PROGRAMMEZ !
Le magazine du développement
PROgrammez !
www.programmez.com
mensuel n°147 Décembre 2011

Android 4.0

SDK, migration, code unique : toutes les NOUVEAUTÉS

Menaces sur FLASH ?

Cpp 11
Le nouveau C++ arrive !

EMPLOI
Recrutement : Evaluer en ligne les compétences en programmation

GPU
Initiation à la programmation GPU

WEBMASTER
Maîtriser les animations HTML5

Hardware pour les Geeks

Darknet
Le réseau TOR n'est plus anonyme

M 04319 - 147 - F: 5,95 €
Printed in France - Imprimé en France - BELGIQUE 6,45 €
SUISSE 12 FS - LUXEMBOURG 6,45 € - DOM Surf 6,90 €
Canada 8,95 $ CAN - TOM 940 XPF - MAROC 50 DH

ATELIER DE GÉNIE LOGICIEL PROFESSIONNEL

WINDEV 17

(MAINTENANT SUR IPHONE ET IPAD)

JUSQU'AU 16 DÉCEMBRE

OPÉRATION 1 PC POUR 1 EURO DE PLUS

ACHETEZ **WINDEV 17** ET RECEVEZ UN PC **DELL** OU DEUX TABLETTES **SAMSUNG** POUR 1 EURO DE PLUS

RENDEZ-VOUS SUR **www.pcsoft.fr**

Offre réservée aux entreprises, administrations, collectivités, indépendants, GIE, associations,... en France métropolitaine. Chaque élément de l'offre peut être acquis séparément. Tous les détails sont sur www.pcsoft.fr. Fin de l'opération le 16 décembre 2011

DELL

ou

Samsung GALAXY Tab 10.1

x2

ou

DELL

CALENDRIER

Montpellier	8 Nov
Toulouse	15 Nov
Bordeaux	16 Nov
Nantes	17 Nov
Paris	22 Nov
Lille	23 Nov
Bruxelles	24 Nov
Strasbourg	29 Nov
Genève	30 Nov
Lyon	1 Déc
Marseille	6 Déc

TOUR DE FRANCE

917 NOUVEAUTÉS

GRATUIT

inscrivez-vous sur ww.pcsoft.fr

Attention: 10.000 places seulement

TOUR DE FRANCE

VENEZ DÉCOUVRIR
WINDEV 17
& WEBDEV 17
PRÈS DE CHEZ VOUS

Fournisseur Officiel de la Préparation Olympique

www.pcsoft.fr

Menaces sur FLASH ?

Adobe flashe le mobile et Flex !

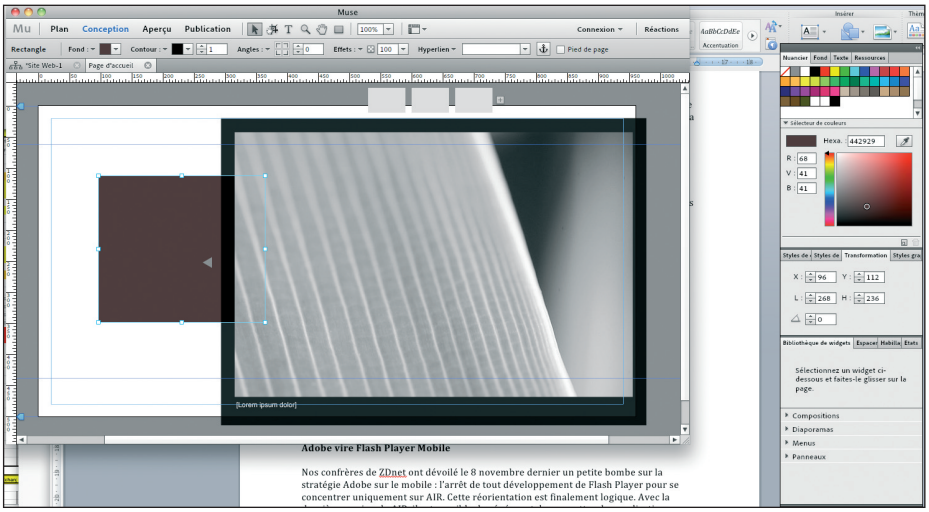
Depuis la conférence Max de début octobre, la stratégie Adobe a beaucoup évolué et les annonces se succèdent : HTML 5, cloud, applications mobiles au top, Flash Player Mobile à la casse et Flex SDK, BlazeDS, Spark, le projet Falcon, passent sous contrôle de la fondation Apache ! Flash est-il menacé à terme ? Oui dans sa forme actuelle.

1 FLASH PLAYER MOBILE : ON ARRÊTE TOUT

Le couperet est officiellement tombé le 9 novembre. Jusqu'à maintenant la présence de Flash sur les smartphones, excepté sur iOS, était un avantage pour Adobe et Android. Mais avec les évolutions des systèmes, les nouveaux terminaux et l'importance des terminaux Apple, il fallait trouver une solution plus simple. Tout d'abord, Flash Player Mobile continuera à être supporté mais aucun nouveau développement ne sera fait. La version 11.1 sera donc la dernière. Que mettre à la place ? Adobe mise sur une double approche : HTML5 et AIR. La plateforme AIR a l'avantage d'être présente un peu partout et les extensions natives permettent de générer des applications natives, incluant iOS. Les prochaines évolutions de AIR renforceront sans aucun doute cette approche. HTML 5 est le nouveau credo d'Adobe. HTML 5 et CSS 3 passent par l'outil Edge. Cependant, une partie de la communauté ne réagit pas bien à ces annonces comme nous l'avons constaté lors de la conférence "Retour sur Max" à Paris fin octobre dernier. Même si HTML 5 / CSS 3 constitue un avenir important pour Adobe, pas question de laisser tomber Flash. Nous verrons si la communauté a été rassurée ou non. Sur le support de WebGL, rien n'a été avancé, mais des annonces pourraient être faites dans quelques mois ! Sur Edge, le développement débute seulement mais devrait évoluer rapidement dans les prochains mois. Pour le moment, le niveau fonctionnel ne rivalise pas avec Flash. Cet arrêt concerne aussi Flash sur les téléviseurs.

2 BONJOUR FLASH 12 SUR PC

Pour l'éditeur, Flash sur les ordinateurs doit offrir des performances, des fonctions nouvelles à l'instar de la v11.x. L'éditeur veut être un conteneur pour le contenu riche, les jeux, la vidéo, la 3D. Les équipes travaillent déjà à la prochaine grande étape : la version 12 qui pourrait offrir la haute définition. L'éditeur travaille aussi autour de WebKit et du W3C pour l'expérience web et comment



mieux interagir, afficher HTML 5.

En attendant, une pré-version de la v11.2 est disponible. Au menu : décodage vidéo multithreadé pour améliorer les performances (tout système), mise à jour arrière-plan sur Windows. Notons aussi que IE de Windows 8 / Metro n'acceptera pas les plug-ins, donc pas de Flash Player, en revanche Air pourra être installé…

3 LA PLATEFORME 4.6 DÉJÀ LÀ

Côté plate-forme, à peine la v4.5 disponible depuis cet été que la 4.6 est annoncée pour fin novembre. Il s'agira d'une mise à jour gratuite de Flash Builder et de Flex. L'un des focus concerne les performances sur mobiles, de nouveaux composants Spark adaptés à la mobilité (splitviewnavigator, dataspinner, champs textes…). On pourra ainsi avoir des interfaces natives à chaque système. Flash Builder améliorera aussi les performances, la prise en charge des extensions natives de AIR, la gestion du captrive runtime, fonction de monitoring du trafic réseau, tests unitaires pour mobiles. Le développeur disposera du framework Starling (pour la 2D / Stage 3D), support ActionScript 3, disponible en open source. A noter que la boutique AIR marketplace a fermé ses portes fin août dernier. Starling est un framework permettant d'utiliser la GPU pour la 2D Flash. Il s'agit d'un portage du framework Sparrow déjà présent sur la plateforme iOS.

4 ADOBE LÂCHE FLEX SDK !

Le 15 novembre dernier, Adobe a annoncé une bombe à la communauté : Flex SDK, BlazeDS (serveur Java), les composants Spark, le projet Falcon (nouvelle version du compilateur ActionScript et de MXML seront gérés par la fondation Apache qui continue à développer la plateforme… Adobe a tout de suite dit qu'une équipe de développeurs contribuera aux nouveaux projets Apache. Il est clair aussi que l'éditeur n'est désormais plus maître de l'avenir de Flex SDK sur lequel repose Flex. C'est une revirement stratégique important qui agitera la communauté sur l'avenir réel de Flex, voire de Flash à terme et de l'opportunité de rester sur cette plateforme.

Dans le même temps, Adobe dit que HTML est l'avenir à long terme et que Flex est actuellement meilleur pour l'entreprise. Cela ressemble à une transition « douce » entre les deux technologies. HTML 5 est clairement l'avenir pour l'éditeur et prépare activement la migration.

numéro 147 - décembre 2011

sommaire\

\\ buzzword

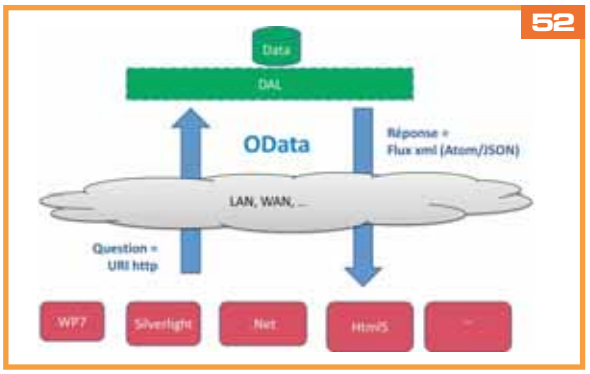
\\ actus

\\ sécurité

\\ veille techno

\\ carrière

\\ technique



\\ dossier

Android 4.0 disponible !

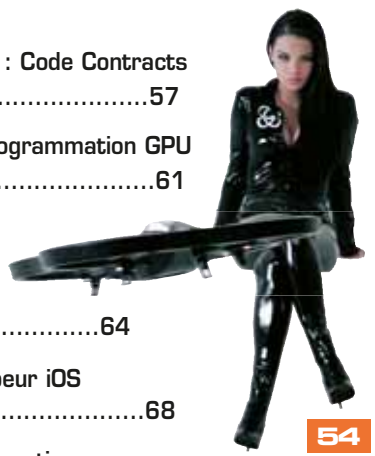
A LIRE DANS LE PROCHAIN NUMÉRO

Janvier 2012-N°148, parution le 30 décembre 2011

✔ **Gaming**
Développez pour la console Xbox Kinect

✔ **Webmaster**
Tester efficacement son site web

✔ **Veille techno**
Les 9 meilleurs projets étudiants de l'Epitech



54

61

Débrouille-toi tout seul !

Etre développeur en France n'est pas toujours de tout repos. Entre le flou des statistiques autour du taux réel de chômage des informaticiens au sens large, la difficulté d'accès aux gros contrats et la sous-traitance en cascade, pas facile de s'y retrouver. Aujourd'hui, le spécialiste informatique travaille beaucoup dans des structures de type TPE / PME qui sont souvent ignorées par les grandes entreprises, la fonction publique pour les appels d'offres. Or, c'est là que l'on trouve de nombreuses compétences que les grandes SSII n'ont pas toujours. La TPE / PME informatique servira même de variable d'ajustement pour sauver un projet ou agir en urgence. Il y a là un problème de fond qu'il faudra régler tôt ou tard. Nous avons mené l'enquête.

Autre point sensible, la réalité du chômage informatique : de 2 % à 6 % selon la source des chiffres... Finalement, la définition même de l'informaticien apparaît peu claire... Surtout que les média répètent souvent que nous manquons d'informaticiens (même si ce terme ne signifie plus grand-chose aujourd'hui). Constat étonnant que vous découvrirez dans nos pages carrières et emplois.

Plusieurs événements ces dernières semaines ont fait sursauter notre communauté. La disponibilité d'Android 4, la mort de Flash Player mobile, la nouvelle norme C++, les diverses failles de sécurité ici et là. Preuve que le développement mobile est toujours la vague déferlante. HTML 5 / CSS 3 est en passe de s'imposer comme la plateforme technique pour le contenu interactif, multimedia. Adobe l'a acté, même si l'éditeur cherche à contourner le problème avec le runtime AIR et les extensions natives. De plus en plus, l'application multi-écran, multi-terminal va s'imposer, particulièrement pour le grand public mais les applications métiers ne vont pas y échapper bien longtemps. De là à dire que l'on va devoir coder une fois, adapter l'interface pour chaque écran, est encore prématuré mais d'ici 12 – 18 mois, ce sera la réalité. C'est maintenant qu'il faut s'y préparer, comprendre les systèmes et les contraintes.

Terminons par le nouvel index des langages de TIOBE Software de novembre 2011, qui est aujourd'hui une des références sur l'utilisation et la population des langages. Une chose est certaine, même si Android domine désormais le smartphone, les terminaux iOS continuent à s'imposer et à intéresser les développeurs. Pour preuve, la nouvelle percée d'Objective-C, le langage par défaut d'Apple. Il passe de la 8e à la 6e place et pèse tout de même quasiment 6 % ! Il pourrait même bientôt dépasser PHP, d'une { !

Le trio de tête reste le même : Java, C et C++. Les deux derniers pèsent 25 %. Pour des langages que l'on donnait pour obsolètes il y a 10 ans, ils se portent bien.

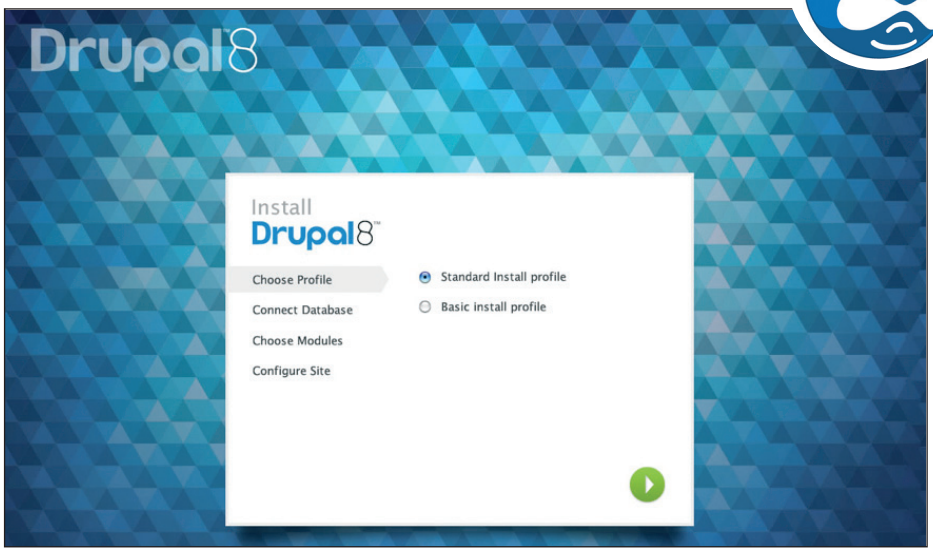
Conclusion : débrouillez-vous tout seul pour votre job, mettez-vous à Objective-C et mangez des pommes, avec ou sans petits robots verts... Et puis...oui, faites du sport.

