVG à l'avenir en simplifiant les interactions avec son contenu depuis une page HTML que ce soit en Javascript ou en CSS.

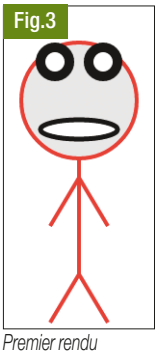
Premiers pas avec SVG

En HTML 5, un document SVG peut être défini directement dans une balise svg au sein d'une page web. Pour produire du SVG, il est possible de s'appuyer sur un éditeur graphique tel que Inkscape. Cette approche est plutôt orientée graphistes mais concerne surtout les formes les plus com-

plexes. Pour nos premiers pas avec SVG, une utilisation directe des balises est suffisante pour définir quelques formes :

```
<html>
<head>
<style>
 #head {
  stroke:red;
  stroke-width:4px;
  fill: #dedede;
 }
 .eye {
  stroke:black;
  stroke-width:10px;
  fill:white;
 }
 #mouth {
  stroke : black;
  stroke-width:5px;
  fill:white;
 }
 .body {
  stroke:red;
  stroke-width : 4px;
 }
</style>
</head>
<body>
 <svg width=»400» height=»400»>
  <circle cx=»120» cy=»70» r=»60» id=»head»/>
  <circle cx=»95» cy=»30» r=»15» class=»eye» />
  <circle cx=»145» cy=»30» r=»15» class=»eye» />
  <ellipse cx=»120» cy=»100» rx=»40» ry=»10» id=»mouth» />
  <line x1=»120» y1=»130» x2=»120» y2=»250» class=»body» />
  <line x1=»120» y1=»150» x2=»90» y2=»200» class=»body» />
  <line x1=»120» y1=»150» x2=»150» y2=»200» class=»body» />
  <line x1=»120» y1=»250» x2=»90» y2=»300» class=»body» />
  <line x1=»120» y1=»250» x2=»150» y2=»300» class=»body» />
 </svg>
</body>
</html>
```

Fig.3



Premier rendu SVG

Le rendu de ce code (figure 3) montre les possibilités de la balise svg et la facilité avec laquelle il est possible de personnaliser via CSS les différents objets définis au sein du SVG de la page HTML.

Carte avec SVG

La cartographie étant un domaine d'application phare de SVG, il existe un grand nombre de cartes au format SVG sur le Web ce qui facilite la création de cartes dans une application.

On peut également noter que des outils permettent de passer d'une carte SVG à du code générant le contenu équivalent via Canvas. Cette parenthèse faite, on choisit une carte SVG du continent Africain disponible ici : http://en.wikipedia.org/wiki/File:BlankMap-Africa.svg .

Au sein de ce document, seules les données relatives aux tracés des pays et du continent sont conservées. La définition d'une suite de points pour un pays est définie de la sorte :

```
<path id=»mayotte» d=»M338.0558,265.1932 C338.3414,264.8074
338.7038,263.9824  338.0558,263.7231  C337.8926,264.0609
337.8092,264.8734 338.0558,265.1932»/>
```

Au sein de l'élément path, l'attribut d déclare la position de départ et les différents points du tracé. Il contient des commandes (il en existe 10) et des points (x,y). La commande M positionne le point de départ du tracé. On peut également citer la commande C permettant de tracer une courbe de Bézier et la commande L pour tracer une ligne simple. Pour afficher la carte (figure 4), il reste ensuite à intégrer les parties définissant les chemins du fichier SVG récupéré au sein d'une page HTML :

```
<html>
<head>
<style type=»text/css»>
 .land{
  fill: #b9b9b9;
  stroke: white;
  stroke-width: 1.5;
  stroke-miterlimit: 4;
 }
 .coast{
  stroke-width: 0.5;
 }
</style>
</head>
<body>
 <svg width=»500» height=»500»>
  <path id=»mayotte» class=»land coast yt»
    d=»M338.0558,265.1932  C338.3414,264.8074  338.7038,263.
9824 338.0558,263.7231 C337.8926,264.0609 337.8092,264.8734
338.0558,265.1932»/>
  // ...
 </svg>
</body>
</html>
```

Interactivité avec SVG

Le rendu de la carte implémenté, la seconde étape concerne l'interactivité qui consiste à afficher sur la carte des données par pays en leur affectant des couleurs spécifiques. De plus, le survol d'un pays par la souris doit permettre d'afficher le nom du pays au sein d'une info-bulle. Alors que le rendu de la carte en SVG n'apporte comme bénéfice par rapport au Canvas qu'un redimensionnement sans perte de qualité, la partie interactivité va tourner en la faveur de SVG. En effet, chaque région étant définie via un objet path, elle devient accessible au sein du DOM par son id ou via une sélection par balise. Il est en outre possible de lui appliquer des styles CSS dynamiques ce qui permet de modifier l'opacité d'un pays au survol de la souris via la règle suivante :

Fig.4

Carte d'Afrique avec SVG

```
path:hover{opacity: 0.5;}
```

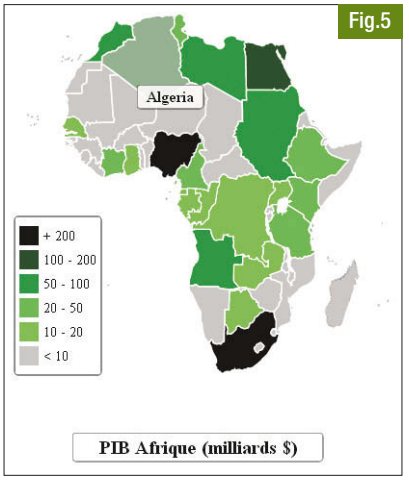
L'application de couleurs spécifiques aux différents pays suivant des données spécifiques ainsi que l'affichage du tooltip avec le nom du pays au survol de la souris sont ensuite implémentés au sein du code Javascript suivant :

```
function init(evt) {
 if ( window.svgDocument == null ) {
  svgDocument = evt.target.ownerDocument;
 }

 tooltip = svgDocument.getElementById('tooltip');
 tooltip_bg = svgDocument.getElementById('tooltip_bg');

 for (var i in ids) {
  elt = document.getElementById(ids[i]);
  elt.onmousemove = function (e) {
   showTooltip(e, capitalizeFirstLetter(e.currentTarget.id));
  };
  elt.onmouseout = function (e) {
   hideTooltip(e);
  };
 }
 colourCountries(data1);
}

function colourCountries(data) {
 for (var colour=0; colour<data.length; colour++) {
  for (var country=0; country<data[colour].length; country++) {
   colourCountry(data[colour][country], colour);
  }
 }
}

function colourCountry(name, colour) {
 var country = svgDocument.getElementById(name);
 var oldClass = country.getAttributeNS(null, 'class');
 var newClass = oldClass + ' colour' + colour;
 country.setAttributeNS(null, 'class', newClass);
}

function showTooltip(evt, mouseovertext) {
 tooltip.setAttributeNS(null,»x»,evt.clientX + 18);
 tooltip.setAttributeNS(null,»y»,evt.clientY + 32);
 tooltip.firstChild.data = mouseovertext;
 tooltip.setAttributeNS(null,»visibility»,»visible»);
 length = tooltip.getComputedTextLength();
 tooltip_bg.setAttributeNS(null,»width»,length + 20);
 tooltip_bg.setAttributeNS(null,»x»,evt.clientX + 8);
 tooltip_bg.setAttributeNS(null,»y»,evt.clientY + 14);
 tooltip_bg.setAttributeNS(null,»visibility»,»visibile»);
}

function hideTooltip(evt) {
 tooltip.setAttributeNS(null,»visibility»,»hidden»);
 tooltip_bg.setAttributeNS(null,»visibility»,»hidden»);
}
```

La méthode init est appelée au chargement du document svg via l'attribut onload de la balise éponyme. On y récupère les formes associées au tooltip à afficher, à savoir le texte et le rectangle d'arrière-plan. On installe ensuite les listeners d'évènements pour afficher le tooltip avec le nom du pays lors de son survol par la souris. Enfin, on colorie les pays en appliquant des styles CSS spécifiques (colour0 à colour5) suivant le contenu des données du tableau data1.

Le rendu final permet de constater la puissance de l'approche SVG avec une carte interactive parfaitement intégrée à la page HTML (figure 5).

Fig.5



Carte interactive avec SVG

Fonctionnant sur la majorité des navigateurs actuels, la carte interactive réalisée grâce à la technologie SVG posera des problèmes sur des versions antérieures de certains navigateurs tels que IE 8 par exemple qui reste attaché au format VML. Pour ce faire, des solutions de contournement existent et l'on citera ainsi la bibliothèque RaphaelJS qui permet de rendre du SVG en s'abstrayant de la définition du contenu via XML. Pour cela, elle propose une API programmable au sein de laquelle on définit les différents éléments SVG à rendre et c'est ensuite la bibliothèque qui se charge de produire le code nécessaire au bon rendu sur le navigateur cible en utilisant SVG lorsqu'il est supporté et une alternative lorsqu'il ne l'est pas.

Conclusion

Sous couvert de la réalisation d'une carte interactive en HTML 5, cet article aura mis en exergue les 2 technologies standard proposées aujourd'hui par le W3C pour réaliser des rendus graphiques au sein des navigateurs. L'implémentation de la carte via ces technologies aura servi de support à la comparaison des avantages et inconvénients des 2 approches tout en gardant à l'esprit que la cartographie reste un domaine dans lequel SVG excelle. A contrario, le composant SVG se révèlera supérieur dans des domaines temps réel où les contraintes de performance devront être maîtrisées au plus près. Le tableau de la figure 6 récapitule brièvement les différences entre ces 2 technologies. Quoi qu'il en soit, aucune des 2 technologies n'est fondamentalement meilleure en soit, et comme à l'habitude en informatique, il sera nécessaire de faire preuve de pragmatisme afin de choisir la solution la plus adaptée au contexte d'utilisation. Les développeurs pouvant s'estimer heureux d'avoir à disposition 2 technologies standard de cette qualité pour réaliser des rendus graphiques côte client.

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

Canvas	SVG
Bitmap (basé sur pixel)	Vectoriel (basé sur forme)
Unique élément HTML	Formes deviennent éléments du DOM
Modifié par script	Modifié par script et CSS
Interaction bas niveau basée sur coordonnées (x,y)	Interaction abstraite basée sur des formes
Performance meilleure sur petites surfaces, un grand nombre d'objets (> 10k)	Performance meilleure sur grandes surfaces, un petit nombre d'objets (< 10k)
Adapté jeux, graphiques boursiers temps réel, rendu d'images	Adapté cartographie, diagrammes, graphiques, rendu d'images

Fig.6 *Comparaison Canvas / SVG*

Gestion des erreurs en ASP.NET MVC

Lors de la conception d'une application web, gérer le comportement du système en cas d'erreur est important et ne devrait pas être laissé au hasard. D'une part, visualiser une page système peu accueillante, non stylisée, et loin des couleurs du site n'est pas toujours agréable pour l'utilisateur qui vient en plus de recevoir une erreur. D'autre part, si certains paramétrages ne sont pas effectués, l'utilisateur pourrait avoir accès aux détails techniques de l'exception.