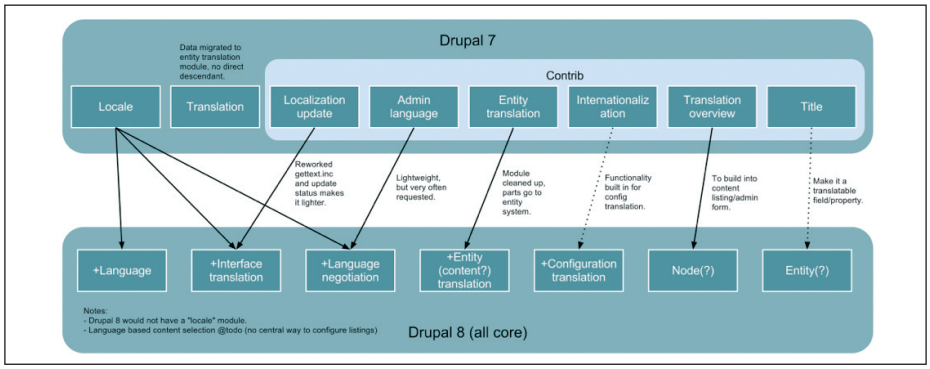
François **Tonic**
Rédacteur en chef

Editeur : Go-02 sarl, 21 rue de Fécamp 75012 Paris - diff@programmez.com.
Rédaction : redaction@programmez.com
Directeur de la Rédaction : Jean Kaminsky. **Rédacteur en Chef :** François Tonic - ftonic@programmez.com. **Ont collaboré à ce numéro :** F. Mazué, S. Saurel, Y. Grandmontagne
Experts : B. Boucard, J. de Oliveira, F. Bellahcene, N. Freier, S. Cordonnier, C. Villeneuve, L. Cacheux, D. Fages, J. Vagnet, P. Marois, PE. Pollet, C. Agoin, H. Desaunois, R. Collet, D. Flament, T. Weissbeck, L. Collet, R. Morel, C. Boucrot, S. Hertrich.
Illustrations couverture : Source Google - Nokia - I'M Watch - D.R.
Publicité : Régie publicitaire, K-Now sarl. Pour la publicité uniquement : Tél. : 01 41 77 16 03 - diff@programmez.com. Dépôt légal : à parution - Commission paritaire : 0712K78366 ISSN : 1627-0908. Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles Belgique.
Directeur de la publication : J-C Vaudecrane
Ce numéro comporte 1 encart jeté OVH

Abonnement : Programmez, 17, Chemin 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

Drupal en folie : Drupal 8, Drupagora…





Drupal 7 tire à peine le bilan de ses premiers mois d'existence que Drupal 8 pointe déjà son nez chez les développeurs. Cette version 8 sera l'occasion d'intégrer la mobilité, de disposer d'une gestion de configurations qui aujourd'hui fait défaut. Grosse nouveauté qui promet d'être le best seller : le content staging. Il s'agit de mettre en préproduction du contenu. Ce contenu est injecté sans être en production. Cela permet de tester, de peaufiner les détails avant son déploiement, un peu sur le principe du staging de Windows Azure. L'autre nouveauté dévoilée mi-novembre, la présence au cœur de Drupal 8 de composants Symfony 2. La core team du CMS décide d'en refondre l'infrastructure pour optimiser et moderniser son code. L'équipe ne voulait pas développer tout de zéro mais s'appuyer sur des projets existants. A ce jour, la communauté Drupal a choisi de baser son socle applicatif sur deux composants Symfony2 : ClassLoader, qui permet d'unifier la gestion des classes du CMS, et HTTPFoundation qui fournit une abstraction objet de la spécification HTTP. Des discussions sont également en cours pour intégrer d'autres composants.

Pour suivre Drupal 8 :

http://drupal.org/project/drupal

L'événement Drupal du 10 novembre dernier à Paris a été un superbe succès : plus de 250 personnes présentes. L'inscription payante y a grandement contribué. Et plusieurs dizaines de personnes, non inscrites, ont défilé toute la journée pour pouvoir entrer. Cette journée était avant tout destinée aux DSI, chefs de projet, et non aux développeurs purs, même si la technique n'est jamais absente de Drupal. Ce fut l'occasion d'entendre plusieurs retours utilisateurs intéressants : Mediapart, France Télévision, Radio France… Une des parties les plus passionnantes fut sans conteste celle abordant Drupal sur les multi-terminaux (mobile, tablette, borne, etc.), avec les experts de Adyax. Il n'est pas étonnant que Drupal soit aussi populaire en France même si pour trouver de vrais experts Drupal c'est devenu difficile et cher !

Les présentations sont disponibles sur : http://www.drupagora.com/programme-drupagora

Les prochaines DrupalCon : Denver (mars 2012), Munich (août 2012)

Big Data : nouveau terrain d'action des données

Depuis quelque temps, nous entendons parler de « Big Data ». Par « grosse donnée », on entend des bases de données tellement importantes que les SGBD classiques ne peuvent plus traiter d'un bloc ou que les données sont devenues tellement complexes à analyser, à déplacer que les bases « classiques » ne peuvent plus absorber efficacement les demandes. Cela concerne par exemple les données géographiques / spatiales, les données non structurées, données industrielles, photos, données RFID, réseaux divers et variés. Avec le NoSQL, le Big Data est une tendance forte de la base de données. Et tous les principaux éditeurs du secteur proposent ou proposeront des fonctions propres à gérer la grosse donnée. C'est d'autant plus une nécessité que la volumétrie de la donnée ne cesse de croître et que les SGBD ne suivent pas. La donnée devient énorme, hétérogène, non structurée. IBM propose deux axes pour répondre au Big Data : le stockage et l'analyse. Plus spécifiquement sur l'analyse, IBM utilise les outils Netezza rachetés en 2010. Ils permettent l'exécution d'analyses complexes sur de très gros volumes de données, ceci avec une rapidité 10 à 100 fois supérieure à celle des solutions concurrentes. Mais le Big Data est une notion aussi vague que large. C'est pour cela que IBM y ajoute InfoSphere BigInsight utilisant Hadoop pour traiter des données non structurées. Hadoop est un framework open source pour gérer, utiliser d'importantes volumétries de données. Il est largement utilisé par les éditeurs : Microsoft, IBM, Actuate, etc. Il est aujourd'hui incontournable pour relier des nœuds de stockages et les données grâce à son fonctionnement massivement distribué. Ce framework est soutenu et développé par la fondation Apache. SkySQL a renforcé son équipe en embauchant un expert Big Data et cloud computing, David Douglas, preuve que le SGBD open source ne veut pas se laisser distancer. On peut s'interroger sur l'avenir du SGBD tel qu'il existe actuellement. Car la donnée relationnelle n'est plus tout à fait une réalité sur Internet et le cloud, et l'explosion du volume d'information le pénalise.

NOUVELLE VERSION

Spread.NET | à partir de € 892

GrapeCity PowerTools

Ajoutez des feuilles de calcul compatibles Excel aux WinForms et apps ASP.NET.

- Accélérez le développement avec les concepteurs de feuilles de calcul, l'Assistant de prise en main et les concepteurs de graphiques
- Renseignement automatique : anticipation de la frappe dans la cellule
- Fonctionnalité de visualisation de données dont Sparklines et Camera Shapes
- Amélioration de jusqu'à 50 % des performances import/export d'Excel
- Créez des graphiques 2D et 3D complets dans vos feuilles de calcul

BEST-SELLER

Syncfusion Essential Studio Enterprise | à partir de € 1,484

Syncfusion

Interface utilisateur, rapports et composants BI dans un ensemble unique.

- Tout nouveau studio pour HTML 5, développement mobile multi-plateforme avec ASP.NET MVC
- Prise en charge des périphériques Windows Phone 7, Android, iPhone, iPad et Blackberry
- Un seul code pour plusieurs plateformes mobiles
- Diagrammes, menus, boîtes de dialogue, barres de progression, boutons, listes et commandes d'évaluation, d'onglets et de curseurs, et bien plus encore

BEST-SELLER

DXperience Enterprise | à partir de € 967

Devexpress

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

- Abonnement de 12 mois pour tous les produits et mises à jour Developer Express et accès aux versions bêta en développement actif
- Composants et outils : grilles, entrée de données, outils d'écriture de code, analyse de données, graphiques, navigation/disposition, planification, solutions reporting, bibliothèques d'impression, outils de remaniement, bibliothèques ORM

BEST-SELLER

Janus WinForms Controls Suite V4.0 | à partir de € 689

Janus systems

Ajoutez des interfaces de style Outlook à vos applications .NET.

- Vues ruban, grille, calendrier, et barres chronologique/raccourcis
- Nouveau – Style visuel Office 2010 pour tous les contrôles
- Nouveau – Support des profils client Visual Studio 2010 et .NET Framework
- Janus Ribbon ajoute Backstage Menus et la fonctionnalité onglet comme dans Office 2010
- Prend désormais en charge la sélection de cellules multiples

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

1&1 HÉBERGEMENT WEB

1&1 APPLICATIONS CLI

Chez 1&1, le succès de votre site Web est notre priorité. Avec les **applications Click & Build prêtes à l'emploi**, vous optimisez votre présence Web à votre guise. Blog, e-commerce, CMS, wiki… ce sont de nombreuses possibilités d'enrichir votre site qui s'offrent à vous ! Nos packs d'hébergement contiennent tout ce dont vous avez besoin pour assurer votre réussite en ligne :

- ✓ **65 applications Click & Build**
  Avec mises à jour automatiques : WordPress, Joomla, Drupal…
- ✓ **Logiciels de webdesign**
  Adobe® Dreamweaver® ou NetObjects Fusion® 1&1 Edition
- ✓ **Double sécurité**
  Hébergement simultané dans 2 centres de données distincts
- ✓ **Hotline non surtaxée**
  Assistance par email gratuite également incluse

MEMBRE DE united internet

*Offre 6 mois gratuits soumise à un engagement de 12 mois et à des frais de mise en service : 4,99 € HT (5,97 € TTC) pour 1&1 Dual Classique ; 9,99 € HT (11,95 € TTC) pour 1&1 e-Boutique Classique. Logiciel de webdesign offert pour toute nouvelle commande d'un pack hébergement 1&1 Dual Hosting et disponible en téléchargement à partir de l'Espace Client 1&1. 1&1 Applications Click & Build uniquement disponibles dans les packs hébergement 1&1 Dual Hosting Linux. **Offre 3 mois gratuits sur 1&1 Serveur Cloud Dynamique** soumise à un engagement de 12 mois et à des frais de mise en service : 19 € HT (22,72 € TTC).

CK & BUILD INCLUSES !

OFFRES VALABLES JUSQU'AU 31/12/11

HÉBERGEMENT

1&1 DUAL CLASSIQUE

- Dual Hosting : sécurité redoublée avec la redondance géographique
- 2 noms de domaine INCLUS
- Trafic ILLIMITÉ
- 100 Go d'espace disque
- 10 bases de données MySQL (1 Go)
- 1&1 Applications Click & Build

6 MOIS GRATUITS !

puis 4,99 € HT/mois (5,97 € TTC/mois)*

E-COMMERCE

1&1 E-BOUTIQUE CLASSIQUE

- 100 catégories, 1000 articles
- Paiement sécurisé PayPal
- Vos produits sur eBay et Kelkoo
- Design : 145 modèles personnalisables
- Trafic ILLIMITÉ

6 MOIS GRATUITS !

puis 19,99 € HT/mois (23,91 € TTC/mois)*

SERVEURS

1&1 SERVEUR CLOUD DYNAMIQUE

- Configuration ajustable
- Jusqu'à **6 cœurs**, **24 Go** de mémoire vive et **800 Go** d'espace disque
- Trafic ILLIMITÉ
- NOUVEAU : gestion de votre serveur à tout moment depuis votre mobile

3 MOIS GRATUITS !

puis à partir de 19,99 € HT/mois (23,91 € TTC/mois)*

.fr

3,99 € HT/an

(4,77 € TTC/an) la 1ère année*

.info

0,99 € HT/an

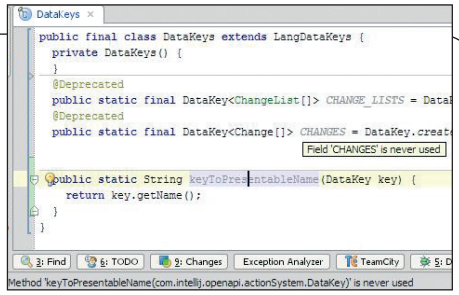
(1,18 € TTC/an) la 1ère année*

1&1

Appelez-nous au 0970 808 911 (non surtaxé)

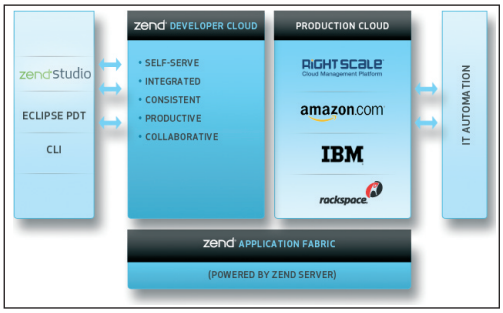
www.1and1.fr

Configuration de base gratuite pendant 3 mois. Configuration supérieure : le prix appliqué sera égal à la différence entre le prix de la configuration souhaitée et celui de la configuration de base.
Offres domaines : prix applicables la 1ère année, puis le .fr et le .info sont à 6,99 € HT/an (8,36 € TTC/an).
Offres sans durée minimum d'engagement également disponibles. Conditions détaillées sur 1and1.fr.



Jetbrains va prochainement dévoiler IntelliJ IDEA 11. Cette version inclura de nombreuses améliorations : performances, nouvel éditeur d'architecture, une interface de refactoring remodelée, support de Subversion 1.7, support étendu de Git, nouveautés natives sur OS X et Linux, apparition de Groovy 1.9, mise à jour de nombreux frameworks (grails, velocity, spring, aspectj…). La v11 sera gratuite pour les utilisateurs de la v10…

Zend dévoile phpcloud.com, qui fournit donc un environnement complet de développement et propose ensuite de déployer son application sur différents PaaS compatibles. Cette plateforme utilise les outils et frameworks Zend. Le déploiement peut se faire comme Rightscale, IBM, Rackspace, Amazon aws. En revanche, Windows Azure n'est pas supporté alors que ce PaaS propose un support PHP. Cette plateforme comprend Zend Framework et les outils de Zend Server et bénéficie de l'élasticité et de la scalabilité du Cloud. Il s'agit avant tout d'un environnement



de développement avec debug, éditeur, outils de tests, gestion du code. Le déploiement se fait via un package Application Fabric (le cloud cible doit être compatible).

JetBrains propose un éditeur de code spécialement pour Objective-C, AppCode. Idéal pour coder pour OS X, iPhone, iPad. Le développeur dispose de la complétion, d'une analyse du code à la volée, du refactoring, d'une compatibilité avec XCode, intégration avec Interface Builder. On dispose aussi d'un test unitaire, d'un émulateur iOS, d'une gestion de version et de code. Site : http://www.jetbrains.com/objc/

Microsoft dévoile les nouveaux systèmes embarqués

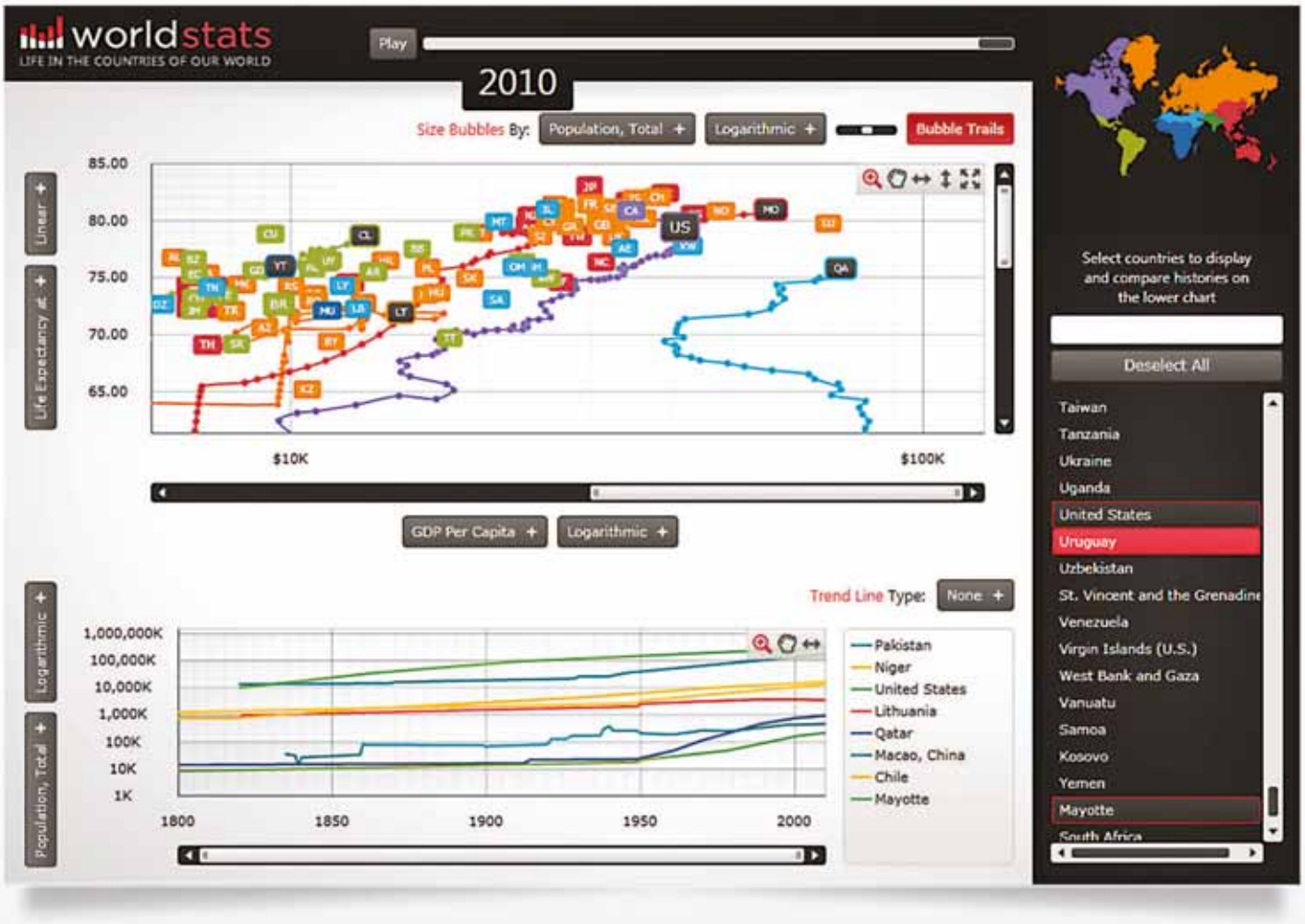
En septembre dernier, Microsoft dévoilait ses nouvelles ambitions autour des systèmes intelligents et du marché de l'embarqué en général. Cette nouvelle stratégie se met en place avec les prochains Windows Embedded : de nouveaux services, du cloud, une ouverture plus grande vers le médical, l'automobile. Le terme « système intelligent » couvre en particulier la capacité du système à rester connecté aux réseaux, à collecter et à traiter des données. Pour y parvenir, Windows Embedded se concentre sur les principaux éléments du système d'exploitation, jusqu'au noyau, et améliore le système de fichiers afin de pouvoir traiter les données générées par un ensemble de dispositifs. L'équipe travaille également en étroite collaboration avec Windows Azure pour s'assurer que les clients puissent inclure aisément le Cloud Computing dans leurs systèmes intelligents.

« Aujourd'hui, avec l'omniprésence des réseaux, l'émergence des services Cloud et la possibilité d'acquérir des processeurs très puissants à faible coût, nos clients peuvent désormais connecter leurs dispositifs embarqués classiques à des infrastructures informatiques plus importantes, ceci leur permettant d'échanger des données en temps réel avec leurs clients », explique Kevin Dallas. Selon IDC, ce « nouveau » marché représentera 800 millions de terminaux en tous genres.

Un des critères vitaux de ces systèmes est la fiabilité absolue de la plateforme. La moindre faiblesse, panne ou faille de sécurité, et c'est toute une entreprise ou un secteur d'activité qui peut être impacté. C'est pour cela que les travaux de l'informatique prouvée, du développement natif sont si importants.

Aujourd'hui, l'éditeur dévoile Windows Embedded Enterprise vNext et Standard vNext. Windows Embedded Enterprise v.Next, totalement compatible avec les applications Windows et d'une puissance équivalente à celle des systèmes d'exploitation Premium de Microsoft sur les systèmes embarqués, sera disponible pour un certain nombre de dispositifs, tels que les guichets automatiques bancaires et les kiosques, environ trois mois après la disponibilité générale de Windows 8 pour les PC. La version d'essai de Windows Embedded Standard v.Next — qui fournira la puissance, la convivialité et la fiabilité du système d'exploitation Windows sous une forme hautement personnalisable et riche en composants sera disponible pour les développeurs, via un programme preview, au cours du premier trimestre 2012. La version commercialisable de Windows Embedded Standard v.Next sera disponible trois trimestres après la disponibilité de Windows 8. Microsoft n'a pour le moment pas communiqué de dates concernant la sortie de Windows 8.

NetAdvantage®
ULTIMATE



RAPPORTS, VISUALISATION DES DONNÉES ET CONTRÔLES D'INTERFACE UTILISATEUR POUR ASP.NET, WINDOWS FORMS, JQUERY/HTML5, WPF,SILVERLIGHT ET WINDOWS PHONE 7

SCANNEZ ICI POUR DECOUVRIR ULTIMATE!
www.infragistics.com/ult

FAITES PASSER VOS APPLICATIONS AU NIVEAU SUPERIEUR
INFRAGISTICS.COM/ULTIMATE

INFRAGISTICS
DESIGN / DEVELOP / EXPERIENCE

Infragistics Ventes France N° Vert 0800 667 307 • Infragistics Ventes Europe +44 (0) 800 298 9055 • Infragistics India +91 80 4151 8042 • @infragistics

Google a dévoilé les sources d'Android 4.0. L'éditeur veut sans doute éviter les polémiques, très vives, autour de la famille 3.x. Une bonne nouvelle pour les fabricants et les développeurs qui veulent voir les entrailles. Par contre, Chromebook et ChromeOS apparaissent de plus en plus comme un échec. Acer, un des partenaires constructeurs, aurait vendu à peine 5000 machines. Samsung n'a pas révélé de chiffres. L'échec est acté et cela pose la question de la pertinence de la solution sur le marché, son utilité réelle. Pour le moment ChromeOS et ChromeBook demeurent au catalogue mais Google devra rapidement trouver une solution pour convaincre constructeurs et utilisateurs. Autre problème : la Google TV, qui elle aussi a connu depuis un an de nombreux déboires. Mais la v2 devrait améliorer les choses sur l'interface, les fonctionnalités, l'arrivée des applications d'Android Market, reste la question du contenu et de son adoption par les constructeurs…

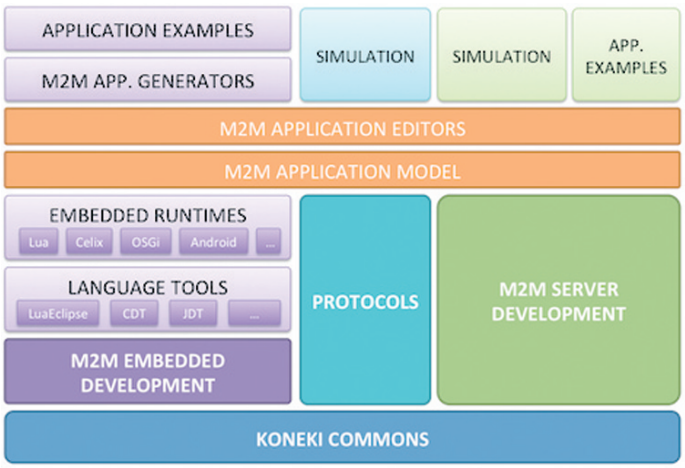
Site :
http://source.android.com/index.html

Siri, la technologie d'interface vocale d'Apple, commence à livrer ses secrets. L'équipe d'Applidium a disséqué Siri, les protocoles, son mode de fonctionnement. Les développeurs ont pu déchiffrer le protocole utilisé. Mais attention, il s'agit d'un cracking complet car pour le moment, Apple n'a pas livré les API pour utiliser la technologie. Il s'avère que Siri capture la voix, la compresse (avec Speex), introduit différentes informations (qualité, pertinence, etc.), crypte puis envoie le tout aux serveurs Apple qui se chargent d'interpréter et de construire la réponse renvoyée à l'iPhone. L'identifiant d'un 4S est obligatoire car Siri vérifie constamment cela. Applidium a écrit plusieurs outils pour exploiter la technologie. Si aujourd'hui, Siri demeure assez limité, il ne fait aucun doute qu'Apple proposera dans les prochains mois une version finale bien plus performante et espérons-le un SDK !

Voir le post détonant :
http://applidium.com/news/comment_dechiffrer_siri/

Ubuntu, la distribution Linux la plus en vogue, veut découvrir de nouveaux terminaux. Et son éditeur, Canonical, l'a clairement exposé : Ubuntu ira sur les tablettes, les smartphones, les téléviseurs, la cuisine, dans la voiture. Bref, s'attaquera aux marchés mobile et embarqué. Pour Mark Shuttleworth, la version 14.04 pourrait parfaitement fonctionner sur ces écrans mobiles et embarqués. L'interface Unity est déjà là pour pouvoir s'adapter rapidement à ces contraintes nouvelles. Entre le dire et le faire, il y a un fossé énorme que Canonical devra franchir pour disposer de versions adaptées, trouver des partenaires, convaincre utilisateurs et développeurs. Surtout, quelle stratégie d'ensemble autour d'Ubuntu ? L'éditeur veut se poser en alternative à iOS, Android, Windows Phone / Windows 8. Mais l'échec de MeeGo et le lancement discret, voire timide, de Tizen, un MeeGo nouvelle formule soutenu par Intel et Samsung, montre que ces marchés sont difficiles et qu'il faut du temps, de l'argent et une armée de développeurs spécialisés.

Eclipse a fêté son 10e anniversaire. Et la fondation a de grandes ambitions avec Eclipse 4 (projet e4), l'environnement de développement en ligne Orion, ou encore sur le marché Machine to Machine avec le projet Koneki. Ce projet a pour but de proposer un modèle, un environnement de développement pour les objets Machine 2 Machine, c'est-à-dire des objets communicants capables d'échanger, de travailler ensemble sans PC. Il faut pour cela concevoir des runtimes, des protocoles, des modèles de données et applicatifs M2M, s'intégrer aux serveurs M2M. Koneki expose des outils, des runtimes embarqués. Nous ne connaissons pas encore le planning et les axes 2012-2013. Mais les ambitions 2011 (roadmap v6) seront sans doute conservées : évolution de la plateforme, amélioration de Eclipse Marketplace, faire évoluer les projets actuels et développer les nouveaux, diversification du modèle économique de la fondation, continuer à mettre en place et à sortir Eclipse 4.x. La v4.2 est en plein développement. L'objectif est toujours de sortir une très importante évolution en juin 2012.

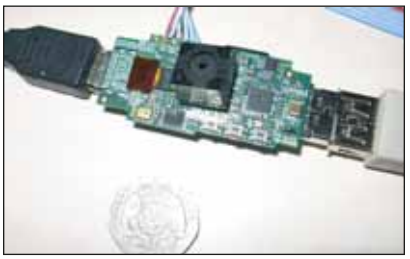


Les dernières build Eclipse :
http://download.eclipse.org/eclipse/downloads/
Roadmap v6 :
http://www.eclipse.org/org/councils/roadmap_v6_0/
Projet M2M :
http://www.eclipse.org/proposals/technology.koneki/

Nokia dévoile

enfin ses premiers terminaux mobiles utilisant Windows Phone, en version 7.5 (Mango). Il s'agit des modèles Lumia 710 et 800. Désormais, Nokia conclut la période transitoire vers Windows Phone. Ces modèles possèdent des processeurs 1,4 Ghz, un écran Amoled de bonne qualité (pour le 800) même si on n'atteint pas forcément la qualité d'un Retina. Nokia a intégré plusieurs optimisations propres au constructeur : Nokia Drive pour la navigation, Nokia Music. Pour compléter son offre, Nokia propose des oreillettes et casques audio. Le modèle 800 est à 499 €, le 710 à 329 €, ce qui est un peu cher par rapport aux offres Android et la différence avec les modèles iPhone 3GS / 4 n'est pas très significative. D'autre part, le constructeur annule la sortie du modèle 600 utilisant Symbian Belle. Mais Nokia sort plusieurs modèles de la gamme Asha, modèles d'entrée de gamme.

Un ordinateur complet à 25 $



Peut-on réellement construire et vendre un ordinateur complet (carte mère, mémoire, GPU, processeur, USB) à un tel prix ? Rhapsody Pi a relevé le défi et proposera en décembre son ordinateur à 25 $ en version A, 35 $ en version B ! La quantité est limitée à 10 000 exemplaires mais technologiquement, le défi était important. Surtout, cet ordinateur tient dans la main ! Les spécifications techniques sont tout de même impressionnantes : processeur ARM, 128 Mo de mémoire, HDMI, USB, Ethernet (version B), compatibilité OpenGL, compatibilité carte SD. La disponibilité pour le « grand public » pourrait intervenir courant 2012, le problème est la production en masse et la disponibilité des composants. Ce PC fonctionne avec une distribution Linux. Site : http://www.raspberrypi.org/

Une GPU monstrueuse !

Trois ventilateurs, double processeur GPU 850 Mhz, 2 Go de mémoire, trois connecteurs DVI, connecteur mini HDMI, la GTX 560 Ti 2Win de EGVA est un monstre de puissance pour les jeux les plus exigeants comme le monstrueux Battleshield 3. Cette carte occupe deux slots. Les tests de HardwareCanucks montrent d'excellentes performances, notamment pour les joueurs les plus exigeants mais vu le prix (presque 500 €) c'est le minimum à offrir.

Une montre Android intelligente

La montre va-t-elle devenir le prochain marché mobile ? La montre intelligente n'est pas très loin. « I'M watch » réussit à construire une montre fonctionnelle et complète utilisant un écran tactile et le système Android. Le boîtier comprend une mémoire de 4 Go, une ram de 64 Mo, un processeur Freescale, un écran de 1,54', Bluetooth pour la connexion sans fil, support des formats mp3 et AAC pour l'audio. Il est doté d'un connecteur mini USB pour recharger la montre. Surtout, elle communique avec iOS, Android, RIM, Bada… Si l'objet est intéressant en soi, il souffre d'une autonomie particulièrement médiocre (24h, 48h sans bluetooth) et la taille du boîtier est un peu trop grosse. Ensuite, quelle utilité réelle pour un tel objet ? L'avenir nous le dira.

Le développeur n'a pas été oublié. Un SDK est disponible depuis le 7 novembre dernier. Il utilise l'environnement Eclipse et le SDK Android, cependant il faudra considérer les limitations de la montre (ressources, écran). L'objectif est de disposer d'applications tierces pour l'enrichir ….

Prix : à partir de 249 €. Site : http://live.imwatch.it/

Un exosquelette français

La société française Rb3d travaille sur le projet Hercule, un exosquelette qui se veut plus simple et plus polyvalent que les modèles japonais (HAL). Actuellement encore à l'état de prototype, mais partiellement fonctionnel, Hercule pourrait être commercialisé au plus tard en 2014 selon un des principaux investisseurs, la DGA. Un des arguments de Hercule est intéressant : L'objectif de ce prototype est simple : renforcer les capacités du corps humain à transporter de lourdes charges, allant de 80 à 100 kg, sans qu'il en ressente d'effort : ce ne sont plus les muscles qui portent le poids, mais la structure du robot. D'autre part, dans la présentation officielle, Hercule aura une utilisation civile et militaire : *« Les finalités de ce prototype sont le port et la manipulation de charges lourdes. Il s'adresse donc à la fois au secteur civil et militaire.* Dans le secteur civil, *on peut d'abord penser au monde hospitalier, qui a de gros besoins. Pouvoir porter sans peine les brancards, les patients… Imaginez, en cas de catastrophe naturelle, la vitesse et l'efficacité que l'on peut gagner ! Les pompiers pourraient déblayer plus rapidement, apporter le matériel de secours là où les véhicules ne peuvent pas passer…. Les armées s'intéressent évidemment de près à ces nouvelles technologies. Quelques adaptations sont nécessaires afin de pouvoir utiliser le système dans les conditions particulières imposées par ce domaine. Hercule est donc étudié pour résister à la boue, à l'eau, à la poussière ainsi qu'aux risques d'impacts.».*

D.R.

Le lapin sauvé par le robot

C'est la saga d'un des premiers objets communicants pour le grand public. Le lapin Nabaztag, renommé en Karotz, était un belle idée : offrir une autre manière d'interagir avec les objets et d'obtenir des informations. Sans doute arrivé trop tôt, le lapin intelligent a connu les pires difficultés pour rester en vie. Mindscape avait fini par racheter l'activité Violet avant que lui-même ne fût obligé de fermer. Nous avons craint pour notre lapin mais finalement, c'est Aldebaran Robotics, le créateur du robot Nao, qui a racheté l'activité. L'un des objectifs est d'échanger les fonctions et les technologies entre Nao et Karotz, ce dernier devrait donc bénéficier de nouvelles fonctions dans les prochains mois, le temps que la période de transition se termine et que tout se mette en place dans les équipes, la stratégie et la roadmap produit. Karotz devient de facto une filiale d'Aldebaran Robotics pour les produits grand public avec comme ambition d'habituer les utilisateurs, le tout public à utiliser un objet intelligent tous les jours, première étape vers le marché robotique de masse. Sur la partie développement, aucun changement visible dans les API depuis l'été dernier. Mais Aldebaran devrait dévoiler des éléments dans les prochaines semaines.

Site : http://www.karotz.com/home

Longue vie au lapin !