Un visiteur mal intentionné pourrait se servir de ces détails pour accumuler des connaissances sur la structure technique du site et trouver plus facilement une faille de sécurité à exploiter. Ainsi, il appartient à l'équipe de réalisation du site de déterminer en amont le niveau de détails d'erreur à délivrer à l'utilisateur, les codes d'erreur HTTP à renvoyer en fonction des situations et les pages d'erreur qui seront affichées. En somme, il s'agit de décider d'un comportement applicatif en cas d'erreur qui soit uniformément suivi au cours des développements. ASP.NET MVC offre de nombreuses solutions pour gérer une exception de manière centralisée et rediriger vers une ressource en particulier. Gérer les erreurs peut donc se faire de plusieurs façons et peut s'adapter à divers scénarios. Ces différentes manières seront exposées au cours de cet article, du niveau le plus global à celui le plus fin.

Définir les url à appeler dans le fichier de configuration

La balise `customErrors` de la section `system.web` du fichier `web.config` situé à la racine du projet permet de définir des URLS de prise en charge lors d'exceptions :

```
<customErrors defaultRedirect=»~/Error/» mode=»On»>
  <error redirect=»~/Error/Index/404» statusCode=»404»/>
  <error redirect=»~/Error/Index/403» statusCode=»403»/>
</customErrors>
```

Il s'agit ici d'URLS correspondant à la table de routage suivante, présente par défaut lors de la création d'un projet :

```
routes.MapRoute(name: «Default»,
  url: «{controller}/{action}/{id}»,
  defaults: new { controller = «Home», action = «Index», id = Url
Parameter.Optional });
```

Par défaut, le mode de la balise `customErrors` ne possède pas la valeur `On` mais la valeur `RemoteOnly`. Ceci signifie que tous les appels au site en *localhost* peuvent accéder aux détails techniques de l'exception mais que les appels distants seront bien redirigés vers les URLS spécifiées. Dans la configuration ci-dessus, les exceptions de type `HttpException`, de code 404 et 403, possèdent un traitement spécial et des URLS de redirection spécifiques leur sont assignées par l'attribut `redirect`. Si le contrôleur et l'action spécifiés sont toujours les mêmes, `Error` et `Index`, un paramètre indiquant le code d'erreur est ajouté. Des paramètres supplémentaires auraient bien évidemment pu être spécifiés. Il aurait été bien sûr possible de rediriger vers des fichiers physiques, des webforms par exemple, d'extension *.aspx ou vers toute autre ressource. Un des attributs de la balise `customErrors`, `redirectMode`, permet de changer la façon dont la requête en erreur est traitée, il attend une valeur de l'énumération de type `CustomErrorsRedirectMode`. Par défaut, lorsque cet attribut n'est pas défini, il possède la valeur `ResponseRedirect`, c'est-à-dire que le serveur renvoie au client un code HTTP 302 indiquant par convention que la ressource demandée se trouve à une autre URL, spécifiée dans l'en-tête de réponse `Location`. La ressource spécifiée dans cet en-tête sera alors l'URL de la page d'erreur. Dans l'exemple ci-dessus, si l'application ASP.NET MVC était déployée sur un sous-site de nom « ErrorHandlingSample » et que l'exception rencontrée était une `HttpException` de code 404, alors l'en-tête de réponse `Location` possèderait la valeur « /ErrorHandlingSample/Error/Index/ 404?aspxerrorpath=/ErrorHandlingSample/ » . A partir de cet en-tête, le navigateur est alors invité à effectuer une seconde requête vers cette URL. Pour éviter une redirection, l'attribut `redirectMode` pourrait être affecté à une autre valeur : `ResponseRewrite`, ce qui équivaut à faire un appel à la méthode `Server.Transfer` disponible sur la classe `HttpServerUtility`, depuis la version 5 de IIS. Dès qu'une exception non gérée est rencontrée, la requête cesse d'être traitée par le gestionnaire HTTP courant pour être transférée, elle ainsi que toutes ses variables, vers une page aspx ou vers toute autre ressource statique. Ce mode n'est alors pas compatible avec le moteur de routes d'ASP.NET MVC; le chemin spécifié doit mener vers un fichier physique, la requête ne passera pas par le moteur de routage une nouvelle fois. De fait, si la valeur `ResponseRewrite` permet d'économiser une redirection client et donc une seconde requête, elle ne permet pas de passer de nouveau dans un contrôleur et une action spécifiques MVC comme avec le mode `ResponseRedirect`. Ainsi, cette gestion d'erreur via le `web.config` peut être utile à un niveau global et s'adapte facilement à des besoins simples :

- Aucun code n'est à écrire, un simple référencement d'URL en fonction des codes d'erreur HTTP est à effectuer dans le `web.config`,
- Cette méthode permet une redirection par défaut pour toutes les erreurs et une spécification d'URLS en fonction de certains codes HTTP,
- Le mode `RemoteOnly` permet aux développeurs ou aux administrateurs du serveur de voir les erreurs détaillées tandis que les utilisateurs ont automatiquement accès aux pages d'erreur spécifiées.

Cette méthode présente cependant certains inconvénients :

- Elle n'est pas compatible avec le paradigme ASP.NET MVC, à moins d'opérer une redirection et donc une deuxième requête,
- Le code de retour HTTP sera 200 par défaut, le code HTTP de l'erreur originelle n'est pas préservé, ou il faut l'affecter manuellement, dans la webform par exemple, pour que le navigateur reçoive un code d'erreur HTTP adapté,
- Lorsque le mode `redirectMode` est affecté à `ResponseRedirect`, il n'est plus possible d'accéder aux détails de l'exception et d'exposer la raison de l'erreur à l'utilisateur de manière plus détaillée,
- Lorsque l'attribut `redirectMode` est affecté à `ResponseRedirect`, un *querystring* est ajouté, « `?aspxerrorpath=` », pour indiquer la page à l'origine de l'erreur et il n'est pas possible de le retirer.

Ainsi, pour une gestion davantage personnalisée des erreurs, le développeur pourra se tourner vers une manière plus manuelle de gérer ses erreurs en s'abonnant à l'événement global d'erreur au niveau du fichier `global.asax` de l'application web. Si aucun mécanisme de gestion d'erreurs alternatif n'a été mis en place, il est fortement recommandé de ne pas désactiver les `customErrors`. En effet, sans mécanisme de repli, l'utilisateur risque d'avoir très facilement accès à tous les détails techniques des erreurs de l'application.

Gérer les erreurs au niveau du global.asax

Lorsqu'une exception non gérée se produit au niveau de l'application web, il est possible de la gérer dans le fichier `Global.asax.cs`, en y ajoutant cet *event handler* :

```
private void Application_Error(object sender, EventArgs e){}
```

Cette méthode sera alors appelée automatiquement en cas d'exception non gérée par l'application. Elle permet d'intercepter un maximum d'erreurs. Il peut s'agir d'exceptions qui se seraient produites dans les actions des contrôleurs mais aussi d'exceptions ayant lieu avant même qu'une méthode ne soit appelée sur un contrôleur, par exemple lorsqu'une route n'est pas trouvée dans la table de routage et ne peut être résolue, ou bien lorsque le *model binder* a rencontré un problème et n'a pu dé-sérialiser les bons paramètres. La méthode `Server.GetLastError` permet alors de récupérer les détails de l'exception. Après traitement, et pour s'assurer qu'aucun module ne prendra ensuite en charge cette erreur, il est préférable d'effectuer un appel à `Server.ClearError`. Ainsi, dans cette méthode, l'exception pourrait être analysée, et le contrôleur responsable de la gestion des erreurs pourrait être appelé en conséquence, avec les bons paramètres :

```
private void Application_Error(object sender, EventArgs e)
    {
      var exc = Server.GetLastError(); //obtenir l'exception d'origine
      int httpCode = 500; //code par défaut, le plus général possible
      if (exc is HttpException || exc.GetBaseException() is HttpException)
      {
        var httpException = exc as HttpException ?? exc.GetBase
Exception() as  HttpException;
        httpCode = httpException.GetHttpCode();
      }
      Request.RequestContext.RouteData.Values.Clear();
      Request.RequestContext.RouteData.Values.Add(«controller», «Error»);
      Request.RequestContext.RouteData.Values.Add(«action», «Index»);
      Request.RequestContext.RouteData.Values.Add(«id», httpCode);
      var controller = new ErrorController() as IController;
      controller.Execute(Request.RequestContext);
      Server.ClearError();
    }
```

Ci-dessus, l'action `Index` du contrôleur `Error` est exécutée et c'est elle qui a la responsabilité de délivrer une vue d'erreur à l'utilisateur. Le contrôleur `Error` contiendrait alors seulement :

```
public ActionResult Index(int? id)
    {
      var errorCode = id.GetValueOrDefault(500);
      ErrorModel model = GetModelAndSetStatusCodeFromErrorCode(errorCode);
      Response.StatusCode = errorCode;
      return View(model);
    }

    private ErrorModel GetModelAndSetStatusCodeFromErrorCode(int errorCodeIn)
    {
      var errorCode = 500;
      var message = «Une erreur interne est survenue.»;
      switch (errorCodeIn)
      {
        case 404:
          {
            message = «La ressource est introuvable»;
            errorCode = 404;
            break;
          }
        case 401:
          {
            message = «Vous n'êtes pas autorisé à accéder à cette
ressource, demandez les droits à l'administrateur»;
            errorCode = 401;
            break;
          }
        case 403:
          {
            message = «L'accès à cette ressource est interdit.»;
            errorCode = 403;
            break;
          }
      }
      Response.StatusCode = errorCode;
      return new ErrorModel(errorCode, message);
    }
```

Dans la méthode qui renvoie un modèle d'erreur, le code HTTP de la réponse est affecté. Une seule requête est donc effectuée, l'URL n'a pas changé, et le navigateur reçoit bien un code d'erreur adapté :

La gestion de l'exception aurait pu être encore plus fine, en faisant par exemple correspondre des codes d'erreur HTTP à des exceptions métier ou en passant d'autres paramètres à l'action Index, ou encore, en appelant des actions différentes qui renverraient des vues spécialisées pour chaque cas. Attention, le mécanisme de gestion d'erreur d'ASP.NET ne prend effet que lorsque la requête a été interceptée par le processus d'ASP.NET qui va tenter d'y répondre. Si l'utilisateur demande un simple fichier html, il sera directement pris en charge par IIS et son module `StaticFileModule`, qui renverra une page d'erreur système. Pour éviter toute page d'erreur système et délivrer une page d'erreur personnalisée, quelle que soit la requête cliente, il est possible de rajouter la ligne suivante dans le `web.config`, dans la section `webserver`, pour que tous les modules de code managé soient exécutés même si la requête arbore une extension qui ne concerne a priori pas de code managé :

```
<modules runAllManagedModulesForAllRequests=»true»></modules>
```

Cependant, ajouter cette ligne implique que pour toute requête, y compris des requêtes très simples visant une ressource basique accessible par un chemin physique sur le serveur, tous les modules managés seront exécutés afin de vérifier qu'aucun d'entre eux ne peut prendre la requête en charge. Le processus ASP.NET aura alors la main sur la requête. Mais cela est susceptible d'entraîner une perte de performances si de nombreuses requêtes qui ne nécessitent pas l'exécution de code managé sont exécutées. Le développeur peut alors se tourner vers d'autres solutions, en développant un *handler* HTTP personnalisé qui traiterait certaines extensions par exemple.

Gérer les erreurs avec des filtres d'erreur au niveau contrôleur

Bien que la marge de manœuvre offerte par la méthode ci-dessus soit assez large, lors d'une exception dans une action, le développeur peut vouloir récolter des informations uniquement disponibles dans le contexte ASP.NET MVC. Par exemple, il peut vouloir savoir si l'action était de type enfant (appelée dans le cadre d'une autre action, dite principale), afin de déterminer s'il faut renvoyer une vue d'erreur comprenant le *layout* ou une vue d'erreur partielle. Or, la méthode permettant de savoir s'il s'agit d'une action enfant est accessible depuis le contexte du contrôleur uniquement :

```
      ControllerContext.IsChildAction();
```

De nombreuses autres informations importantes ne sont accessibles que depuis le contexte du contrôleur. Ainsi, pour accéder au contexte du contrôleur

dans lequel l'exception s'est produite, un filtre d'erreur peut être appliqué à certaines actions en y apposant l'attribut, ou bien, à toutes les actions de l'application en enregistrant le filtre dans les `GlobalFilters`, dans la méthode `Application_Start` du `global.asax`. Par défaut, les modèles de projet ASP.NET MVC embarquent un filtre d'erreur nommé `HandleErrorAttribute`, ajouté aux filtres globaux de l'application, dans la classe `FilterConfig` :

```
public static void RegisterGlobalFilters(GlobalFilterCollection filters)
    {
      filters.Add(new HandleErrorAttribute());
    }
```

Il s'agit d'un attribut car il hérite de la classe `FilterAttribute` et il est dédié à la gestion des exceptions car il hérite de l'interface `IExceptionFilter` en implémentant la méthode `OnException`, appelée lorsqu'une exception non gérée se produit. L'implémentation offerte par `HandleErrorAttribute` ne gère que les exceptions HTTP de code 500, ne concernant pas les actions enfant. Si aucune vue n'est spécifiée lors de la déclaration de l'attribut, par défaut, l'attribut cherche une vue de nom `Error`, soit dans le dossier concernant le contrôleur courant, soit dans le dossier `Shared`. Le nom de la vue peut être spécifié comme suit :

```
   [HandleError(View = «MyErrorView» )]
 public ActionResult Index()
 {}
```

Le développeur peut cependant proposer une implémentation personnalisée en héritant de `HandleErrorAttribute` et en surchargeant sa méthode `virtual`, `OnException`. Il reçoit en paramètre un type intéressant : `ExceptionContext`, qui hérite de `ControllerContext`. Il s'agit donc bien de recueillir des informations concernant le contrôleur et l'action à l'origine de l'exception. De plus, ce paramètre permet directement, par certaines de ses propriétés accessibles en écriture, de gérer la réponse et d'indiquer un résultat via sa propriété `Result`, laquelle attend un type `ActionResult`. Avant tout traitement, il convient d'effacer la réponse précédemment construite, pour la remplir selon l'erreur :

```
filterContext.HttpContext.Response.Clear();
```

Via le paramètre `ExceptionContext` et une fois l'exception traitée et le résultat affecté, il est préférable de marquer explicitement l'exception comme gérée. De la sorte, si une gestion globale des erreurs existe au niveau du `global.asax`, celle-ci ne viendra pas interférer avec le traitement effectué par l'attribut. Le développeur peut également activer la propriété `TrySkipIisCustomErrors` sur la réponse `HTTP` pour s'assurer que le mécanisme des `customErrors` vu précédemment ne prenne pas le pas :

```
filterContext.ExceptionHandled = true;
 filterContext.HttpContext.Response.TrySkipIisCustomErrors = true;
```

Voici par exemple une implémentation de la méthode `OnException`. Dans cet exemple, le développeur vérifie s'il s'agit d'une requête Ajax ou d'une action enfant qui a causé l'exception pour savoir s'il doit retourner un `ViewResult`, c'est-à-dire une page entière avec le *layout* de base compris, ou une page d'erreur partielle, sans *layout*, qui viendrait s'insérer dans le DOM de la page web :

```
public override void OnException(ExceptionContext filterContext)
  {
    string controllerName = (string)filterContext.RouteData.Values
[«controller»];
    string actionName = (string)filterContext.RouteData.Values
[«action»];
    HandleErrorInfo model = new HandleErrorInfo(filterContext.
Exception, controllerName, actionName);
    var viewName = «Error»;
    var viewData = new ViewDataDictionary<HandleErrorInfo>(model);
    var tempData = filterContext.Controller.TempData;
    if (filterContext.IsChildAction || filterContext.HttpContext.
Request.IsAjaxRequest())
    {
      filterContext.Result = new PartialViewResult
      {
        ViewName = viewName,
        ViewData = viewData,
        TempData = tempData
      };
    }
    else
    {
      filterContext.Result = new ViewResult
      {
        ViewName = viewName,
        MasterName = Master,
        ViewData = viewData,
        TempData = tempData
      };
    }
    int httpCode = 500; //code par défaut, le plus général possible
    var exception = filterContext.Exception;
    if (exception is HttpException || exception.GetBaseException
() is HttpException)
    {
      var httpException = exception as HttpException ?? exception.Get
BaseException() as HttpException;
      httpCode = httpException.GetHttpCode();
    }
    filterContext.HttpContext.Response.Clear();
    filterContext.ExceptionHandled = true;
    filterContext.HttpContext.Response.StatusCode = httpCode;
    filterContext.HttpContext.Response.TrySkipIisCustomErrors = true;
  }
```

Tout comme pour la gestion des erreurs dans le `global.asax`, une seule requête est effectuée et un code d'erreur HTTP adapté peut être renvoyé. Les deux systèmes restent bien sûr totalement compatibles et collaborent efficacement. Cet article a ainsi introduit plusieurs manières de gérer les erreurs en ASP.NET MVC. Elles ne sont pas exclusives les unes des autres et peuvent au contraire s'avérer complémentaires. Tandis que l'attribut d'erreur personnalisé, apposé globalement à toutes les actions, ou visant certaines actions en particulier, permet une gestion plus fine de l'exception et de la réponse à l'utilisateur, l'événement `Application_Error` du `global.asax` permet de ne laisser passer aucune exception non gérée, dans le cadre d'ASP.NET, et s'adresse aux développeurs soucieux d'avoir un contrôle total sur les réponses effectuées au client en cas d'erreur. Par ailleurs, cela permet de logguer facilement les erreurs ou de tracer certaines informations précises. Pour ceux qui attendent au contraire un moyen simple et rapide de fournir des pages d'erreurs statiques et sans complexité, le système des `customErrors` du `web.config`, notamment en mode `RewriteResponse`, peut s'avérer idéal.

Anna **Yafi** – *Consultante Infinite Square*
ayafi@infinitesquare.com

INFINITE SQUARE

Développement de sites Internet et référencement naturel

« Mon site est valide W3C, c'est bon il sera bien référencé ». Voici une phrase que j'ai entendue et revue récemment sur le site « Webagencyfail. ». Heureusement cela reste anecdotique, mais il est intéressant de voir que sur le fond, ou sur la forme, le référencement ou SEO (Search Engine Optimisation) reste méconnu par ceux qui développent les sites Internet. Dommage, leurs créations seront invisibles !

Je vais donc répondre à cette question sans plus tarder : avoir un site valide W3C n'a que très peu d'impact pour le SEO (optimisation pour les moteurs de recherche). Cela reste cependant un bon indicateur pour connaitre la « maturité » d'un site, savoir s'il a été créé et entretenu avec soin.

L'expérience nous montre d'ailleurs qu'avoir un site avec une grande volumétrie de pages et compatible W3C est pratiquement impossible. Des oublis ponctuels ou des intégrations de modules suffisent à lever des avertissements et des erreurs. Le HTML, étant un langage de présentation, nous permet de voir facilement lorsqu'une balise n'est pas à sa place ou est mal fermée. Attention cependant à l'interprétation faite par les navigateurs; ils corrigent désormais les erreurs mineures afin de restituer aux internautes un ensemble cohérent. Faire une vérification du code source des pages, avant mise en production, n'est pas une option. Le HTML actuel reste stricte pour être parfait. L'arrivée de HTML5 va cependant le rendre plus flexible, moins contraignant.

Ayant un passé de développeur (plus ou moins actif mais cela reste mon premier amour), je me suis rendu compte qu'il y avait une certaine méconnaissance du SEO dans mon entourage proche, parmi mes amis qui font des lignes de code.

Détrompez-vous le référencement n'est pas du « Marabout », du « travail supplémentaire, sans but », c'est un passage (parfois douloureux) nécessaire ! Le référencement devrait idéalement être réfléchi en amont dans la chaîne de production, être intégré au cahier des charges. Faire un site fonctionnel c'est bien, mais si personne ne le visite, il perd de son utilité.

Rassurez-vous les bonnes pratiques du SEO sont simples à appréhender, nous allons aborder ensemble les différents points qui influent dans le référencement d'un site.

Le référencement, une histoire de visibilité

Que vous travailliez sur un site e-commerce ou sur un site de presse vivant de la publicité, il va avoir besoin de trafic pour vivre. Un site Internet est un moyen d'information ou de service.

Les moteurs de recherche sont là pour faciliter aux internautes l'accès à l'information, que ce soit celle de votre site ou celle de votre concurrent. Les moteurs de recherche proposent deux types de mise en avant pour chaque site :

- les liens commerciaux (Adwords dans le cas de Google), liés au SEA (publicité sur les moteurs de recherche)
- les liens organiques (ou naturels), liés au SEO

Google proposent en effet aux sites d'acheter leur visibilité au travers de sa régie intégrée Adwords. Si vous souhaitez apparaître lorsqu'un internaute fait une recherche, vous pouvez acheter votre présence. Le système d'enchère a très bien été pensé par Google : dans un secteur d'activité très concurrentiel, le coût s'envole. Obtenir un visiteur sur des requêtes liées à la banque peut coûter près de 40 € avec ce système ! La facturation se fait heureusement au clic (CPC). Vous ne payez que si un utilisateur clique sur votre annonce.

Heureusement pour le SEO (qui offre une visibilité gratuite), la présence des affichages sponsorisés est limitée dans les moteurs de recherche. Faire du SEA permet cependant d'obtenir les premières places, les plus convoitées.

Le référencement naturel reste malgré cette concurrence la source de trafic principale (> 60%) pour une grande majorité des sites. Dans ce métier il ne suffit cependant pas d'acheter des mots clés. Le SEO repose sur de nombreux facteurs.