Une balle sous contrôle

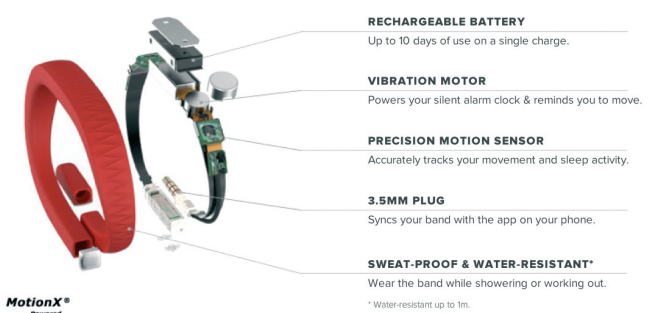
Vous rêvez de contrôler votre petite baballe préférée ? D'énerver votre chat ? Sphero a pensé à vous avec la balle Sphero. Il s'agit d'une sphère contrôlable avec un terminal iOS et Android. Pour le développeur, des API seront disponibles pour pouvoir personnaliser, intégrer les interactions Sphero dans son application. Le contrôle se fait depuis son terminal mobile (téléphone, tablette). La partie matérielle est construite par Orbotix. La sphère est faite en polycarbonate. Elle pèse 168 grammes. La connexion se fait en bluetooth. La batterie a une durée d'utilisation d'environ 60 minutes (temps de charge : 3h). Outre la direction, Sphero peut changer de couleur via un système de led. Geek à fond ! Disponibilité : 2012. Prix : environ 130-140 $.
Site : http://www.gosphero.com

Rechargez grâce à une sacoche !

Aviiq propose une station de recharge quand vous êtes en déplacement ou même au bureau. Vous pouvez y brancher jusqu'à 4 appareils en USB : smartphone, tablette. La sacoche contient une batterie 10V qui fournit l'énergie pour la recharger. Cela évite d'emporter les câbles et chargeurs que l'on oublie souvent ! Simple et plutôt élégant. Prix : 79,99 $.
Site : http://www.aviiq.com/pages/portable-charging-station

Un bracelet étonnant

Avez-vous déjà rêvé d'un bracelet vous disant qu'il est temps de faire un peu de sport, de vous réveiller après une bonne nuit de sommeil, de mieux manger ? Jawbone l'a fait ! Le bracelet UP rassemble toutes ces fonctions grâce à ses capteurs. Ce bracelet comprend une électronique complexe, une prise jack pour le connecter à votre mobile (iOS) pour récupérer / synchroniser les données. Il est résistant à l'eau. Vous pouvez suivre, via l'application, votre activité, les progrès réalisés. Prix annoncé : 99 $ (pas encore disponible en Europe). Site : http://www.jawbone.com/up



Les Rapports DevExpress

PRESENTATION CONTROLS | REPORTING CONTROLS
BUSINESS APP FRAMEWORKS | IDE PRODUCTIVITY TOOLS

XtraReports Suite

Cross-Platform .NET Support

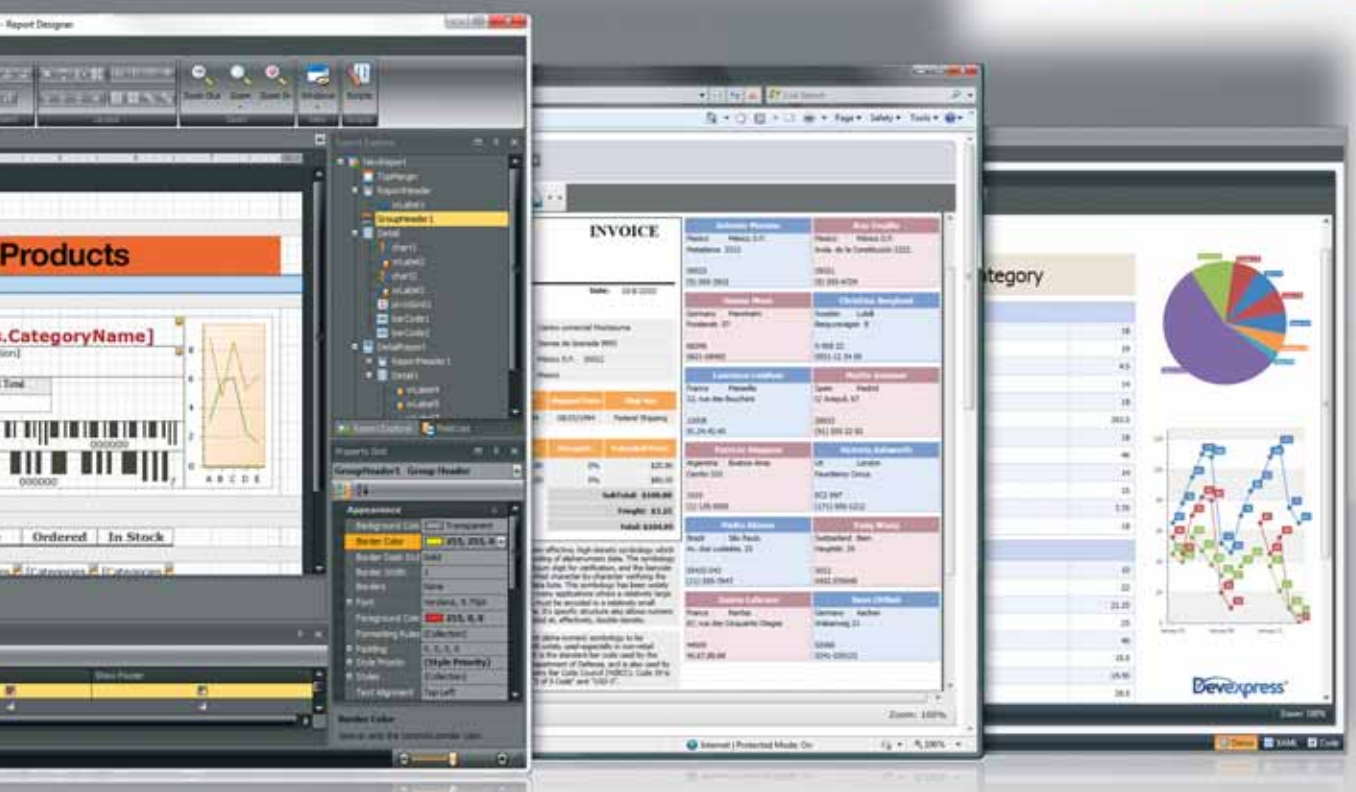
DevExpress crée des contrôles de présentation aux fonctionnalités complètes, des systèmes de rapports, des outils de productivité pour IDE et des frameworks d'applications Business pour Visual Studio .Net. Nos technologies vous aident à construire ce qu'il ya de meilleur, à avoir une vision plus claire des logiciels complexes, à améliorer votre productivité et à créer, dans le temps le plus court, des applications étonnantes pour Windows et pour le Web.

XtraReports Suite, de DevExpress, est une plateforme de Reporting de nouvelle génération pour Visual Studio. Elle vous permet de créer très rapidement des rapports professionnels, au pixel près, ciblant toutes les plateformes majeures Windows, incluant **WinForms**, **ASP.Net**, **Silverlight**, **WPF**. Au-delà de son ensemble de fonctionnalités de niveau professionnel, XtraReports Suite est livré avec un Designer de Rapports ergonomique pour l'utilisateur final, vous permettant de répondre avec la plus grande flexibilité aux exigences de Reporting.

Téléchargez aujourd'hui votre version gratuite d'évaluation et faites l'expérience de la Différence DevExpress.

www.DevExpress.com/Reporting

WIN ASP SL WPF



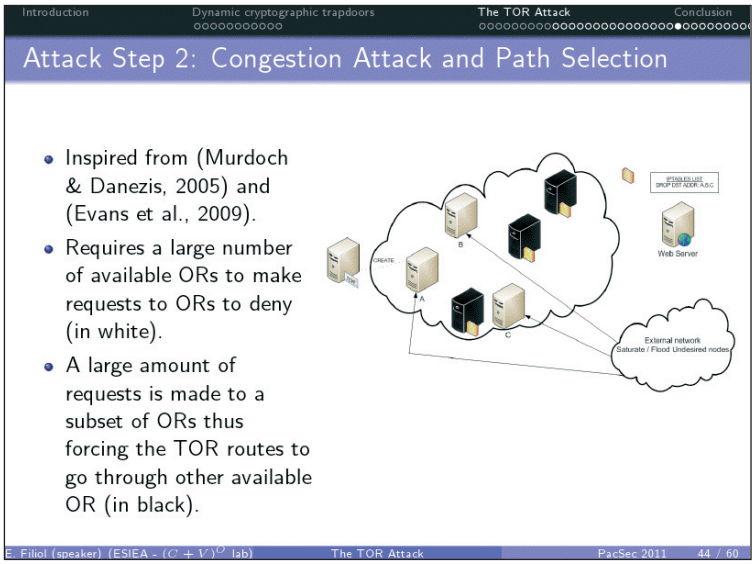
DevExpress™

WWW.DEVEXPRESS.COM

TOR perd son anonymat, le DARKNET tangue

Lors de la conférence H2HC 2011 (Hack to Hack Conference) à Sao Paulo, Brésil, Eric Filiol, directeur du Laboratoire de virologie et cryptologie de l'ESIEA (École Supérieure d'Informatique et d'électronique, Automatisme) à Laval, a décidé de révéler comment on pouvait pirater le réseau TOR réputé jusqu'à présent inviolable.

Une publication importante car TOR est utilisé principalement pour circuler sur Internet de manière anonyme et sécurisée. Un réseau d'anonymisation donc qui sert principalement à des populations à risques, quelle que soit leur position géographique, et qui ont un réel besoin de protection des communications : militaires, journalistes, hacktivistes, ou opposants politiques comme dissidents. Les ONG préconisent même l'utilisation d'un tel support et par la force des choses, TOR est devenu une véritable norme de fait, un réseau de confiance.



Pour **Eric Filiol**, *« Rappelons qu'à l'origine de TOR se trouve la marine américaine. Puis c'est devenu dans le temps une fondation. Fondation qui semble cependant restée sous contrôle étatique si l'on considère son actuel président (un ex de la NSA). Je les ai prévenus de la faiblesse sécuritaire qui permet le piratage des échanges mais ils n'ont pas bougé ce qui me pousse, à publier sur les dangers de hacking de TOR. Cette publication qui repose le cas du full disclosure ou non, est importante car elle sensibilise les utilisateurs potentiels qui pensent être protégés, à tort, sur ce réseau. Et je pars également du principe qui si moi, je l'ai trouvée, d'autres peuvent le faire et des personnages pas forcément du bon côté … Et si certains pensent que le réseau n'a pas encore été piraté parce qu'il n'y a eu aucune information sur une attaque potentielle, je pense que c'est une situation utopique comme dramatique car si un gouvernement totalitaire décide d'espionner, il pourra le faire en toute impunité. »*

Un problème d'implémentation

Après observation de TOR et notamment de sa partie cryptographique, Eric Filiol se rend vite compte que l'implémentation qui en est faite est mal réalisée et qu'il est facile de la détourner. Un virus peut facilement infecter un nœud du réseau, changer un octet en mémoire et faire pas mal de choses mais un seul virus ne peut infecter l'ensemble des nœuds qui composent le réseau TOR… Un ensemble de nœuds constitué par des volontaires géographiquement distribués sur toute la planète et qui mettent leurs machines à contribution sachant que chacun configure sa propre machine, sans contrôle dessus, seulement des préconisations. Ces machines forment un lieu de passage et quand des communications passent par chacune d'elle, elles sont chiffrées. Quand quelqu'un veut devenir un contributeur à TOR, il télécharge le code source qui a été, bien entendu, analysé par l'équipe de Filiol. Un code source qui laisse apparaître, outre des nœuds, neuf serveurs répartis dans le monde. Un code source divisé en deux parties, une publique et une cachée contenant les nœuds TOR volontairement cachés. *« La façon dont tout cela marche ? Quand Alice désire communiquer avec Bob, ils mettent tous les deux leurs machines à disposition. Elle peut chater ou envoyer des messages sécurisés, chaque intervenant ayant des communications lentes car il existe différentes couches de chiffrement sur le réseau TOR qui ralentissent d'autant le trafic. Suite à cette requête de communication, l'application TOR va, quant à elle, choisir aléatoirement trois nœuds sur une liste de nœuds par lesquels passeront obligatoirement les paquets échangés entre Alice et Bob. Une communication donc chiffrée mais également anonymisée car il est impossible de voir l'adresse IP ni l'origine de la machine, les traces sont couvertes …sauf celles sur la machine sur laquelle le destinataire est directement relié naturellement. »*

Lors de cette expérience, Eric Filiol a pris en charge la partie virus et contournement de la cryptographie. Il s'est cependant lancé dans l'aventure en essayant d'avoir une vision de l'ensemble la plus globale possible, vision purement tactique qui révèle ses origines militaires et son passé dans les scénarios de cyberguerre. Dans ce cadre, la phase de renseignements est essentielle et il faut savoir combiner les opérations tactiques et techniques. Ainsi tous les nœuds ont été repérés, soit 9000 adresses IP dans le monde. La répartition géographique en a même été extraite. Reste que réellement, il ne doit y avoir pas plus de 5000 nœuds car le mode d'adressage utilisé par TOR est dynamique ce qui signifie qu'un même nœud peut cumuler plusieurs adresses IP. Sur chaque nœud, le chiffrement AES est utilisé, un bon chiffrement en soi mais ici pas proposé dans le meilleur des modes … Après la découverte de tous

VOTRE CODE EST MULTI-PLATEFORMES
Windows, Linux, .Net, Java, PHP, Mac, J2EE, XML, Internet, SaaS, Cloud, Windows Phone, CE, Mobile, Android, iOS, ...

ATELIER DE GÉNIE LOGICIEL PROFESSIONNEL

WINDEV® MOBILE 17

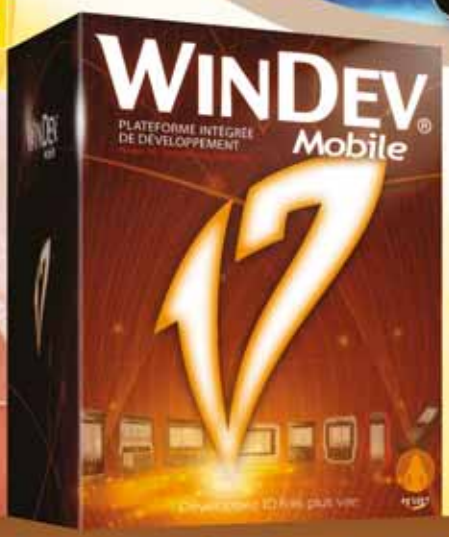
NOUVELLE VERSION

DÉVELOPPEZ 10 FOIS PLUS VITE

917 NOUVEAUTÉS

ET MAINTENANT SUR iPHONE & iPAD

VERSION EXPRESS **GRATUITE**
Téléchargez-la !



Opérations en cours :
- 1 PC pour 1 Euro de plus
- Tour de France de la version 17 (11 villes).

Tous les détails sur pcsoft.fr

iOS (iPHONE, iPad)
ANDROID
WINDOWS PHONE
WINDOWS MOBILE

Et bien entendu votre code est compatible entre les matériels, mais également avec Windows, Linux, Mac, Internet : c'est ça la magie WINDEV !

▶ DEMANDEZ LE DOSSIER GRATUIT
Dossier gratuit 260 pages sur simple demande. Tél: **04.67.032.032** info@pcsoft.fr

Fournisseur Officiel de la Préparation Olympique
www.pcsoft.fr

les nœuds et une observation d'ensemble, il ressort rapidement que si Alice veut de nouveau communiquer sur le réseau, le chemin utilisé sera obligatoirement différent et c'est là, la faiblesse de TOR qu'Eric Filiol a repéré : le changement permanent de chemins par le choix de trois nœuds différents à chaque communication. Une fois cette vision globale obtenue, le plan de bataille est sitôt mis en place : il sera mis en œuvre une attaque qui tire parti de ces changements permanents.
La première étape est la mise au point d'un virus qui ne vise que certains nœuds. Virus qui ne sera activé et désactivé que quand il sera nécessaire et au moment voulu qui peut être différent pour chaque nœud. La mise au point du virus a été faite après analyse du code source … L'idée est de non pas changer le mode de chiffrement mais de le contourner. Comment ? Le virus attaque en mémoire sur le nœud cible choisi et change l'algorithme seulement dans la mémoire comme cela, tout se passe à cet endroit et le réseau TOR physique n'est aucunement modifié : pas de modification de crypto ici mais une manipulation de celle-ci… Il sera assez difficile de repérer ce type d'attaque car il peut être déclenché sur des plages horaires non connues, restreintes et qui peuvent être différentes pour chaque assaut. Un superviseur du réseau TOR ne peut rien voir et ce, même s'il est attentif car s'il tombe par hasard sur la plage horaire choisie, elle sera modifiée le coup d'après et donc quelle que soit sa vigilance, cela rendra impossible tout traçage de l'attaque globale en soi. Ce type d'attaque porte un nom, c'est la notion de trappe cryptographique dynamique qui correspond à affaiblir à un moment choisi.

Un crack étape par étape

Parallèlement, les 9000 (5000 réels) adresses IP ont été répertoriées dans une carte Google Earth. Le mode de repérage est classique : les nœuds publics sont tous repérés dans le code source. Il a été réalisé un fingerprinting sur chaque adresse pour récupérer les données nécessaires : système d'exploitation, version dudit système, scan de vulnérabilités pour connaître les faiblesses de chaque machine et répertorier celles qui seront facilement infectables. En ce qui concerne les fameux nœuds cachés, connus seuls de la fondation TOR et qui permettent de compliquer la tâche du fournisseur d'accès qui voudrait reconstituer une portion du réseau TOR, il a été écrit une librairie spécifique de découverte automatique … des nœuds cachés. *« On a un algorithme assez compliqué qui réalise des requêtes particulières car un dissident peut, par exemple, demander un nœud avec une sécurité augmentée à la fondation TOR qui lui octroie. L'algorithme scripte ce genre de demande de façon automatique et les effectue, jusqu'à repérer le dernier des nœuds non publics. Au bout du compte, on obtient une vision complète de la topologie de TOR »*, explique Eric Filiol.
C'est le moment du démarrage de la seconde étape qui correspond au blocage de la partie du réseau voulue afin de forcer les chemins entre deux machines. Pour bloquer le réseau, pas d'attaque de type Ddos qui bloquerait le réseau efficacement mais mettrait au courant la planète entière d'une telle attaque car l'objectif est de prendre le contrôle du réseau TOR sans que personne ne s'en rende compte. Précédemment les machines ont déjà été repérées, 5000 et avec une nouvelle couche de sampling statistique et de Nessus appliqués au réseau, il ressort de cet audit au moins 30% de nœuds vulnérables sur tout le réseau TOR qui permettront de réaliser des augmentations de privilèges et de gagner des privilèges système facilement. Car là est une autre grande faiblesse de TOR : la configuration des machines qui le composent. Beaucoup de gens contribuent mais de ce fait, tout le monde est tributaire du niveau de sécurité de chaque machine et outre des préconisations au participant du réseau pour configurer sa machine de façon sécurisée, il n'est pas fait de contrôle dessus. Résultat, près d'une machine sur deux est potentiellement infectable. *« Après avoir réalisé du fingerprinting des machines françaises, on a reconstitué des clones de ces machines en interne pour réaliser la suite de notre expérience. »*
Les trois machines sélectionnées pour composer le chemin, sont chiffrées. Chacune génère sa propre clé et c'est ce qui provoque le sur-chiffrement du message qui transite, d'où la notion de pelure d'oignon pour décrire un nœud quand on considère ces différentes couches de cryptage, des nœuds appelés en anglais onions routers. Il faut donc enlever les pelures pour obtenir le message en clair, ce qui est le but de cette expérience. Jusqu'à présent, il aura été repéré toutes les vulnérabilités des machines du réseau TOR ainsi que la réalisation d'une carte repérant géographiquement les nœuds. Reste maintenant à forcer le passage du message sur un chemin donné puis d'effectuer le décryptage de ce dernier. En ce qui concerne le décryptage, un virus est injecté sur les machines qui sont donc sous contrôle. Il permet de fixer les clés et les algorithmes d'initialisation de la crypto. C'est grâce à cela, qu'il est possible d'obliger, sur des plages horaires prédéterminées, à ce que ces machines utilisent les mêmes clés et le même vecteur d'initialisation. A noter que si l'implémentation de l'algorithme de cryptage AES sur le réseau TOR avait été réalisée dans les règles de l'art, ce hacking aurait été impossible … Par conséquent deux couches de chiffrement ont volé en éclat, il en reste une dernière. L'an passé, le laboratoire a publié une librairie de cryptanalyse, Mediggo, qui permettait de détecter les communications chiffrées faibles et donc cassables en quelques minutes.

Deux types d'attaques sont possibles

En ce qui concerne le passage obligé sur trois nœuds infectés, sur l'expérience qui utilise 50 machines clonées de TOR, c'est un élève du cursus réseau qui s'en est chargé. Deux types d'attaques sont possibles pour le faire, ce sont des attaques par congestion localisée. Pour la première, si l'on dispose d'un botnet, il est possible de lancer des requêtes normales sur le réseau sur des adresses ciblées et de se servir du Time Out d'un réseau pour contrôler les nœuds voulus à la seconde près et les congestionner sans trop attirer l'attention. La seconde utilise la méthode du *packet spinning*, ce sont des requêtes qui bouclent en engorgeant de cette façon certains serveurs. Les nœuds font donc des cycles et tournent en rond, car c'est une attaque en loopback (certaines requêtes obligent les serveurs à se consulter entre eux). Tout est calculé au millimètre près pour que le réseau soit saturé juste ce qu'il faut.
Ainsi depuis le début, dès que la cartographie du réseau a été réalisée et que tous les nœuds faibles ont été repérés, il a été « facile » de faire de la gestion dynamique des nœuds infectés tout cela pour que le réseau paraisse normal aux yeux de tous alors qu'il était sous contrôle. C'est la même façon de procéder que le virus Conficker.

■ Solange **Belkhayat-Fuchs**

Avis d'expert

Sécurité et développement :

vers une attribution des rôles et une sensibilisation des chefs de projets, des développeurs et des architectes réseaux et sécurité

Au cours des 10 dernières années, le monde de l'informatique a été révolutionné par les machines performantes, les connexions Internet à haut débit, les services Web et plus récemment les applications sur terminaux mobiles et les services SaaS. Toutes ces évolutions, liées à de nouveaux outils, entrées dans le quotidien de nombreux utilisateurs, sont-elles déployées de manière fiable et sécurisée ?

On observe trois types de population qui agissent, chacun à leur niveau, sur les projets de déploiement applicatifs. Alors, quelles sont les responsabilités de chaque intervenant pour que l'utilisation de ces nouveaux outils et nouvelles applications soit synonyme de sécurité et de fiabilité ?

Les chefs de projets

Chargé de mener le projet et de contrôler son bon déroulement, le chef de projet informatique doit fédérer les équipes sur la réalisation d'objectifs court terme. Pour s'organiser de manière optimale et réduire le cycle de vie logiciel, il devra choisir parmi de nombreuses méthodes de développement agiles type XP, RAD, Scrum, ou encore la SDL: « Security Development Lifecycle », méthode de Microsoft introduisant des phases de design et de tests dédiés à la sécurité. Ces méthodes de développement permettront de dégager du temps à réinvestir dans la sécurité, mais partant du principe que les logiciels auront toujours des bugs et donc des failles, elles ne pourront que réduire le nombre de vulnérabilités introduites ainsi que leur gravité.

L'encadrement et la formation des équipes de développement sont très importants. Une équipe compétente et avertie sera beaucoup plus sensibilisée et réactive face aux nouvelles menaces. C'est pourquoi de nouveaux cursus de formation spécialisés, tels ceux proposés par le groupe SANS (formation Secure-Coding) et l'EC-COUNCIL (ECSP : EC-Council Secure Programmer) ont vu le jour. Dans une optique de sécurité offensive (Ethical Hacking), les participants apprennent à hacker leurs applications pour mieux les sécuriser. SQL-Injection, XSS, CSRF, gestion des sessions et cookies, tout est passé au crible.

Les développeurs

Le choix des bons outils est la base de tout projet applicatif. Ils doivent être maîtrisés et audités avant d'être déployés. Les Frameworks proposent aujourd'hui un bon nombre de fonctions liées à la sécurité : validation de données d'entrées, validation de robustesse des mots de passe, moteur de gestion de sessions. Malheureusement, leur évolution est à suivre en phase de pilotage car, comme toute portion de code, ces derniers peuvent introduire des failles. Le suivi des mailing-list de sécurité de l'éditeur et référentiels de vulnérabilités tels ceux de SecurityFocus, SANS, OWASP garantiront une réactivité face aux nouvelles failles découvertes.

Validation des données de formulaires, filtrage de caractères spéciaux, validation du code inclus (modules), stockage des données sensibles de manière cryptée ou hashée, le respect des règles de base de la sécurité est toujours un bon départ, mais désormais le pentest interne des applications devient une démarche nécessaire pour remonter les failles éventuelles. Pour cela, de nombreux outils, souvent méconnus, existent. Parmi les projets intéressants nous pouvons noter BackTrackLinux qui est une distribution Linux, véritable « couteau suisse » de l'audit de sécurité regroupant des Sniffers HTTP, WebFuzzers, et simulateurs de charge applicative visant à observer le comportement d'une application en situation anormale.

Pour s'initier aux problématiques de sécurité actuelles, quelques challenges d'auto-formation peuvent être relevés en ligne. En première ligne, le défi de Google-Gruyère accessible sur : http://google-gruyere.appspot.com permet de se placer dans la peau d'un hacker type, et de voir comment celui-ci trouve et exploite les failles pour mieux les contrer. Dans le cadre de ce défi, l'apprenti est guidé étape par étape dans l'attaque d'un site factice qui lui permettra de tirer un bon retour d'expérience dans une démarche pragmatique.

Les architectes réseaux et sécurité

Responsables de l'hébergement et de la disponibilité des applications, les architectes réseaux et sécurité sont chargés de choisir les équipements adaptés et de les configurer pour appliquer une politique de sécurité stricte.
Next-Generation firewalls, IPS, Data Loss Prevention, Web Application Firewalls, toute une panoplie d'équipements destinés à renforcer la sécurité des systèmes d'informations existe. Ces derniers, de plus en plus déployés dans les architectures sensibles, permettent de repousser les attaques connues en entrée de plateforme, mais aussi d'analyser le contenu du trafic et le comportement des utilisateurs. Ce premier rempart de sécurité permettra de garantir que l'application ne recevra que du trafic sain et qualifié de « normal », donc de protéger le réseau informatique : le vecteur de propagation.

En conclusion, un grand nombre de solutions existent pour sécuriser les outils et les applications. Aux différents intervenants de les adopter et de se sensibiliser sur le sujet de la sécurité pour appliquer une politique globale et ainsi combattre efficacement les nouvelles menaces.

■ Romain **Morel**,
Consultant Réseaux et Sécurité Nomios.

C'est l'histoire** d'une entreprise qui voulait offrir des services virtualisés à ses propres clients. Mais après avoir mis en place sa nouvelle infrastructure informatique, elle s'est retrouvée dans une vraie galère : ses serveurs manquaient de puissance, et le réseau utilisé pour son stockage n'était pas aussi fiable qu'elle l'espérait. **Alors qu'avec OVH...

- vSphere® as a service
- Des ressources livrées en 3 minutes
- Haute disponibilité
- Facturation horaire ou mensuelle
- Une infrastructure évolutive

Nouvelle certification
vmware vCLOUD POWERED

Cpp11, une évolution majeure du langage C++

Il s'est fait attendre 8 ans mais ça en valait la peine. Cpp11, le nouveau standard, définitivement ratifié par le comité de normalisation, apporte au langage C++ des améliorations et des nouveautés séduisantes. Découvrons-en quelques-unes qui décoiffent.

On n'y croyait plus ! Mais c'est enfin fait, le comité de normalisation a ratifié le nouveau standard du langage C++. Ce standard, à l'origine nommé C0x parce qu'il devait voir le jour avant 2010, s'appellera finalement C++11. Il aura donc fallu 8 ans d'intenses discussions pour que tout cela aboutisse. En effet, le dernier standard en vigueur remonte à 2003, comme son appellation C++03 l'indique. Au cours des discussions, le projet C++0x s'est souvent vu remanié. Ainsi, un article paru dans Programmez! 116 (février 2009) vous a présenté C++0x en faisant la part belle à une innovation de taille, les concepts, qui finalement s'est vue abandonnée peu après la parution dudit article, en juillet 2009. Cette notion de concepts, en examen depuis 6 ans, ayant été finalement jugée immature :-) C++11 n'en est pas moins riche en nouveautés, bien au contraire. Il n'est même pas envisageable de les passer toutes en revue. Nous allons en découvrir quelques-unes parmi celles que nous jugeons, tout à fait subjectivement, les plus intéressantes pour cet article.

L'ESPRIT, LES OUTILS, ET LA DOCUMENTATION

Le comité explique que le présent standard a été élaboré en suivant quelques lignes directrices bien définies. Notamment et tout d'abord, garder la stabilité et la compatibilité avec C++98 et même avec le langage C. L'introduction de nouvelles fonctionnalités est faite de préférence dans la bibliothèque standard plutôt que dans le cœur du langage. Sont aussi préférés les changements susceptibles de faire évoluer les techniques de programmation. Garder les capacités à travailler directement avec le matériel et améliorer les performances reste primordial. A ce titre, le zero-overhead, autrement dit *seul le nécessaire est inclus au code généré,* est la règle. Le comité a aussi eu comme préoccupation de *"rendre le C++ facile à apprendre et à enseigner, sans enlever les fonctionnalités requises par les programmeurs experts"*. C++ est-il désormais facile à apprendre ? Rien n'est moins sûr :-) Mais pour nous familiariser avec C++11, nous avons déjà le compilateur gcc, qui dans sa version 4.6, supporte un bon nombre des nouveautés. On trouvera ce compilateur dans les distributions Linux récentes, Ubuntu 11.10 par exemple. Pour compiler du code C++11, on procédera ainsi, au minimum :

```
g++ -std=c++0x -o lecode lecode.cpp
```

Le niveau de support du nouveau standard par le compilateur peut être consulté à http://gcc.gnu.org/gcc-4.6/cxx0x_status.html. Le compilateur de Visual Studio 2010 propose lui aussi une implémentation très partielle de ce nouveau standard. Nous utilisons exclusivement gcc 4.6 dans cet article. Le document utilisé pour le préparer est le "draft", c'est-à-dire le standard en cours de révision. C'était le seul document disponible à ce moment. Quand vous lirez ces lignes, le standard officiel, seul document de référence, sera paru. Il est possible de l'acquérir au format PDF pour quelques dollars à www.iso.org.

LA PROGRAMMATION CONCURRENTE

La programmation concurrente, ou multi-thread, devient une problématique constante du développement moderne, qui a besoin d'exploiter au mieux les capacités des processeurs multi-coeurs. Tous les langages proposent des solutions. Ainsi Java vient depuis Java 5 avec un jeu de classes évoluées pour la programmation concurrente, et propose depuis toujours des classes pour lancer des threads. Classes rudimentaires certes, mais qui avaient malgré tout le mérite d'exister dès le début du langage. Microsoft propose des librairies ou frameworks nombreux et puissants (Axum, CCR, etc.), pour la programmation concurrente en C#, et plus généralement sur la plateforme .NET. Jusqu'à maintenant, C++ ne proposait quant à lui absolument rien, ce qui donnait beaucoup d'arguments aux détracteurs du langage, même si des librairies tierces comme BOOST amenaient des solutions. La lacune est maintenant comblée et cette fois non seulement C++ supporte les threads, mais il le fait dans un standard, ce qui n'est pas peu de chose. Cela mérite que l'on s'y attarde. Ce que propose C++11 est très inspiré de BOOST. Voici un exemple minimaliste :

```
#include <thread>
#include <iostream>
using namespace std;

void worker() {
 while(true) {
   cout << "Programmez!"<< endl;
   sleep(1);
 }
}

int main(int argc, char const* argv[]) {
 thread t(worker);
 t.join(); // requis
 return 0;
}
```

Avant toute chose ce code doit être compilé correctement. Ainsi le seul

```
g++ -std=c++0x -o basic basic.cpp
```

va certes compiler le code, mais, nous aurons un crash à l'exécution comme le montre l'illustration [Fig.1]. Ce qui se passe est que nous n'avons pas fait d'édition de liens entre notre code et une librairie système et n'avons pas non plus demandé à g++ de générer du code multithreadé. Un message d'avertissement émis par le compilateur aurait été le bienvenu. L'incantation correcte pour compiler est celle-ci :

```
g++ -std=c++0x -lpthread -pthread -o basic basic.cpp
```

Comme on le voit, lancer un thread avec C++ est désormais tout ce qu'il y a de simple. Nul besoin d'écrire toute une classe. Une simple fonction fait l'affaire. Pour démarrer le thread, il suffit d'instancier la classe standard thread en passant à son constructeur la fonction constitutive du thread. Automatiquement le thread démarre. Notre programme tourne en boucle. Un Ctrl-C y mettra fin. Cependant, un programme mal écrit pourrait avoir une fin bien plus inattendue. Bien remarquer :

```
t.join();
```

La méthode join de la classe standard thread est une méthode de synchronisation. Dans notre exemple, nous n'avons à priori nul besoin d'invoquer cette méthode, puisque notre thread ne termine pas. Nous devons cependant l'invoquer, obligatoirement. En effet, le résultat sera ici de bloquer le thread principal du programme. Si nous omettons de le faire, le thread principal va se terminer, sans prendre soin de nettoyer ce qui doit l'être, en l'occurrence le thread secondaire, comme le feraient peut être d'autres langages. La politique de C++ est toujours minimaliste : ce qui n'est pas explicitement demandé n'est pas effectué. Le standard ne spécifie aucun comportement dans ce cas. Nous avons donc un comportement indéterminé et c'est pourquoi avec gcc, sans invoquer join(), notre programme crashe une seconde fois et ne termine pas correctement, comme le montre l'illustration. [Fig.2] Le résultat sera probablement similaire avec des implémentations autres que gcc. C'est donc au programmeur qu'il incombe de toujours s'assurer de la bonne terminaison des threads qu'il lance.