Le SEO, au cœur de nombreuses synergies

Une multitude de critères sont pris en compte pour classer les différents sites (plus de 200). Nous pouvons les classer en trois grands axes : la structure, le contenu et la popularité Fig.1 et Fig.2.

La structure

La structure d'un site reste sa base et est donc à travailler avec soin. Nous incluons dans cette partie toutes les recommandations dites « techniques », incompréhensibles au commun des mortels. Par chance cette partie vous concerne directement, réfléchissez avant de l'aborder, vous n'aurez plus d'excuses !

Au plus bas niveau d'un site nous trouvons le nom de domaine. Son ancienneté influe sur le référencement, le saviez-vous ? Un site ancien ou un nom de domaine réservé pour de nombreuses années envoie un message positif aux moteurs : ce site est fait pour durer dans le temps, vous devriez le considérer avec intérêt.

L'hébergement, ensuite, a bien entendu lui aussi son rôle à jouer. Les moteurs, lorsqu'ils explorent un site (via leurs robots, appelés crawlers), lui accordent un temps d'attention limité.

Fig.1

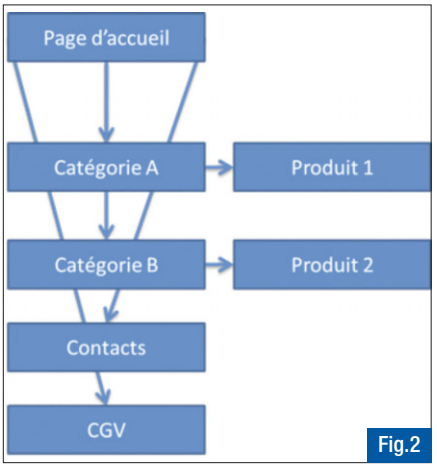


Fig.2

Sur des sites de taille importante, toutes les pages ne sont pas visitées par les moteurs. Malheureusement les pages ignorées à ce stade ne pourront pas être exposées aux internautes. Posséder un serveur rapide (avec un faible temps de réponse) permet d'envoyer davantage d'informations, d'avoir davantage de pages accessibles au travers des moteurs de recherche. De plus, si des incidents occasionnels ne nuisent pas au SEO, un seuil de tolérance à la panne existe. Un bon hébergement doit répondre, et répondre vite !

Puis à un niveau plus haut, nous avons l'arborescence du site qui est un élément important.

Les critères sont les mêmes que ceux utilisés par les sites e-commerce, comme le nombre de clics avant l'achat. Par exemple avoir un lien depuis la page d'accueil directement vers le produit phare indique que ce produit est important aux utilisateurs, mais aussi aux moteurs.

Et pour finir, nous avons la mise en page du contenu à travers le HTML. Il existe des balises permettant de mieux faire comprendre aux moteurs le contenu de notre site. Nous avons pour commencer la balise « <title> » et la balise « <H1> » permettant d'indiquer le titre de la page. Ces éléments sont ensuite repris lors de l'indexation pour résumer la page autour de mots clefs. On peut également citer les extraits enrichis, avec différents formats possibles : les micro-données, les micro-formats et le RDFa. Il en existe pour les avis, les auteurs, les recettes ou encore les événements. En balisant vos types de contenu, vous donnerez un meilleur aperçu de vos pages sur les résultats de recherche Google. Vous pourrez vous différencier de vos concurrents et augmenter votre taux de clics Fig.3.

Le contenu Fig.4.

Éléments roi dans le référencement, car sans contenu, il est très difficile d'apparaitre dans les moteurs de recherche.

L'idéal est bien le contenu sous format texte, car les moteurs savent très bien l'interpréter, mais d'autres formes peuvent être utiles pour le référencement, comme la vidéo, les images.

Ensuite, il y a une différence dans l'approche éditoriale suivant les médias utilisés, typiquement le titre d'un article papier sera travaillé pour être le plus attractif possible, avec une belle image de présentation. A la différence du média Internet, où le titre sera plus informatif, car les utilisateurs "scannent" l'information pour ensuite la consommer.

Fig.4 © D.R.

Un intérêt de travailler ces titres de façon plus informative est celui d'être retrouvé par les utilisateurs des moteurs de recherche.

Par exemple, il sera très dur de retrouver un article avec comme titre : « La saga de l'or noir se poursuit au pays de l'or vert » à la place de « Problème pétrolier au Brésil ».

La popularité Fig.5.

Le dernier axe est celui de la popularité, ou dans le cas de Google le « Page Rank ». Pourquoi la popularité est un point important, Google est parti de ce simple constat :

Vous recherchez un plombier, la première approche consiste à rechercher dans un annuaire, mais vous avez une multitude de possibilités et peu de critères de différenciation.

La seconde est de demander à votre entourage si quelqu'un connait un bon garagiste.

Le système de Page Rank est basé sur la seconde approche. En effet, lorsqu'un site fait un lien vers un autre site, Google l'interprète comme un vote.

Le Page Rank correspond aux nombres de liens qui pointent vers une page, elle varie entre 0 et 10 et les tranches pour atteindre un nouveau palier sont exponentielles.

Cela signifie qu'il faut par exemple 100 liens pour avoir un Page Rank de 1. En revanche, il ne faut pas 200 liens pour atteindre le Page Rank de 2, mais plutôt 1000 liens, etc.

Il n'y a que très peu de domaines ayant des Page Rank de 10. Il y a google.com, adobe.com, microsoft.com, mais ensuite les plus hauts sont à 9, 8. Il faut savoir qu'un Page Rank moyen est dans les alentours de 3.

Le profil d'un bon référenceur

Le monde du référencement est composé de personnes généralement hétéroclites, il n'y a que très peu d'écoles qui enseignent ce métier. Cela s'explique, car c'est un métier assez jeune comparativement aux métiers en dehors d'Internet.

Par contre, ils ont tous un point commun, c'est celui d'être curieux.

En effet, même s'il est impossible de connaitre l'ensemble des critères utilisés par les moteurs de recherche, l'idée est d'en connaitre suffisamment et d'utiliser ceux qui fonctionnent.

L'idée est donc de faire du « reverse engineering » contre des entreprises qui emploient généralement des talents du monde entier.

Même faire du « Social Engineering » en dehors des États-Unis est très difficile, car dans la plupart des cas il n'y a pas d'équipe locale de développement sur les parties cœurs.

Ce qui est amusant, car c'est le jeu du chat et de la souris entre les ingénieurs des moteurs et des référenceurs : chacun souhaiterait savoir ce que fait l'autre.

Benjamin **Bussière** et l'équipe SEO de LSF interactive

Fig.3

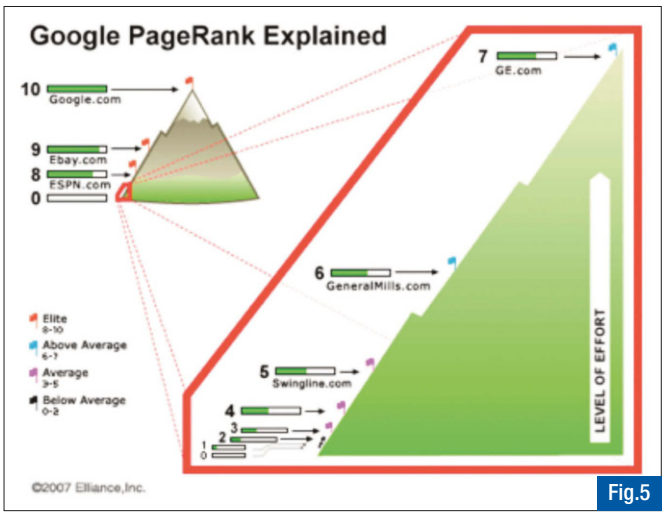


Fig.5

Les sites « one-page », tendance passagère ou véritable orientation du Web en 2014 ?

Le web a toujours été ponctué par diverses tendances, qu'elles soient graphiques ou techniques. Certains grands acteurs du web innovent dans ce sens, insufflant généralement de nouvelles inspirations pour des milliers de webdesigners, intégrateurs et développeurs. Si la création web en 2013 a été marquée par une tendance, c'est bien celle du grand retour des sites « one-page » ou « single-page ».

L'idée est de regrouper l'ensemble de son contenu dans une seule et unique page. Nous verrons dans cet article quels sont les principaux intérêts de ces sites, pour quels types de projets ils sont les plus adaptés, mais également quelles contraintes prendre en compte d'un point de vue technique sans négliger les objectifs de référencement.

DESIGN, ERGONOMIE ET MARKETING

Un site compréhensible

Un site one-page, c'est avant tout un site qui se veut simple, clair et efficace. Cela s'inscrit dans une tendance de simplification que l'on observe sur le web depuis plusieurs années, à l'image de la communication de Google. Aérer le contenu, faciliter la compréhension du message, en clair se débarrasser des fioritures.

Quoi de mieux pour faciliter la compréhension du message que de raconter une histoire ? C'est ce que l'on appelle le storytelling. Cette étape consiste à définir la hiérarchisation du contenu, à savoir ce que l'on souhaite que l'internaute voie en priorité, et de le scénariser. Et c'est là que le site « one-page » prend tout son intérêt... En effet, ce type de site se compose généralement d'une longue page divisée en sections. Chaque section correspond à une thématique, ou à une étape de notre histoire. L'internaute décide de passer de l'une à l'autre par le scroll-down. Il devient donc acteur, ou plutôt spectateur omniscient, de l'histoire qui lui est racontée. La barre de scroll représente ainsi une ligne de temps virtuelle, contrôlée par le visiteur. Il n'a qu'à suivre à son rythme un chemin tout tracé vers l'appel à l'action. On voit alors clairement l'attrait marketing de ce type de schéma : le corps de la page est consacré au discours commercial, menant en bas de page à la démarche d'action (collecte d'email, processus d'achat ou de contact...) Fig.1.

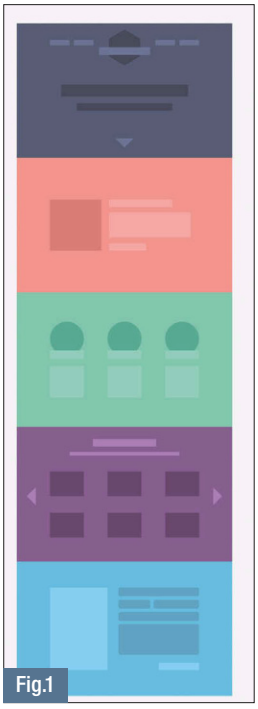


Fig.1

Une expérience utilisateur sensible

Investir l'internaute dans l'histoire suscite un réel intérêt ludique pour ce dernier. C'est l'expérience utilisateur qui s'en trouve enrichie. Cela passe par l'interaction, mais également par le vocabulaire et le ton employés. En s'adressant directement à l'internaute, on peut apporter une réelle personnalité à un site web, et ainsi instaurer un rapport plus humain avec l'utilisateur. Un internaute qui passe un bon moment sur votre site gardera une bonne opinion de votre marque, et y reviendra potentiellement.

L'ergonomie joue également un rôle primordial dans l'expérience utilisateur, et là aussi les sites « one-page » tirent leur épingle du jeu. En effet, tout le contenu étant présent sur une page, il sera chargé une seule fois, offrant un confort de navigation non négligeable. Finis les temps de chargement interminables entre chaque page pour accéder à un contenu spécifique. Nous le verrons plus loin, pour que cela devienne un réel avantage, il faut s'efforcer d'optimiser au maximum le chargement initial.

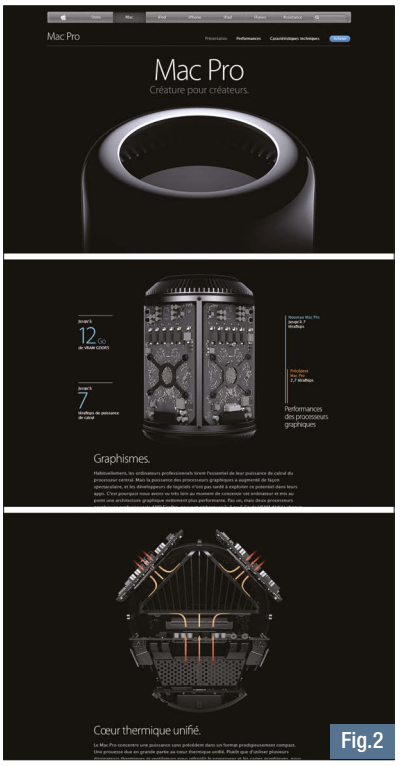


Fig.2

Un autre aspect ergonomique important avec ce type de mise en page : la quantité d'informations affichées à l'écran (au-dessus de la ligne de flottaison). On considère ainsi qu'au premier coup d'œil l'internaute lit moins, mais lit mieux. L'information est davantage aérée, et catégorisée en sections. Chaque sujet est traité en son temps, évitant de noyer le visiteur sous une avalanche d'informations. Mais l'intérêt de la page d'accueil n'est-il pas de donner à l'internaute un aperçu global du site qu'il visite ? Et s'il ne descend pas jusqu'au bas de page ? Il appartient au webdesigner de guider l'internaute afin de rendre sa navigation la plus aisée et fluide possible. Cela peut par exemple passer par un menu dédié composé d'ancres HTML, qui agirait comme un fil d'Ariane.

L'expérience utilisateur passe enfin par le design, et en matière de design, les sites « one-page » sont loin d'être les derniers. Un espace d'expression infini (ou presque), un moyen d'interaction universel à savoir le scroll, du pain béni pour un graphiste qui le pousse à s'éloigner de la structure traditionnelle d'un site classique.

Pour la plupart des sites monopages, l'image est mise à l'honneur, souvent traitée en plein écran. L'interaction se fait par le scroll, et permet de déclencher l'apparition de blocs ou d'animations à une hauteur définie, de jouer sur des effets de profondeur avec parallaxe ou de 3d avec les manipulations canvas... Un des exemples les plus réussis dans ce domaine reste la communication du géant Apple lors de la sortie de son nouveau Mac Pro (http://www.apple.com/fr/macpro).

On se souvient aussi du site Nike Better World, l'un des premiers à pleinement exploiter l'effet parallaxe. Des exemples de mariages réussis de simplicité et d'effet Wow. L'effet Wow oui, car c'est bien de cela que nous parlons. Savoir provoquer surprise et émotions, en cherchant à rendre les interfaces plus sensibles Fig.2.

Moi aussi j'en veux un !

Nous l'avons vu, le site « one-page » peut se révéler être un excellent « objet marketing ». Il peut également être un moyen pour une entité de se démarquer de ses concurrents par une expérience utilisateur aboutie. Alors, aurait-on trouvé la recette idéale pour un site impactant ? Non bien sûr, car on le sait, en matière de création web il n'existe pas de recette miracle. Nous nous adaptons à chaque client, à son contenu, à sa cible et au message qu'il souhaite véhiculer. Il nous incombe de lui proposer les choix technologiques les plus adaptés à son projet. En règle générale, on considère que le site « one-page » est réservé aux sites vitrines, promotionnels au contenu peu abondant. Il faut bien garder à l'esprit que ce type de sites bouscule les habitudes des internautes, il s'agit donc d'une réelle démarche qui se doit d'être motivée par une réflexion à la fois du point de vue du design, mais également en ergonomie. Mais alors, si votre client vous réclame un « one-page » pour son site e-commerce, que faire ? Soyez habile, détournez et remaniez pour vous adapter à vos projets. Pourquoi ne pas imaginer une page d'accueil basée sur un modèle « one-page » avec des call-to-actions vers vos pages internes à chaque étape ? Car au-delà du site constitué d'une page unique, le « one-page » reflète une autre tendance diabolisée pendant des années : les pages « à rallonge ». On a en effet longtemps considéré que le clic était plus naturel que le scroll pour l'internaute, et pourtant... Le clic est bien souvent synonyme de temps de chargement, rebutant pour l'internaute qui, c'est bien connu, veut tout, tout de suite. Mais le retour en force des blogs et l'émergence des applications web telles que Facebook ou Twitter nous ont bien montré que le scroll n'avait pas dit son dernier mot...

ASPECTS TECHNIQUES

Trop de scroll tue le scroll ?

Il est vrai que le scroll effraie de moins en moins, mais soyons honnête, il peut vite devenir désagréable de devoir en user à outrance. Le risque est que l'internaute n'ait pas le courage de descendre l'ensemble de votre page, et rate donc le précieux appel à l'action. Heureusement il existe des solutions afin d'améliorer l'ergonomie : il faut voir votre page comme un grand slideshow. Le plugin « fullPage.js » basé sur les librairies jQuery et jQuery UI permet de créer simplement une page responsive, avec des transitions, le tout compatible tous navigateurs, mais également sur tablettes ou mobiles (source : http://alvarotrigo.com/fullPage/). Chaque section de votre page représente un slide qui s'adapte aux dimensions de la fenêtre, et la navigation se fait par le scroll ou par les flèches du clavier. Du côté du scroll, un « cran » de molette suffit à passer d'un slide à l'autre. Pour ce qui est des flèches de navigation, leur usage reste beaucoup moins répandu sur le web, et donc moins évident pour l'utilisateur. Ce type de navigation permet cependant de naviguer sur deux axes, et donc d'imaginer une structure de page bien plus complexe. Au niveau du code, le plugin s'installe très simplement.

On commence par inclure les fichiers javascript et la css :

```
<script  src=»http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js»></script>
<script src=»http://ajax.googleapis.com/ajax/libs/jqueryui/1.9.1/jquery-ui.min.js»></script>
<script type=»text/javascript» src=»jquery.fullPage.js»></script>
<link  rel=»stylesheet»  type=»text/css»  href=»jquery.fullPage.css» />
```

Ensuite chaque section est définie par une div avec la class « section » :

```
<div class=»section»>Some section</div>
<div class=»section»>Some section</div>
<div class=»section»>Some section</div>
```

Il est possible de créer des slides horizontaux en divisant une section en div avec la class « slide » :

```
<div class=»section»>
    <div class=»slide»> Slide 1 </div>
    <div class=»slide»> Slide 2 </div>
    <div class=»slide»> Slide 3 </div>
</div>
```

Il suffit enfin d'initialiser le slider :

```
$(document).ready(function() {
    $.fn.fullpage();
});
```

Beaucoup de paramètres permettent de configurer les animations, d'ajouter un menu basé sur les ancres HTML, ou encore d'ajouter l'ancre active dans l'url, permettant d'utiliser les fonctions « précédant / suivant » du navigateur. Attention toutefois à cette méthode, elle exploite en fait une fonctionnalité du navigateur et empêche l'indexation par google. On préfèrera utiliser la méthode URL hash avec la librairie jQuery BBQ par exemple. L'atout de ce plugin est qu'à lui seul, il parvient à régler plusieurs problèmes constatés sur un site mono-page. Il permet de charger le contenu au fur et à mesure du scroll, de contrôler les évènements souris et les interactions tactiles, il inclut également le support des transitions css3 pour les anciens navigateurs. Un bémol tout de même, il ne permet pas de créer de sections plus hautes que la hauteur de la fenêtre, les slides s'adaptant automatiquement à sa taille. Il aurait pu être intéressant de pouvoir varier les hauteurs, d'alterner entre scroll classique et accéléré... Côté code, c'est surtout en javascript que tout se passe. Pour détecter l'évènement souris on utilise le plugin jQuery Mouse Wheel Plugin qui permet l'utilisation de l'évènement “mousewheel” :

```
$(window).on('mousewheel', function(event) {
  console.log(event.deltaX, event.deltaY, event.deltaFactor);
});
```

On détecte ensuite si l'utilisateur scrolle vers le haut ou vers le bas pour effectuer le déplacement. En ce qui concerne le responsive design, le plugin passe là aussi par du javascript. Il est étonnant d'ailleurs de ne pas utiliser le css3 et ses media queries. Voici par exemple comment sont redimensionnées les tailles des textes :

```
function resizeMe(displayHeight, displayWidth) {
  // Hauteur standard, pour laquelle la taille de texte est de 100%
  var preferredHeight = 825;
  var windowSize = displayHeight;

  if (displayHeight < 825 || displayWidth < 900) {
    if (displayWidth < 900) {
      windowSize = displayWidth;
      preferredHeight = 900;
    }
    var percentage = (windowSize * 100) / preferredHeight;
    var newFontSize = percentage.toFixed(2);

    $(«body»).css(«font-size», newFontSize + '%');
  } else {
    $(«body»).css(«font-size», '100%');
  }
}
```

Côté css, les sections sont fluides, positionnées en relative :

```
.section {
    width: 100%;
    position: relative;
    -webkit-box-sizing: border-box;
    -moz-box-sizing: border-box;
    box-sizing: border-box;
}
```

Pour les animations, on utilise les transitions css3 :

```
.slides {
    height: 100%;
    overflow: hidden;
    position: relative;
    -webkit-transition: all 0.3s ease-out;
    -moz-transition: all 0.3s ease-out;
    -o-transition: all 0.3s ease-out;
    transition: all 0.3s ease-out;
}
```

Le scroll comme timeline

Une autre approche du site mono-page est de considérer la barre de scroll comme ligne de temps pour déclencher une animation. Ce principe est très efficace pour créer un lien entre les sections et pousser à continuer sa lecture. Chanel a lancé au printemps 2013 un site dédié à sa nouvelle montre : http://embedxppremiere.chanel.com/watch/premiere/embed/experience/. La marque exploite pleinement le scroll avec un cadrant de montre qui s'anime, auquel viennent se greffer les pièces constituant le bijou en slow-motion...