PASSER DES ARGUMENTS À UN THREAD

Il est naturel de souhaiter pouvoir passer des arguments à une fonction de thread. Le cas est prévu et cela se fait très simplement :

```
#include <thread>
#include <iostream>
#include <string>
using namespace std;

void worker(string conseil) {
 while(true) {
   cout << "Programmez!" << " " << conseil << endl;
   sleep(1);
 }
}



int main(int argc, char const* argv[]) {
 //thread t(bind(worker, "Abonnez-vous :-)"));
 thread t(worker, "Abonnez-vous :-)");
 t.join();
 return 0;
}
```

La liaison "à la BOOST" entre une fonction de thread et ses arguments est supportée, comme on le voit dans la première ligne de code (en commentaire) dans *main*. On préférera cependant la simplicité proposée par le standard, en passant directement l'argument au constructeur de la classe thread. Il est possible de passer plusieurs arguments à ce constructeur. Ceci est rendu possible par l'arrivée dans C++11 des templates variadiques, qui acceptent plusieurs arguments comme le fait la fonction variadique *printf* de C par exemple. Du côté client, c'est tout simple :

```
void worker(string s1, string s2) {
 while(true) {
   cout << s1 << " " << s2 << endl;
   sleep(1);
 }
}
thread t(worker, "Programmez!", "Abonnez-vous :-)");
```

LES VARIABLES LOCALES À UN THREAD

C++ vient avec un nouveau mot-clé (parmi d'autres) thread_local. Ce mot-clé a pour effet de faire des copies des variables déclarées dans un thread, chacune des copies étant affectée à l'espace d'exécution de chaque thread. Ainsi voici ce qu'il est possible d'écrire :

```
#include <thread>
#include <iostream>
using namespace std;

void worker(int id) {
 thread_local int compteur = 5;
 while(compteur > 0) {
   cout << "Id thread: " << id << " compteur: " << compteur << endl;
   sleep(id);
   compteur--;
 }
}

int main(int argc, char const* argv[]) {
 thread t1(worker, 1);
 thread t1(worker, 2);
 t1.join();
 t2.join();
 return 0;
}
```

Ici la fonction worker est threadée deux fois. Une variable thread_local compteur sera dupliquée au démarrage des threads pour être ensuite décrémentée par chaque thread séparément. Malheureusement, au moment de la rédaction de cet article, gcc n'était pas encore capable de compiler ce programme. Alors en attendant, il reste possible d'encapsuler notre fonction de thread dans une classe, de façon plus classique. Mais le code est beaucoup plus lourd :

```
#include <thread>
#include <iostream>
using namespace std;

class MaClasse {
 thread monthread;
 int compteur;
 int id;

public:
 MaClasse(int id) : compteur(5) {this->id = id;}

 void start() {
```

```
   monthread = thread(&MaClasse::worker, this);
 }

 void worker() {
   while(compteur) {
     cout << "Id: " << id << " Programmez!"
       << " Compteur: " << compteur << endl;
     sleep(id);
     compteur--;
   }
 }

 void join() {
   monthread.join();
 }
};


int main (int argc, char const* argv[]) {
 MaClasse mc1(1);
 MaClasse mc2(2);
 mc1.start();
 mc2.start();
 mc1.join();
 mc2.join();
 cout << "Programme terminé" << endl;
 return 0;
}
```

Le point intéressant de ce code est qu'il montre comment passer une méthode de classe au constructeur de la classe thread. Un constructeur couteau suisse à géométrie variable décidément bien sympathique.

MUTEX ET VERROU

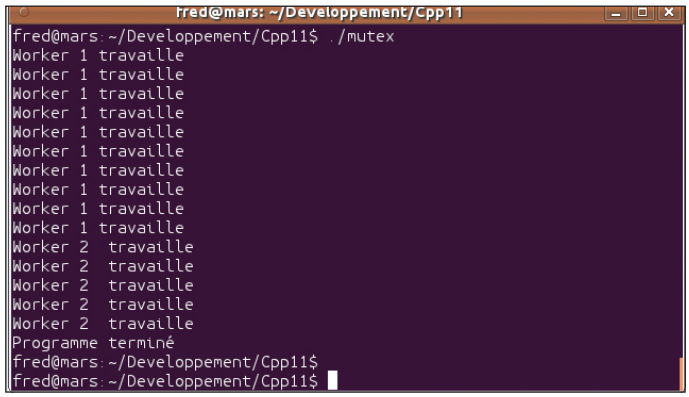
Le nouveau standard C++ fournit une quantité d'objets de synchronisation. Au centre de tout cela le mutex, objet d'exclusion mutuel, et des verrous, ou lock. Avec des idées bienvenues, comme des verrous dotés de timeout. Un tel verrou libère le mutex qu'il verrouille, quoi qu'il arrive, à l'issue du délai spécifié. Le programme ci-dessous, très basique, illustre le fait qu'un thread se place en attente quand il essaie de verrouiller un mutex déjà verrouillé par un autre thread.

```
#include <thread>
#include <iostream>
#include <mutex>
using namespace std;

mutex m;
int compteur = 10;

void worker1() {
 unique_lock<mutex> monlock(m);
 while(compteur > 0) {
   sleep(1);
   cout << "Worker 1 travaille" << endl;
   compteur--;
 }
}

void worker2() {
 sleep#
(1);
 unique_lock<mutex> monlock(m);
 while(compteur > -5)
 {
   sleep(1);
   cout << "Worker 2  travaille" << endl;
   compteur--;
 }
}

int main(int argc, char const* argv[]) {
 thread t1(worker1);
 thread t2(worker2);
 t1.join();
 t2.join();
 cout << "Programme terminé" << endl;
 return 0;
}
```

Dans ce code, on remarque, que, dans le plus pur esprit C++, il suffit d'instancier une classe de verrou pour verrouiller, ou attendre de pouvoir le faire. Lorsqu'on quitte la portée du verrou, le mutex verrouillé est libéré. Ainsi dans notre exemple, worker2 doit attendre que worker1 soit terminé pour pouvoir lui-même travailler. On remarquera encore le tout premier appel à sleep(1) dans worker2. Sa raison d'être est d'assurer que worker1 démarre bien en premier. En effet l'ordre d'instanciation des threads dans *main* n'est pas du tout une garantie pour que cela se passe ainsi, car au final c'est le système d'exploitation sous-jacent qui va effectivement faire démarrer les threads, plutôt que notre code. C++ reste un langage de bas niveau :-)

SÉMANTIQUE DE DÉPLACEMENT ET RÉFÉRENCES DE RVALUE

Nous abordons maintenant une autre nouveauté majeure de C++11 : l'arrivée de la sémantique de déplacement. Visual Studio 2010 implémente cela, et gcc aussi (pas tout à fait totalement), ainsi presque tout le monde pourra expérimenter avec cette nouveauté et l'apprécier. C++ est considéré comme le langage de la vitesse, de la performance pure. Pourtant la sémantique de déplacement et les références de Rvalue peuvent apporter jusqu'à, dit-on, 70% de gain de performances. Cette valeur de 70% est une valeur "moyenne pifomé-



Grâce à un mutex, notre second thread ne démarre qu'une fois le premier terminé.

triquement estimée" chez les experts du langage, mais elle est tout à fait impressionnante. Un des problèmes de C++, jusqu'à C++03 inclus est qu'il est basé sur la sémantique de copie exclusivement. Voyons brièvement de quoi il s'agit avec un peu de pseudo code :

```
string a,b,c ;
string conseil() {
 return "Abonnez vous :-)";
}
a = "Programmez!";
b = a;
c = conseil();
```

Dans ce code lorsque **b** est affecté par le contenu de **a**, le contenu de **a** est copié, ou, autrement dit, dupliqué. C'est la sémantique de copie. Après l'affectation, **b** et **a** contiennent la même chose, mais modifier le contenu de **b** à postériori ne modifiera pas le contenu de **a**. C'est toute la raison d'être de cette sémantique : que les Rvalue ne soient pas modifiables et soient vues comme des types const T&. (Pour faire simple on appelle une Rvalue ou Right value, une valeur à droite d'un signe égal). Dans la ligne de code c = conseil(); nous avons le même phénomène. La fonction construit une instance de string, puis cette instance est copiée pour affectation dans c. Des copies d'objets comme la classe string sont dites des copies profondes, car de telles classes détiennent des pointeurs sur des zones mémoires qu'il faut allouer/désallouer au cours des opérations et dont, bien sûr, il faut copier le contenu. Tout cela peut vite poser des problèmes quand les données ainsi copiées sont volumineuses. Il peut même arriver que cette copie soit impossible faute de ressources système. En outre, si les copies sont répétitives, les performances du code vont s'effondrer. Or, les programmeurs C++ le savent bien, les opérations (cachées) de copie faites à l'insu de leur plein gré peuvent être très nombreuses, même dans les lignes de code les plus innocentes. Pourtant, si l'on reprend c = conseil(); on se dit qu'il n'y aucune raison que la copie soit faite finalement. Qu'importe que l'on puisse ou non changer la Rvalue, puisque celle-ci sera détruite au sortir de la portée de conseil. C'est ici qu'intervient la sémantique de déplacement et la référence aux Rvalues. En C++11, notre fonction peut être écrite ainsi :

```
string&& conseil() {
 return "Abonnez vous :-)";
}
```

Le double && dit que notre fonction ne retourne plus une instance de string, mais une référence sur ce qui est de facto une Rvalue dans c = conseil(); Comment est-ce que cela fonctionne ? Très simplement. Au cours de l'affectation, ce que détient c est d'abord nettoyé. Ensuite le pointeur sur les données de la Rvalue est passé à c. Et celui de la Rvalue est mis à zéro. Au sortir de la portée, la Rvalue est détruite. Lors de cette destruction, c'est un delete 0 qui est effectué, c'est-à-dire rien. A l'issue de ce petit tour de passe-passe, c détient les données pour un coût d'exécution quasi nul. Le plus beau dans tout cela est qu'il n'y a même pas besoin, dans un cas simple comme celui de notre exemple, de modifier le code existant. Le compilateur se chargeant, par défaut, de modifier la signature de notre fonction afin qu'elle retourne une référence à une Rvalue. Des classes peuvent toutefois demander à ce que leur code soit modifié. Des méthodes devront être surchargées et il devra y avoir un constructeur de déplacement. Ainsi, par exemple, la classe standard vector de C++11 se voit enrichie (entre autres!) de :

```
vector(vector&&);
void push_back(T&& x);
```

Mais cela est finalement peu de chose en comparaison des gains obtenus.

INFÉRENCE DE TYPE

Le mot clé **auto** se voit doté d'une nouvelle sémantique. Auparavant c'était un indicateur de stockage pour indiquer qu'une variable n'était valable que dans la portée où elle était déclarée, ce qui était de toute façon le comportement par défaut. Autrement dit, ce mot-clé était parfaitement inutile :-) Désormais **auto** peut prendre la place du type dans une déclaration de variable et le type réel est déduit de la façon dont la variable est initialisée.
Par exemple :

```
auto i = 1; // i est un entier
auto f = 0.69; // f est un flottant
auto s = string("hello"); // s est une instance de string.
```

Quel intérêt demanderez-vous ? Est-ce que C++ est en train de se déguiser en Javascript ? Non, pas tout à fait. Dès que le programmeur C++ aura goûté à **auto**, il ne pourra plus s'en passer. Par exemple :

```
#include <iostream>
#include <vector>
using namespace std;

int main(int argc, char const* argv[]) {
 vector<int> v = {1, 2, 3};

 for (vector<int>::iterator p = v.begin(); p!=v.end(); ++p) {
   cout << *p << endl;
 }

 cout << "avec auto" << endl;
 for (auto p = v.begin(); p!=v.end(); ++p) {
   cout << *p << endl;
 }
}
```

Formidable ! **auto** dispense d'écrire vector<int>::iterator p et le remplace par auto p. On aura compris qu'en vertu même du fonctionnement de ce nouveau mot-clé, toute variable **auto** devrait être initialisée lors de sa déclaration. Dans le domaine des types, un autre mot-clé fait son entrée, **decltype**

```
int i;
decltype(i) j = 2;
```

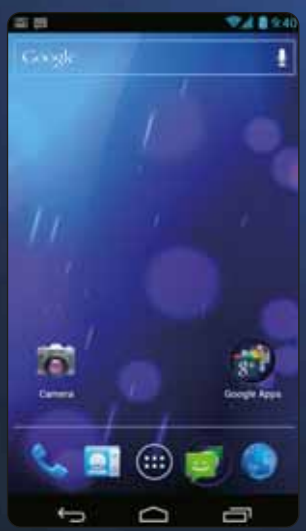
Ici **decltype** indique que **j** prend le type de i, donc int en l'occurrence. **decltype** va se révéler une aide précieuse pour les programmeurs qui écrivent des templates.

EN GUISE DE CONCLUSION

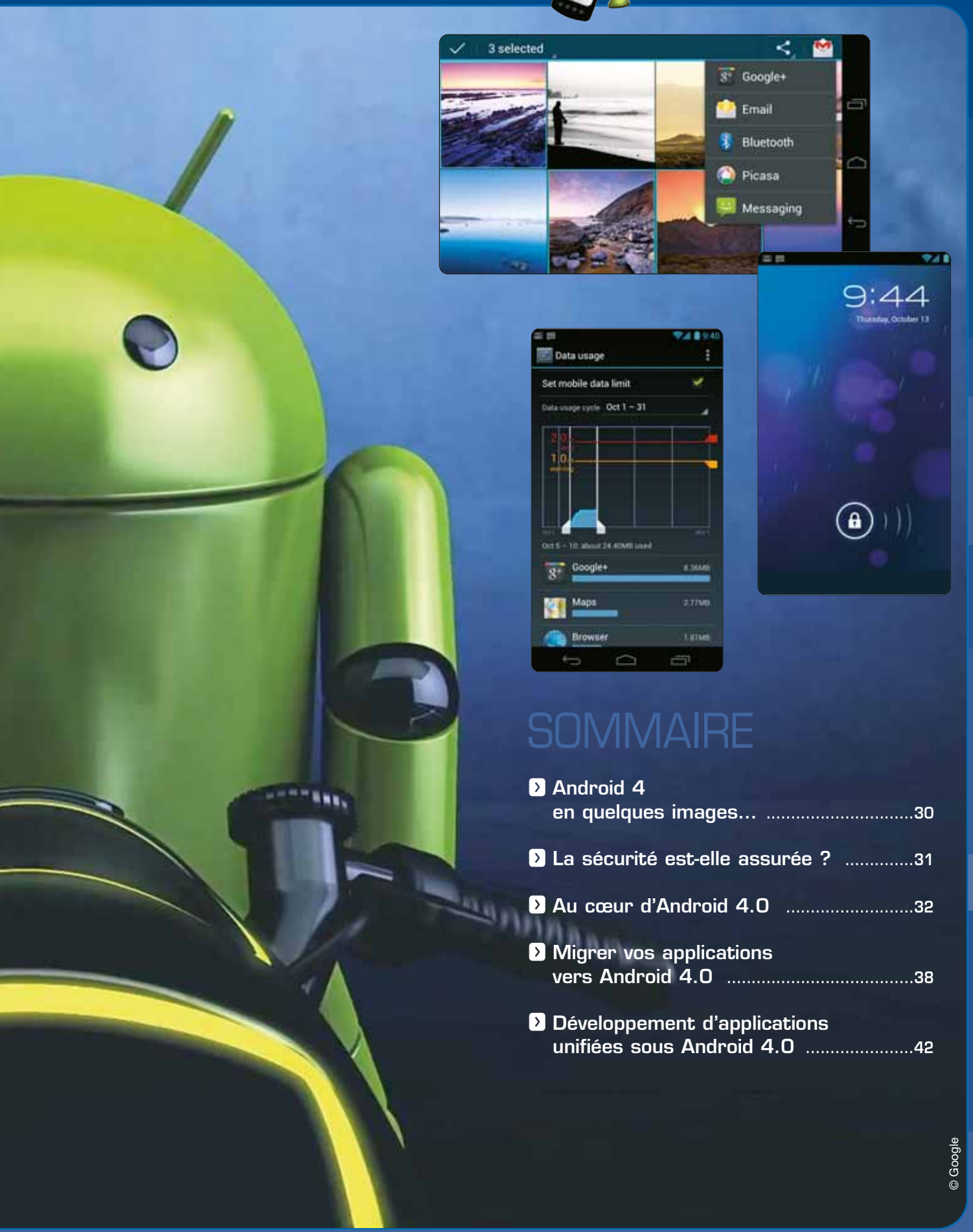
Nous sommes bien loin d'avoir fait le tour de C++11 qui est réellement une évolution majeure de ce langage plus puissant que jamais. Nous aurions pu parler des expressions lambda, des boucles foreach, des templates variadiques, des tuples, des expressions régulières, et de tant d'autres nouveautés. Pour une prochaine fois :-)

■ Frédéric **Mazué** - *fmazue@programmez.com*

Android 4.0
Enfin disponible !