Une belle expérience de scroll avec une réflexion autour du temps. Pour réaliser ce type d'effet, on détecte là encore le scroll pour déclencher un déplacement en javascript, ou une transition css. Une solution simple à mettre en place consiste à ajouter une classe au body de votre page, à une hauteur définie de scroll :

```
[visuel : capture-site-chanel.tif]
$(document).ready(function() {
  $(window).on(«scroll», function() {
    var fromTop = $(window).scrollTop();
    $(«body»).toggleClass(«down», (fromTop > 300));
  });
});
```

Il suffit ensuite adapter la css lorsque la class est active :

```
.section {
    position: absolute;
    left: 100%;
    -webkit-transition: all 0.3s ease-out;
    -moz-transition: all 0.3s ease-out;
    -o-transition: all 0.3s ease-out;
    transition: all 0.3s ease-out;
}
.down .section {
    left: 0;
    -ms-transform: rotate(3deg);
    -webkit-transform: rotate(3deg);
    -o-transform: rotate(3deg);
    transform: rotate(3deg);
}
```

Ici, la div se déplace vers la droite en subissant une rotation de 3 degrés lorsque l'utilisateur a scrollé une hauteur de 300 pixels. Pour aller plus loin, il va falloir à nouveau passer par du javascript, avec jQuery Srcoll Path par exemple (source : https://github.com/JoelBesada/scrollpath). Ce plugin permet de définir un chemin de navigation personnalisé. Il utilise une syntaxe similaire à celle des canvas et permet de dessiner le chemin avec des directives telles que moveTo ou lineTo. Des effets tels que défilement, rotation, transformation... sont possibles.

Les solutions alternatives

De nouveaux services web proposent désormais de créer son site « one-page » en quelques clics. C'est de cas de ScrollKit, Strikingly ou OnePager par exemple (sources : https://www.scrollkit.com/, https://www.strikingly.com/, http://onepagerapp.com/). Dans une interface simple et épurée, vous créez votre page comme sous Photoshop, et le code est généré. Pas besoin de compétences particulières donc, mais des possibilités d'animations limitées et une qualité de code très moyenne. On trouve par exemple 41 erreurs sur la page d'accueil de ScrollKit au validateur w3c... Ces solutions clés en main ont l'intérêt d'être accessibles à tous, mais leur qualité et leur optimisation restent douteuses.

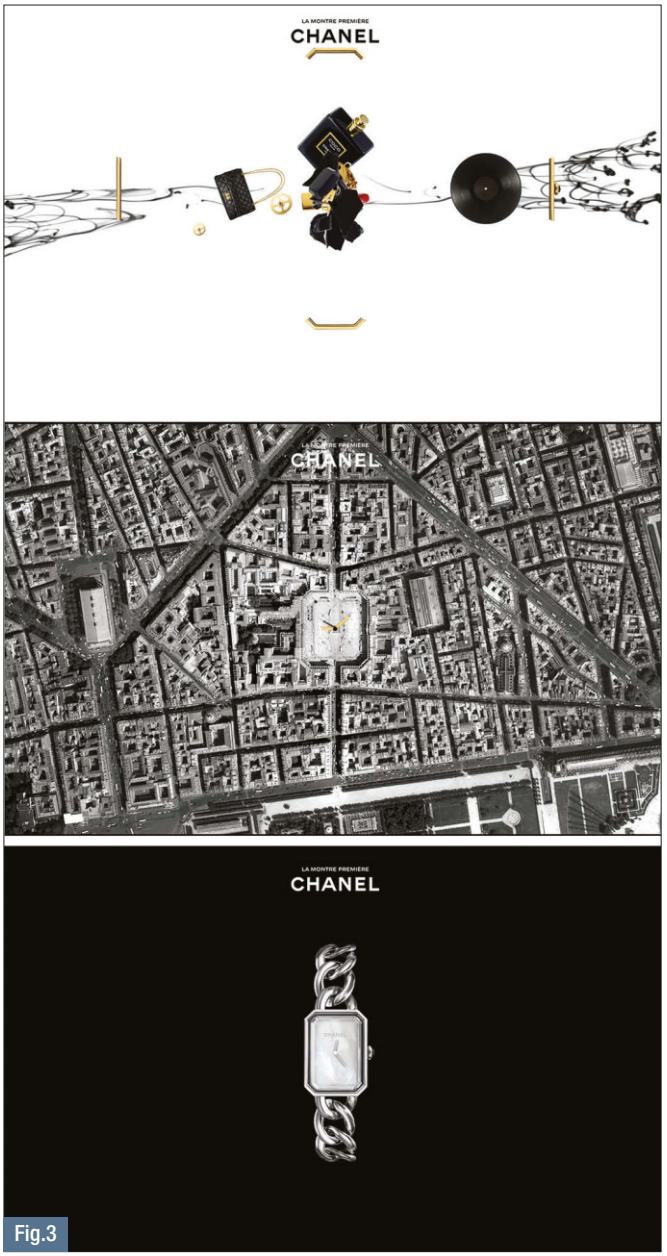


Fig.3

RÉFÉRENCEMENT

Une seule page pour Google

Pour ce qui est du référencement, les sites « one-page » partent avec un sérieux handicap : ils n'ont qu'une page à indexer, ils sont donc moins bien optimisés que les sites multipages au niveau du balisage sémantique. Un seul h1, un seul title et un ou deux mots clés... c'est un peu juste. Quelles sont alors les solutions à mettre en place pour contrebalancer ? Nous apportons ici quelques éléments de réponse.

Une page de démo a été créée (par John Mueller, Maile Ohy, et Joachim Kupke) pour expliquer comment indiquer aux robots la façon de bien indexer la totalité du contenu d'un site mono-page, en prenant en compte toutes les informations, sans duplicate content (source : http://googlewebmastercentral.blogspot.fr/2014/02/infinite-scroll-search-friendly.html). On retiendra qu'il faut configurer une url unique pour chaque section :

- Good : example.com/category?name=fun-items&page=1
- Good : example.com/fun-items?lastid=567
- Less optimal : example.com/fun-items#1

Il est également conseillé de placer la navigation dans le <head> de la page avec les balises <link rel> :

```
<link rel=»next» href=»/items?page=11»>
<link rel=»prev» href=»/items?page=9»>
```

Le blog

Le meilleur moyen de se positionner sur davantage de mots clés et de s'assurer un bon référencement reste le blog. Hébergé sur le même domaine, il va permettre de produire plus de contenu et de cibler des requêtes spécifiques. C'est le concept de Longue Traîne : cibler les mots clés moins courants, mais qui représentent un volume de recherche relativement important. La concurrence étant moins rude sur ces mots clés, il sera plus facile de se positionner. D'où l'importance de traiter les tendances pour être positionné sur des requêtes qui répondent à une question ou un besoin immédiat, avec en bonus un trafic qualifié.

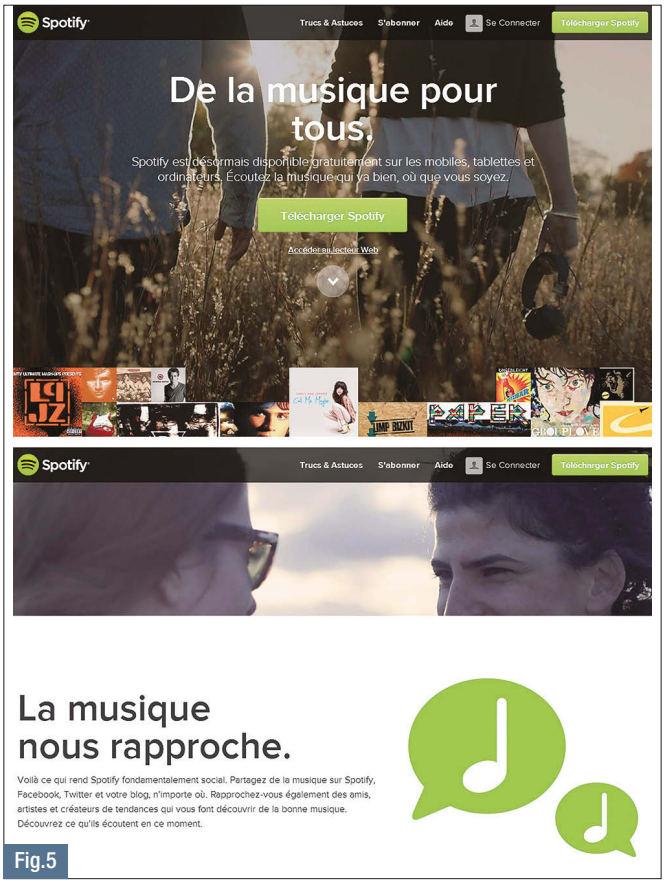


Fig.5

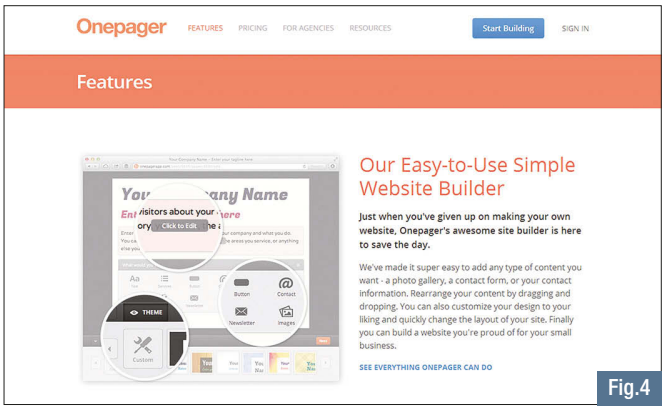


Fig.4

Réseaux sociaux

Pour obtenir de bons résultats, il faut donc paraître crédible aux yeux de google, et cela passe également par une présence active sur les réseaux sociaux. Google a tendance à favoriser les grandes marques, et un des meilleurs moyens de démontrer sa valeur est d'être présent et d'échanger avec une communauté qualitative et engagée.

Pensez également à identifier l'auteur des articles. Lier son blog à un profil Google+ est un moyen de prouver à google que l'on est un auteur actif, et apporte du crédit à votre site. Il semblerait que la prochaine évolution majeure du référencement Google soit l'Author Rank : les informations vérifiées liées à des profils en ligne bénéficieront d'un meilleur positionnement que les autres.

Des pages internes ?

Il est toujours possible d'ajouter des pages internes tout en conservant un principe de « one-page » en page d'accueil. Nous l'avons vu, ce type de page présente un sérieux intérêt en termes de communication et d'ergonomie. Il serait dommage de devoir s'en priver pour des questions d'architecture de site ou de référencement. Mais rien d'empêche de créer quelques pages annexes traitant de sujets particuliers. Le site de Spotify est construit de cette manière, et il compte parmi les sites musicaux les mieux référencés (source : https://www.spotify.com/fr/). La page d'accueil reste la plus influente, et le travail de référencement sur les pages internes se concentre sur des requêtes plus spécifiques, avec du contenu qualitatif. S'il y a une règle à retenir en SEO, c'est de travailler du contenu optimisé, de qualité.