Android 4.0 se veut plus beau, plus simple, plus intelligent. Cela passe par une interface redesignée, de nouveaux objets.



SOMMAIRE

Android 4 en quelques images...

Android 4.0 se veut plus beau, plus simple, plus intelligent. Cela passe par une interface redesignée, de nouveaux objets. Faisons le tour des nouveautés en images.

LE COIN DES UTILISATEURS

[Fig.1]

Le choix des applications ouvertes est plus compréhensible pour l'utilisateur, notamment sur les applications actuellement en exécution.

[Fig.2 et 3]

Le verrouillage bénéficie des nouvelles fonctions : contenu en direct provenant de widget, notification, réponse à un SMS sans déverrouillage. Possibilité de déverrouiller par reconnaissance faciale.

[Fig.4]

Android 4 introduit un nouveau système vocal pour dicter des textes. Il gère les pauses, les longues « dictées ». Cela ressemble un peu à Siri même si fonctionnellement il est plus limité.

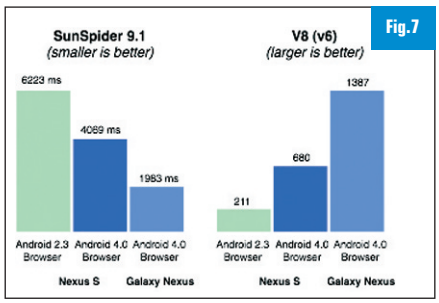
[Fig.5 et 6]

Android 4 propose de nouvelles fonctions liées à la caméra et à la manipulation des images : focus continu, gestion de l'exposition, mode panorama. La galerie photo a été revue et corrigée en incluant un éditeur photo. Sur la partie vidéo, on bénéficie de filtres de transformation temps réel.

SUPER V8

Google a beaucoup travaillé sur les performances du moteur javascript V8 et sur le moteur de rendu web, Webkit. L'éditeur a fourni des jeux de résultats par rapport à la v2.3 mais pas par rapport à la v3. Les progrès sont nets. Les utilisateurs seront ravis [Fig.7].

Fig.7



LE COIN DES DÉVELOPPEURS

La suite du dossier sera uniquement consacrée aux développeurs. Mais avant de démarrer, retenons quelques améliorations et nouveautés. Tout d'abord, Android 4 est la première version unifiant le système smartphone et tablette tactile, suivant l'exemple d'Apple avec iOS. Cela signifie que les API de Android 3.x, version tablette, sont intégrées, l'interface et les API d'interface ont été unifiées dans un unique framework.

Ainsi, les développeurs smartphone accéderont aux notifications enrichies, à la sélection multiple, accélération matérielle 2D, streaming live http, support du protocole RTP, framework pour le DRM, support des périphériques d'entrées (manette, clavier, souris...).

Le SDK r14, ou supérieur, est requis. A noter que la r15 a été déployée très rapidement après la r14. Cette version corrige de nombreux bugs

Sur les changements d'Api : http://developer.android.com/sdk/api_diff/14/changes.html

N'oubliez pas de consulter le guide développeur (prise en compte de la v4) : http://developer.android.com/guide/index.html

Fig.1

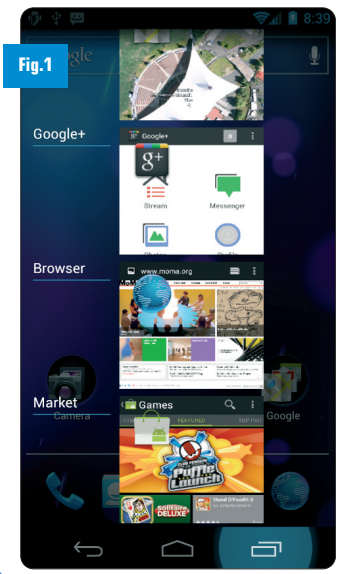


Fig.2

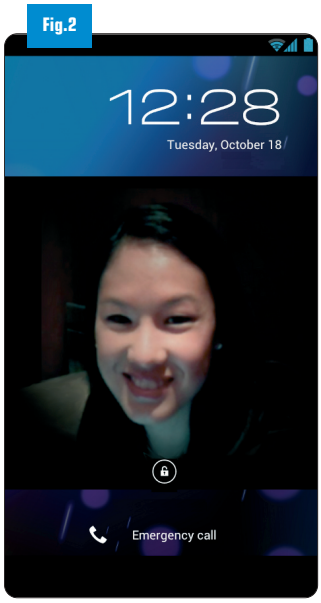


Fig.3

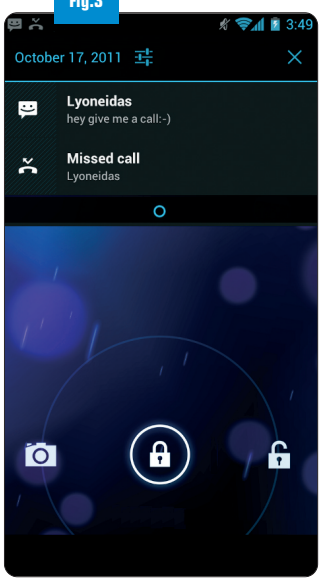


Fig.4

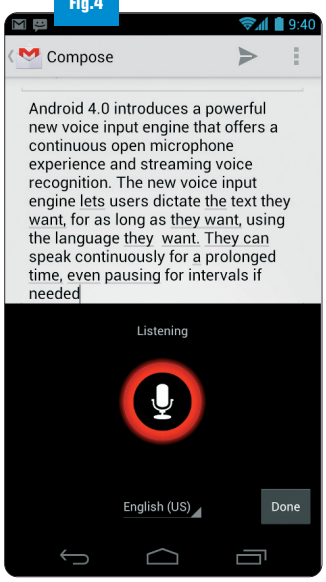


Fig.5

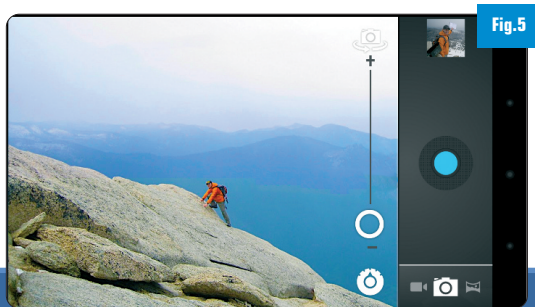
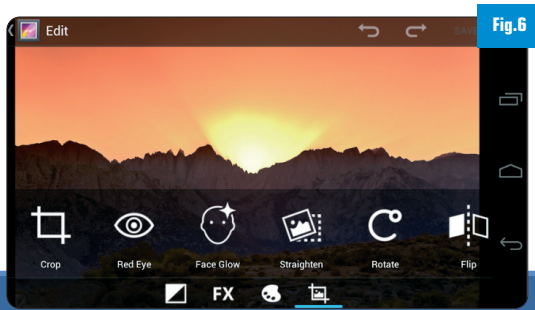


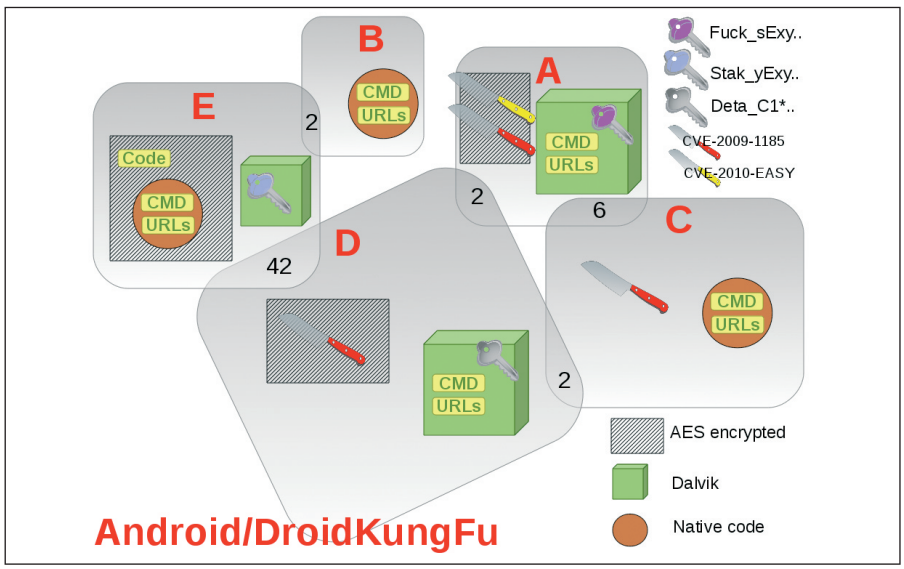
Fig.6



La sécurité est-elle assurée ?

Il n'y aura pas de jaloux. Si le PC connaît des vagues d'attaques depuis de nombreuses années, le smartphone n'allait pas être oublié bien longtemps. Depuis quelques mois, les attaques se multiplient. Comme nous l'a précisé Axelle Apvrille (Fortinet), jusqu'à présent, il s'agissait d'utiliser des failles du système, des applications et des API. Maintenant, nous passons aux exploits, ce qui est plus préoccupant. Et Android, tout comme iOS, devient une cible de choix.

Pour preuve, l'éditeur de sécurité Fortinet a investigué la sécurité sur smartphone et notamment sur Android. L'un des problèmes actuels concerne des attaques de type Botnet à l'instar de DroidKungFu, pouvant charger des logiciels malveillants, ouvrir des applications, le navigateur, supprimer des fichiers. Pour l'éditeur, ce Botnet est une évolution des menaces sur smartphone. *"Tandis que les premières tentatives de logiciels malveillants sur Android, comme Zitmo (Zeus in the Mobile), sont capables d'intercepter le type d'authentification à deux facteurs que les banques utilisent pour valider l'identité du titulaire du compte lorsqu'il se connecte, DroidKungFu fait beaucoup plus. En prenant la forme d'une application client VPN légitime, le logiciel malveillant s'implante rapidement dans les appareils en utilisant l'ingénierie sociale. Une fois exécuté, DroidKungFu télécharge d'autres logiciels malveillants, ouvre des URL dans un navigateur, lance des programmes et supprime des fichiers du système,"* indiquait récemment Derek Manky (Fortinet).



Les variantes apparues de DroidKungFu

Les URL courtes, un danger réel !

Mais le botnet n'est pas l'unique danger. Ainsi, on parle de plus en plus des raccourcis d'URL comme TinyURL. *« Cependant, l'avantage du service de raccourcis d'URL est aussi sa plus grosse faiblesse, car le service permet aux criminels de masquer des liens malveillants qui peuvent infecter le système d'un utilisateur. Historiquement, Fortinet a toujours recommandé aux utilisateurs de placer leur curseur sur l'URL douteuse avant de cliquer dessus pour voir si le lien redirige vers une page douteuse. Cette mesure de sécurité n'est pas applicable aux URL raccourcies. Il n'y a aucun moyen sûr de prévenir à l'avance un utilisateur qui clique sur une URL raccourcie s'il sera redirigé vers un site malveillant. Une façon de déterminer si une URL raccourcie pointe vers un site malveillant est de vérifier le domaine à la fin du lien. La plupart des services de raccourcis d'URL malveillants observés récemment ont utilisé le domaine .info. Une autre façon de détecter si une URL raccourcie redirige vers un site malveillant est de vérifier le lien douteux dans un outil de filtrage d'URL, comme le URL Lookup de Fortinet. Enfin, une bonne solution de filtrage Web protège contre les services de raccourcis d'URL car le domaine tout entier est encore déterminé et vérifié »*, précise l'éditeur.

L'attaque par exploit

L'exploit le plus redoutable est la prise de contrôle du terminal en mode root, permettant à l'attaquant d'utiliser le smartphone : récupérer le carnet d'adresses, effacer des données, lancer des sites, injecter du code illégal, etc. Des logiciels tels que GingerMaster exploitent ce type d'attaque, l'exemple suivant est assez éloquent : http://www.cs.ncsu.edu/faculty/jiang/GingerMaster/

L'un des objectifs des attaquants est clairement l'argent. Les attaques à l'aide d'un malware bancaire se multiplient notamment avec des logiciels de type Zeus (Zitmo pour la version mobile). Il s'agit de récupérer les informations bancaires que l'utilisateur pourrait recevoir par SMS. L'un des problèmes de conception que ces attaques montrent, concerne le niveau de privilèges / permissions des applications. Il faut être très rigoureux sur ce point et restreindre les permissions des applications au risque d'ouvrir des portes au cœur même du système. Le site CERT-LeXSI l'explique très bien ici : http://cert.lexsi.com/weblog/index.php/2011/07/12/416-zeus-in-the-mobile-android

Pour aller plus loin :
http://blog.fortinet.com/tag/android/

■ François **Tonic**

Au cœur d'Android 4.0

La nouvelle version d'Android vient de sortir, enfin ! Avec elle, promesse est faite par Google de nous donner accès aux sources des évolutions apportées depuis Android 2.3. Ces évolutions sont nombreuses dans cette version 4.0 dite Ice Cream Sandwich (ICS pour les intimes) et particulièrement au niveau de l'API. Nous allons vous présenter dans cet article les principales améliorations apportées à l'API du point de vue du développeur. La plupart des fonctionnalités traitées seront accompagnées d'exemples concrets vous facilitant la prise en main de l'API d'ICS. [Fig.1]

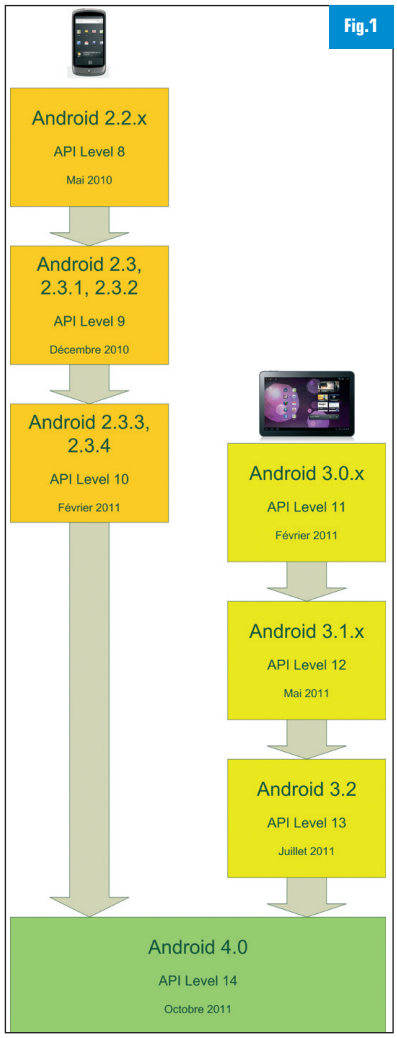


Fig.1

Nouveautés de l'API de contacts

L'API de contacts d'Android n'a pas subi de profonde refonte, contrairement au passage à la version 2.0 (Eclair). En revanche, de nombreux éléments supplémentaires font leur apparition.