CONCLUSION

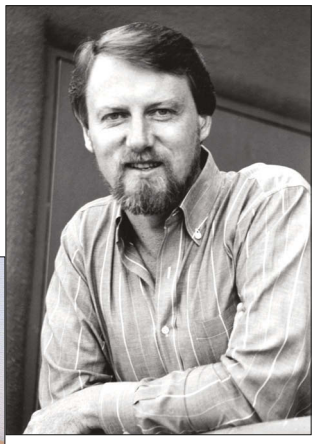
Les sites « one-page » ont un intérêt certain d'un point de vue graphique, mais aussi commercial. Ils présentent cependant certaines contraintes qu'il faut bien mesurer. La compatibilité sur les appareils mobiles reste limitée et l'impact sur le référencement est important. Il faut donc savoir utiliser cette tendance à bon escient, s'en inspirer et la détourner. Il est probable que l'on continue à voir sortir de nombreux sites monopages cette année, ce principe répondant finalement aux nouvelles attentes des internautes. Le web évolue constamment, mais relativement lentement. Ses fondements restent les mêmes, viennent s'y greffer de nouvelles technologies ou méthodes qui permettent des environnements plus dynamiques et interactifs. Comme un phénomène de mode, il faut savoir anticiper et penser aujourd'hui le web de demain.

Simon **Gombaud**
Webdesigner / Intégrateur - www.useweb.fr

Timeline : 1981

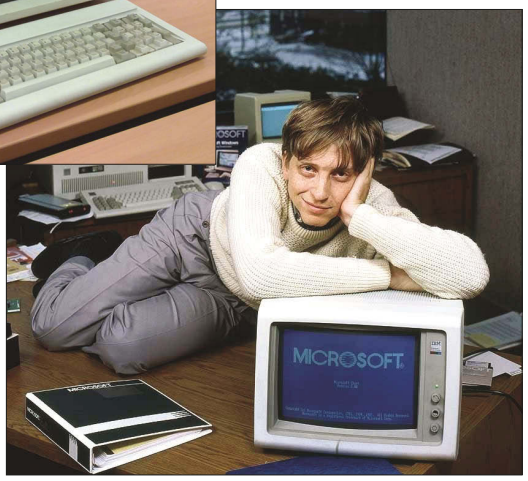
Objet : La guerre des DOS est déclarée

Durant 10 ans, Microsoft, Digital Research et IBM ont bataillé pour imposer leur propre version de DOS sur le marché du PC.



Gary Kidall, l'inventeur de CP/M.

l'IBM PC original, de son vrai nom l'IBM 5150.



Bill Gates, à l'époque où Microsoft commence à vendre son MS-DOS à des centaines de constructeurs voulant imiter IBM.

En 1980, IBM décide de lancer un petit ordinateur pour démocratiser l'informatique dans les entreprises. Pour cette machine qu'on n'appelle pas encore PC, pas question de réinventer la roue. La carte mère, issue du projet d'ordinateur Datamaster, doit fonctionner avec un processeur courant, un Intel 16 bits, et un système d'exploitation mature : CP/M. CP/M est conçu depuis 1974 par Digital Research, la start-up d'un certain Gary Kidall, pour les ordinateurs 8 bits basés sur le processeur 8080 d'Intel, ou son clone, le Z80 de Zilog. CP/M permet de taper des commandes pour exécuter des applications sur disquettes (chose rare à l'époque, les micro-ordinateurs utilisant plutôt des cassettes) et sa logithèque comprend déjà le traitement de texte WordStar (Micropro), le tableur SuperCalc (Sorcim), la base de données dBase (Ashton-Tate), ainsi que de nombreux langages de programmation (Basic, Fortran, Cobol... pour la plupart écrits par Microsoft).

CP/M détrôné par PC-DOS

Le rendez-vous avec Gary Kidall est un fiasco. Le jeune homme, 24 ans, roule des mécaniques, refuse qu'IBM distribue son système sous le nom de PC-DOS (DOS, pour Disk Operating System, est le nom du système d'exploitation des mainframes IBM depuis les années 60) et réclame de toucher des royalties sur chaque copie du système vendue, alors que le constructeur veut juste lui acheter une licence globale à 250.000 dollars. Dépités, les gens d'IBM prennent alors conseil auprès de Microsoft, l'un des plus gros acteurs de l'écosystème CP/M. Bill Gates, 24 ans aussi, saute sur l'occasion : son entreprise fournira PC-DOS ! L'associé de Bill Gates, Paul Allen, fonce alors chez Seattle Computer Products, une bande d'étudiants qui planche depuis deux ans sur un prototype d'ordinateur CP/M basé sur le nouveau processeur 16 bits d'Intel, le 8086... Celui-là même qui doit justement équiper le PC d'IBM ! Leur développeur, Tim Paterson, a décompilé le code de CP/M pour le réécrire dans une version compatible avec ce processeur, car Gary Kidall traînait des pieds pour le faire. Paul Allen signe un chèque de 75.000 dollars pour acheter les droits exclusifs de ce système d'exploitation pas encore terminé - et qui aurait dû s'appeler 86-DOS - et embauche Tim Paterson pour qu'il adapte le système à l'électronique du PC. PC-DOS 1.0 sort finalement avec l'IBM PC, en août 1981. Constatant dès les premiers jours que PC-DOS est un plagiat de son CP/M, Gary Kidall menace IBM de poursuites. Paniqué à l'idée de se faire une nouvelle fois attaquer pour avoir soi-disant volé de la propriété intellectuelle (une tradition dans son histoire), IBM prend deux décisions. La première est sans conséquence : il propose son PC, au choix, avec PC-DOS ou CP/M-86, l'adaptation de CP/M aux processeurs 16 bits d'Intel que Gary Kidall consent finalement à écrire avant fin 1981. Mais comme l'IBM PC coûte 200 dollars de plus avec CP/M-86, cette option fait un bide. Les exécutables n'étant pas compatibles entre les deux systèmes, CP/M-86 disparaît rapidement alors que PC-DOS draine tous les développements.

MS-DOS engendre les compatibles PC

L'autre décision d'IBM est plus lourde de conséquence : le constructeur laisse la propriété de PC-DOS à Microsoft, pour que ce soit ce dernier qui aille devant les tribunaux en cas de plainte sur la propriété intellectuelle. Si bien que Microsoft se retrouve avec un système qu'il a le droit de revendre à n'importe quel fabricant d'ordinateur basé sur le processeur 8086 d'Intel ou ses dérivés, les 8088 et 80186. Un peu comme Digital Research qui vend son CP/M à tous les constructeurs d'ordinateurs basés sur le 8080 ou le Z80. Or, l'intérêt de proposer un ordinateur qui exécute les mêmes applications que l'IBM PC grandit rapidement. Et c'est ainsi qu'en 1982 sortent chez Columbia Data Products, Eagle, Zenith et Compaq les premières machines « compatibles MS-DOS » (le MS signifiant Microsoft, par opposition à PC-DOS qui reste la marque du système vendu avec le PC d'IBM). Dans la foulée, le couple MS/PC-DOS passe en version 2.0, pour gérer les disques durs, et le tableur 1-2-3 de Lotus impose le PC en entreprises au détriment de l'Apple II. C'est un succès : jusqu'en 1984, Microsoft vend des licences de son MS-DOS à plus de 200 fabricants de micro-ordinateurs 8086.

Cela dit, la compatibilité n'est pas parfaite. Les applications programmées en assembleur sur IBM PC plantent ou ralentissent aléatoirement sur les compatibles, lesquels sont conçus à partir de chipsets radicalement différents de la machine originale. À cette époque, Lotus 1-2-3 et le simulateur de vol Microsoft Flight Simulator servent d'ailleurs de banc d'essai pour évaluer la compatibilité - et donc la qualité - des clones.

En juin 1984, l'éditeur Phoenix résout le problème de compatibilité en mettant au point un BIOS qui, placé dans la ROM de chaque clone, émule le matériel de l'IBM PC. Les cloneurs sont ravis, car leurs nouvelles machines n'échouent plus aux tests de qualité. Quant à Microsoft, il n'a plus besoin de développer des versions spécifiques de MS-DOS pour chaque constructeur. À ce stade, IBM fait toujours figure de locomotive et lance une nouvelle version de son PC, le PC-AT, désormais basé sur le processeur 16/32

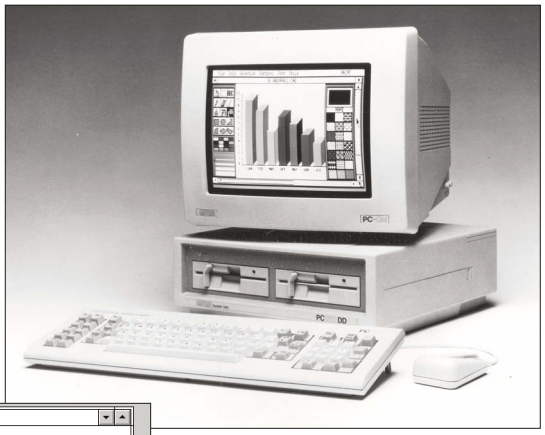
bits d'Intel, le 80286. MS/PC-DOS passe alors en version 3.0 pour supporter la nouvelle plate-forme.

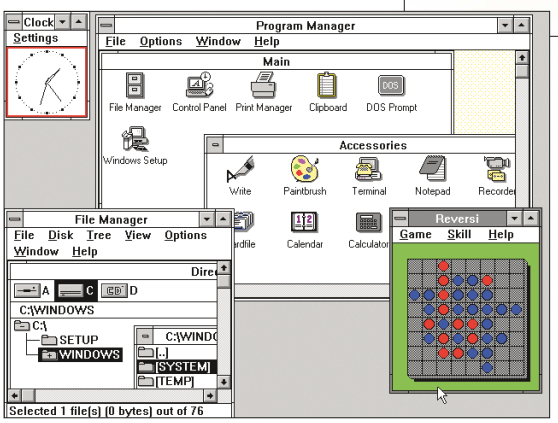
La situation échappe petit à petit à IBM

Problème, en 1986, la compatibilité PC part dans tous les sens. En haut de gamme, Compaq lance avant IBM un PC basé sur le tout dernier processeur 386 d'Intel, entièrement 32 bits, qui supporte jusqu'à 8 Mo de RAM et qui se montre deux fois plus rapide que le PC-AT sur tous les tests. Sa compatibilité avec l'existant est possible grâce à MS-DOS 3.2, lequel apporte aussi le support du réseau avec des cartes non IBM et la gestion des disquettes 3,5 pouces, plus robustes que les 5,25 pouces qu'utilise encore IBM. En entrée de gamme, Amstrad lance le premier PC avec un environnement graphique digne de celui du Mac. Et cet environnement, qui séduit aussi d'autres constructeurs, est GEM, de Digital Research, lequel revient au-devant de la scène. Pour exécuter GEM, les PC doivent être livrés avec DOS Plus, une nouvelle version de CP/M-86, devenue compatible avec les exécutables 8086 de MS-DOS et qui, en plus, apporte le multitâche. En Europe, le britannique Apricot annonce l'exclusivité d'un MS-DOS 4.0 multitâche pour ses machines.

En 1986, Compaq dépasse IBM grâce au processeur 80383 et à MS-DOS 3.2.

Toujours en 1986, Amstrad propose un PC avec le DOS de Digital Research.





Alors qu'il est censé travailler sur OS/2, Microsoft dote son MS-DOS 3.3 d'un environnement graphique, Windows 3.

Fin 1986, IBM en a marre. Ses ventes ne représentent même plus la moitié du marché du PC. Son offre est en retard avec des PC 286 sous PC-DOS 3.1, sans multitâche ni interface graphique. Le constructeur décide alors de réinventer son micro-ordinateur. Le PS/2 sera une nouvelle machine non copiable, avec un nouveau matériel et un nouveau système, OS/2, entièrement graphique et multitâche. Le lancement est prévu pour 1987. Microsoft a d'ailleurs l'ordre de se mobiliser sur l'écriture d'OS/2 et de laisser tomber tout le reste.

Les manœuvres de Microsoft pour évincer ses concurrents

Pour Bill Gates, cet OS/2 qu'il n'aura pas le droit de vendre aux autres constructeurs est une plaie. Et IBM est devenu un boulet. Alors, il va laisser le constructeur partir dans le mur. Pour commencer, les développeurs de Microsoft ne produisent rien de concret sur OS/2 entre 1987 et 1991. On explique à IBM que c'est compliqué, qu'on ne comprend pas bien ses demandes. En 1987, Compaq publie lui-même un MS-DOS 3.31 qui supporte les disques durs jusqu'à 512 Mo, alors que le MS/PC-DOS 3.3 officiel est encore limité à des partitions de 32 Mo. On se doute que le système a été écrit en douce par Microsoft. En 1988, IBM tente de reprendre la main en écrivant lui-même un PC-DOS 4.0 et en proposant ce système dans le commerce pour remplacer MS-DOS sur les compatibles. Hélas, PC-DOS 4.0 est totalement bogué. Microsoft laisse le constructeur se ridiculiser tout seul. En juillet 90, Microsoft lance Windows 3.0, le premier environnement graphique qui permet enfin à MS-DOS 3.3 d'utiliser Excel et Word à la souris aussi bien que sur Mac. Officiellement, il s'agissait d'un galop d'essai pour l'interface d'OS/2. Mais le fait que cet environnement soit disponible pour MS-DOS coupe l'herbe sous le pied du futur système. Au même moment, Digital Research lance son DR DOS 5 (5 pour succéder à la version 4 de PC-DOS), un nouveau DOS Plus avec un environnement graphique ViewMAX (nouvelle version de GEM), et le propose aussi en boutiques pour remplacer MS-DOS. Cela incite 40% des constructeurs à vendre des PC moins chers, livrés sans MS-DOS. Dès lors, Microsoft mobilise ses troupes pour, d'une part, lancer un MS-DOS 5 équivalent et, d'autre part, piéger ses logiciels : Word, Excel, Windows et toutes les applications écrites avec ses langages de programmation se mettront à fonctionner moins vite s'ils s'exécutent sur DR DOS. En juillet 91, le couple MS-DOS 5.0/Windows 3.0 est proposé à tous les constructeurs en échange de faibles royalties sur les ventes, ce qui incite tout le monde à livrer ses machines avec, tuant dès lors la carrière commerciale de PC-DOS, DR DOS et même d'OS/2 (car ce dernier a été entretemps recadré pour apporter la technologie IBM à tous les PC).

C'est ainsi que prend fin la guerre des DOS. En situation de monopole, Microsoft écrasera définitivement ses concurrents en remplaçant MS-DOS par un Windows autonome dès 1995.

Yann **Serra**

Abonnement : Programmez, 17, Route des Boulangers, 78926 Yvelines Cedex 9 - Tél. : 01 55 56 70 55 - *abonnements.programmez@groupe-gli.com* - Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30.
Tarifs abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € - CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € Canada : 68,36 € - Tom : 83,65 € - Dom : 66,82 € - Autres pays : nous consulter.
PDF : 30 € (Monde Entier) souscription exclusivement sur www.programmez.com

Une publication **Nefer-IT**
7 avenue Roger Chambonnet
91220 Brétigny sur Orge
redaction@programmez.com
Tél. : 01 60 85 39 96

Directeur de la publication & rédacteur en chef : François Tonic
Ont collaboré à ce numéro : S. Saurel, Y. Serra
Secrétaire de rédaction : Olivier Pavie
Experts : Gregory Renard, J. Dollon, P. Léothaud, A. Ribault, B. Legeard, A. vernotte, T. Guenoux, E. Issartial, M. Engelmann, F. Noé, J-S Perrier, U. Manceron, V. Hugues, C. Burceaux, V. Munoz, T. Joubert, F. Fadel, M. Thomas, C. Pichaud, C. Villeneuve, J. de Oliveira, F. Bellahcene, P-A Gury, T. Ouvre, C. Peugnet, Brice Cornet, M. Bosi, E. Dumortier, T. Cavin, S. Civetta, Y. Grenzinger, T. Guérin, A. Yafi, B. Bussière, S. Gombaud
Crédits couverture : © iStock - 12-19-13 - nabihariahi
Maquette : Pierre Sandré
Publicité : Régie publicitaire, K-Now sarl. Pour la publicité : Tél. : 01 41 77 16 03 - diff@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
Ce numéro comporte un encart jeté sur une partie du tirage.

Contacts
Rédacteur en chef : ftonic@programmez.com
Rédaction : redaction@programmez.com
Webmaster : webmaster@programmez.com
Publicité : diff@programmez.com
Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1215 K 78366 - ISSN : 1627-0908

LINUX Solutions Libres & Open Source

Le salon dédié à linux et aux logiciels libres

20&21 MAI 2014

CNIT - Paris La Défense

Toutes les solutions et nouveautés informatiques en Open Source...
Pour encore plus de libre au service de l'entreprise !

Un événement Tarsus France Groupe Média B to B

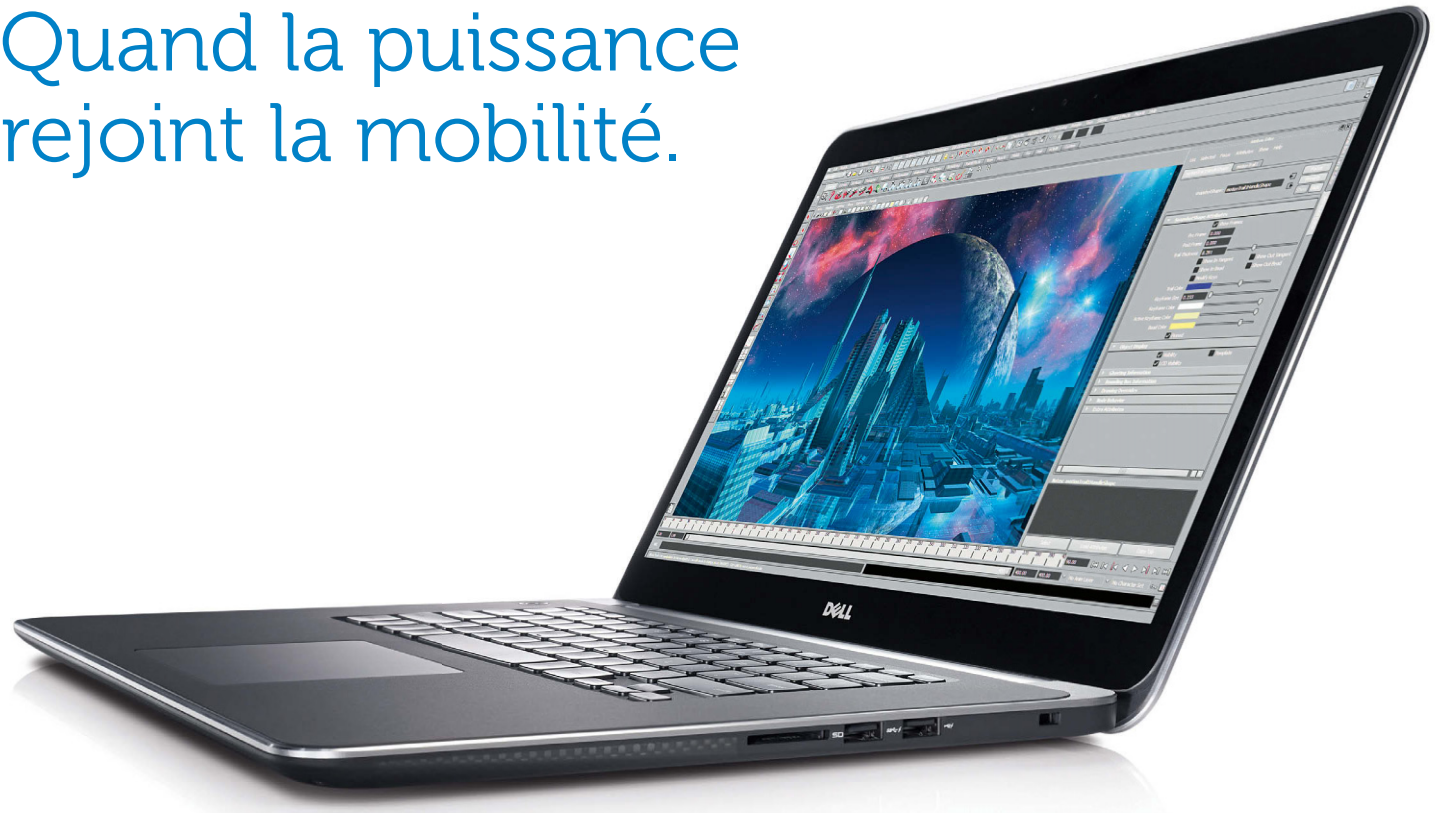
Partenaire officiel monannuaire pro.com

www.solutionslinux.fr

Dell recommande Windows.

Le pouvoir d'en faire plus

Nouvelle station de travail Dell Precision M3800. Quand la puissance rejoint la mobilité.



Certaines applications sont vendues séparément et peuvent varier d'un pays à l'autre

Découvrez la station de travail 39 cm (15,6") Dell Precision M3800.

Une conception fine (18 mm) et légère (1,88 kg)[1] qui répond à vos envies et vous offre des performances adaptées à vos besoins. Exécutez vos logiciels professionnels les plus exigeants. Profitez de la puissance des processeurs Intel® Core™ et des cartes graphiques NVIDIA® Quadro®. Repoussez les limites de votre imagination avec la station de travail Dell Precision M3800. Écran tactile QHD+ (3 200 x 1 800) disponible en option.

Rendez-vous sur **Dell.fr/pme** ou trouvez le partenaire Dell le plus proche sur **Dell.fr/findapartner**

La station de travail Dell Precision M3800 est disponible avec les processeurs Intel® Core™ i7. La station de travail Dell Precision M3800 est une marque commerciale de Dell Inc. Intel, le Logo Intel, Intel Inside, Intel Core et Core Inside sont des marques de commerce d'Intel Corporation aux États-Unis et dans d'autres pays. Dell S.A. Capital: 1 782 769 €, 1 Rond Point Benjamin Franklin - 34938 Montpellier Cedex 9 France. RCS Montpellier N° 351 528 229 - APE 4651 Z. [1] Poids de départ. Le poids varie en fonction des paramètres de configuration et de fabrication.