Il est désormais possible d'accéder au profil de l'utilisateur du téléphone de la même façon que les autres contacts. C'est la classe ContactsContract.Profile qui se charge de fournir les informations pour la nouvelle table. De la même façon qu'un contact peut avoir plusieurs "raw contacts" représentant plusieurs comptes différents liés ensemble, le profil utilisateur peut lui aussi posséder plusieurs "raw contacts". On pourra ainsi retrouver un "raw contact" pour le compte Google, un pour le compte Facebook, un pour le compte Twitter… Deux nouvelles permissions "READ_PROFILE" et "WRITE_PROFILE" permettent d'accéder respectivement en lecture et en écriture au profil utilisateur.

L'autre gros ajout à l'API de contacts est la possibilité pour l'utilisateur d'envoyer à ses contacts des invitations pour ses différents réseaux sociaux, directement depuis la nouvelle application de gestion des contacts, People. Pour apparaître dans la liste des invitations possibles, une application doit fournir un service de synchronisation. Regardons par exemple le fichier AndroidManifest.xml du projet SampleSyncAdapter, fourni avec le SDK :

```
<service
  android:name=".syncadapter.SyncService"
  android:exported="true">
  <intent-filter>
    <action android:name="android.content.SyncAdapter" />
  </intent-filter>
  <meta-data
    android:name="android.content.SyncAdapter"
    android:resource="@xml/syncadapter" />
  <meta-data
    android:name="android.provider.CONTACTS_STRUCTURE"
    android:resource="@xml/contacts" />
</service>
```

[Fig.2]

On définit la classe SyncService qui servira de base à notre service. Celle-ci initialise une instance de SyncAdapter, où se trouvera le code nécessaire à la synchronisation. On ajoute ensuite deux

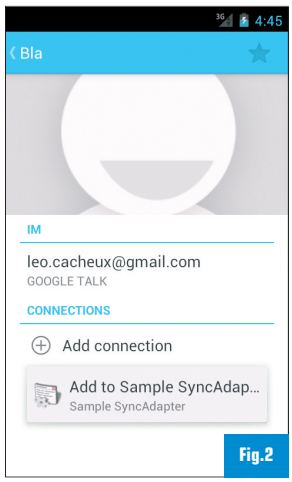


Fig.2

fichiers XML. Le premier, syncadapter.xml, définit quelques paramètres sur le type de compte utilisé et n'offre rien de nouveau sous Ice Cream Sandwich. Le second, contacts.xml, contient en revanche de nouvelles informations :

```
<ContactsAccountType
  xmlns:android="http://schemas.android.com/apk/res/android"
  inviteContactActivity="com.example.android.samplesync.
activities.InviteContactActivity"
  inviteContactActionLabel="@string/invite_action_label"
  viewContactNotifyService="com.example.android.samplesync.
notifier.NotifierService"
  viewGroupActivity="com.example.android.samplesync.activities.
ViewGroupActivity"
  viewGroupActionLabel="@string/view_group_action_label"
>

  <ContactsDataKind
    android:mimeType="vnd.android.cursor.item/vnd.samplesync
adapter.profile"
    android:icon="@drawable/icon"
    android:summaryColumn="data2"
    android:detailColumn="data3"
    android:detailSocialSummary="true" />
```

```
</ContactsAccountType>

• inviteContactActivity indique l'activité à appeler lorsque l'utilisateur demande l'ajout d'une connexion via notre application
• inviteContactActionLabel indique le texte affiché pour l'ajout d'une connexion
• viewContactNotifyService indique un service qui pourra récupérer l'évènement lorsque le contact est ouvert
• viewGroupActivity indique l'activité qui sera ouverte lorsque l'utilisateur demande les informations d'un groupe lié à notre application depuis l'application People
• viewGroupActionLabel indique le texte affiché pour ouvrir cette activité
```

Deux autres nouveautés mineures font leur apparition. La première est la gestion de photos plus grandes (256x256 au lieu de 96x96), fournie par la classe ContactsContract.DisplayPhoto. Enfin, la classe ContactsContract.DataUsageFeedback permet de récupérer des informations sur l'utilisation faite des contacts : nombre d'utilisations d'un numéro de téléphone ou d'une adresse mail, envois de SMS…

API calendrier

Afin d'unifier les applications de calendrier, Android propose désormais une nouvelle API, similaire à celle des contacts. Un provider permet aux applications dotées des permissions READ_CALENDAR et/ou WRITE_CALENDAR d'accéder à l'ensemble des données de calendrier. L'application de Google dans sa nouvelle version utilise bien évidemment cette interface. Ainsi, on peut par exemple imaginer une application de concerts alimentant cette base, tandis qu'une autre application de visualisation de planning intègrerait ces données ainsi que celles de tous les calendriers de l'utilisateur. C'est la classe CalendarContract et toutes ses sous-classes qui permettent d'accéder aux calendriers. Le principe est exactement le même que celui de ContactsContract, avec des noms de tables et de colonnes différents. Le code suivant permet par exemple de récupérer la liste complète des calendriers :

```
String[] projection = new String[] {
  CalendarContract.Calendars._ID,
  CalendarContract.Calendars.NAME
};
Cursor c = getContentResolver().query(CalendarContract.Calendars.
CONTENT_URI, projection, null, null, null);
```

Le code suivant, quant à lui, permet de récupérer les évènements à venir d'un calendrier donné :

```
String[] projection = new String[] {
  CalendarContract.Events._ID,
  CalendarContract.Events.TITLE,
  CalendarContract.Events.DTSTART,
  CalendarContract.Events.DTEND,
};
String condition = CalendarContract.Events.CALENDAR_ID + " = ? AND " +
          CalendarContract.Events.DTSTART + " >= ?";
Cursor c = getContentResolver().query(CalendarContract.Events.
CONTENT_URI,
                projection, condition,
                new String[] { calendarId, ""+System.current
TimeMillis()},
  CalendarContract.Events.DTSTART);
```

Le champ TITLE se passe de commentaires. Les champs DTSTART et DTEND contiennent les dates de début et de fin sous forme d'entier en millisecondes, qu'il sera facile de convertir dans un format textuel grâce aux classes Date et DateFormat. On précise également le champ DTSTART comme dernier paramètre de la méthode query afin de trier les évènements par date et heure.
Il existe de nombreux autres champs et tables contenant les différentes informations d'un calendrier. La classe CalendarContract.Reminders permet d'accéder à la table contenant les rappels programmés pour un évènement. CalendarContract.Attendees représente celle des personnes inscrites ou invitées. CalendarContract.Instances permet de lister l'ensemble des occurrences d'un évènement récurrent (tel un anniversaire ou un jour férié). Enfin, CalendarContract.ExtendedProperties permet de stocker des données supplémentaires pour un évènement particulier sous forme de paires clé/valeur. Ces différentes classes sont décrites dans la documentation de référence d'Android.
Mais la nouvelle application de calendrier permet également d'ajouter facilement de nouveaux évènements via l'envoi d'un intent, sans permission particulière.

```
// On utilise un objet Calendar pour générer une date/heure
Calendar cal = new GregorianCalendar(TimeZone.getTimeZone
("GMT"));
cal.set(2012, 4, 24, 0, 0, 0);
Intent intent = new Intent(Intent.ACTION_INSERT, Events.
CONTENT_URI);
intent.putExtra(Events.TITLE, "Google I/O 2012"); // Titre
de l'évènement
intent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, cal.
getTime().getTime()); // Date de début : on récupère le times
tamp sous forme de long. Le double appel est nécessaire car
le premier getTime() retourne un objet Date.
cal.set(2012, 4, 25, 23, 0, 0);
intent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME, cal.
getTime().getTime()); // Date de fin
intent.putExtra(Events.EVENT_LOCATION, "Moscone Center West,
San Francisco"); // Emplacement
intent.putExtra(Events.DESCRIPTION, "Le rendez-vous incontournable
des Google fans"); //Description
intent.putExtra(Intent.EXTRA_EMAIL, new String[] {"andy.
rubin@google.com"}); // Liste des e-mails d'invités
//intent.putExtra(Events.RRULE, ""); // Récurrence (chaîne
telle que définie dans le format iCalendar)
intent.putExtra(Events.ACCESS_LEVEL, CalendarContract.Events
.ACCESS_DEFAULT); // Accès (public ou privé)
intent.putExtra(Events.AVAILABILITY, CalendarContract.Events.
AVAILABILITY_FREE); // Disponibilité (est-il possible d'avoir
 d'autres évènements en même temps)
startActivity(intent);
```

[Fig.3].

API Voicemails

A l'instar du nouveau fournisseur de contenu pour les calendriers, l'API d'Ice Cream Sandwich propose également un fournisseur de contenu afin d'unifier la gestion des voicemails (les messages de répondeurs). Les applications utilisant ce type de contenu pourront utiliser la classe VoicemailContract et accéder aux deux tables VoicemailContract.Voicemails et VoicemailContract.Status. La permission ADD_VOICEMAIL est nécessaire pour l'ajout de nouveaux contenus dans cette base .

Communication entre appareils

Le package android.nfc ne contient pas beaucoup de modifications, mais celles-ci sont particulièrement intéressantes, puisqu'il est désormais possible d'envoyer des messages NDEF, et plus seulement d'en recevoir. Il devient donc possible de faire communiquer entre eux deux appareils Android à très courte distance (moins de 4 centimètres), par exemple pour échanger des liens, des contacts… [Fig.4].

Pour remplacer le Bluetooth et permettre des connexions peer-to-peer plus rapides et sur une plus grande distance entre des appareils Android, Ice Cream Sandwich apporte un nouveau package android.net.wifi.p2p au framework Android. Il contient toutes les API pour découvrir, initialiser, gérer les connexions etc., entre les appareils tournant sous Android 4.0 via leur connexion WiFi sans point d'accès intermédiaire.

Pour utiliser les fonctionnalités accessibles via une connexion WiFi pair à pair, une application devra déclarer les permissions ACCESS_WIFI_STATE, CHANGE_WIFI_STATE et même si techniquement, l'application n'accèdera pas nécessairement à internet, la permission INTERNET.

De nouveaux Intents sont maintenant régulièrement envoyés par le système afin de prévenir des éventuels changements dans l'état des connexions WiFiDirect. Par exemple, un changement dans la liste des appareils appairés lancera un Intent : WIFI_P2P_PEERS_CHANGED_ACTION, tandis qu'un changement dans les détails concernant l'état du WiFiDirect de l'appareil lui-même lancera : WIFI_P2P_THIS_DEVICE_CHANGED_ACTION. Une application souhaitant savoir quand le WiFiDirect est activé ou non et si l'état d'une connexion change devra pouvoir réagir aux Intent WIFI_P2P_STATE_CHANGED_ACTION (contenant un extra sur l'état du wifi) et WIFI_P2P_CONNECTION_CHANGED_ACTION (contenant des extras sur les infos des connexions réseaux).

Pour gérer des connexions via WiFiDirect, la classe la plus importante est sans aucun doute Wifip2pManager que l'on peut obtenir en appelant, comme tous les services systèmes, la méthode Context.getSystemService(WIFI_P2P_SERVICE). C'est cette classe qui contient toutes les méthodes pour bien gérer les connexions P2P. Elle permettra, entre autres, de découvrir les candidats à l'appairage, configurer des connexions, demander une liste des appareils déjà appairés, etc.

Les réponses aux requêtes étant asynchrones, les développeurs auront par conséquent besoin d'une instance de WifiP2pManager.ActionListener pour recevoir les callbacks onSuccess() et onFailure() indiquant si les actions entreprises l'ont été avec succès ou pas. Il existe d'autres interfaces utiles comme WifiP2pManager.PeerListListener dont le callback fournit une liste WifiP2pDeviceList d'objet de type WifiP2pDevice permettant d'obtenir des informations (comme le nom, l'adresse, le type…) sur les appareils à portée de WiFi.

Enfin, le dernier ajout sur les protocoles de communications concerne le support du matériel médical compatible Bluetooth, avec l'ajout du profil HEALTH et de la classe BluetoothHealth.

Nouveautés de l'API caméra

Plusieurs nouveautés font leur apparition pour gérer la caméra. La principale est la capacité de reconnaître les visages. Déjà présentée comme moyen de déverrouiller le terminal, les développeurs pourront également l'intégrer à leurs applications. La nouvelle méthode startFaceDetection() de la classe Camera enclenche la détection de visage. Il faut avant cela déclarer une instance de la classe Camera.FaceDetectionListener via la méthode setFaceDetectionListener() afin de capturer l'évènement.

La détection d'un visage déclenchera alors la méthode onFaceDetection() du listener, en lui passant en paramètre un tableau d'objets de type Camera.Face. Chacun de ces objets permet de connaître la position du visage (via un objet Rect), le taux de confiance de la détection (entre 1 et 100), un identifiant unique pour différencier le visage parmi plusieurs, ainsi que des objets

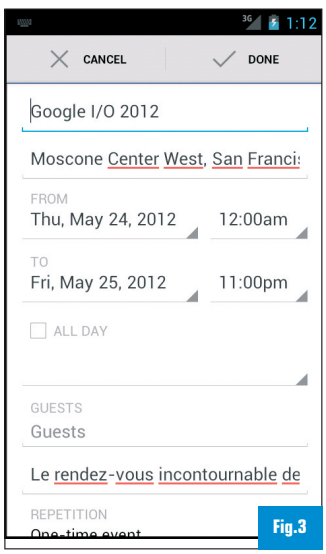


Fig.3

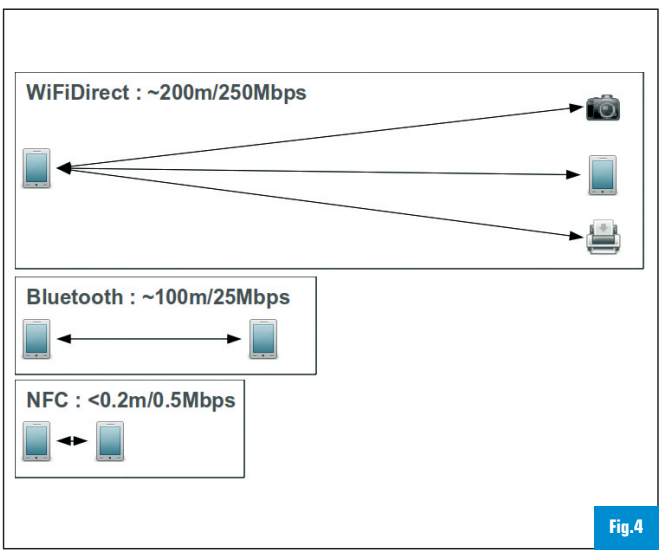


Fig.4

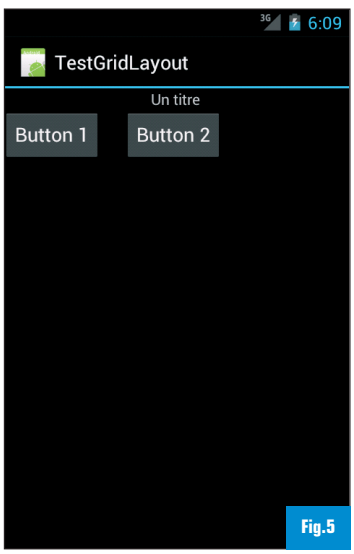


Fig.5

Points pour indiquer la position de la bouche et des deux yeux. Notez toutefois que les capacités de détection de visage peuvent varier d'un terminal à l'autre.
Une nouvelle classe Camera.Area permet également de définir sur quelles zones doit s'effectuer le focus. Un mode FOCUS_MODE_CONTINUOUS_PICTURE peut activer l'auto-focus en continu lors de la prise de photos. D'autres méthodes font leur apparition : isVideoSnapshotSupported() indique s'il est possible de prendre des photos lors de l'enregistrement de vidéos, avec la méthode takePicture(). setAutoExposureLock() et setAutoWhiteBalanceLock() verrouillent et déverrouillent les options d'exposition et de balance des blancs automatiques. Enfin, la méthode setDisplayOrientation() peut être appelée à n'importe quel moment pour changer l'orientation de l'appareil, et non pas obligatoirement avant l'initialisation de la caméra, comme c'était le cas avec les anciennes versions de l'API.

Vues et Interface Utilisateur

Un nouveau type de layout fait son apparition: le GridLayout. Ce nouveau ViewGroup permet d'organiser ses vues descendantes sur une grille rectangulaire. Ces vues spécifient quelle(s) ligne(s) et colonne(s) elles doivent occuper (une cellule pouvant occuper plusieurs lignes et/ou colonnes). Il est également possible de spécifier l'orientation du *GridLayout* de manière à déterminer un placement en séquence par défaut des vues (selon l'horizontale ou la verticale). Enfin, l'arrivée d'une nouvelle vue accompagne celle du *GridLayout:* la vue Space, qui permet de définir l'espace entre les différentes vues de la grille sans forcément utiliser le paramètre *layout_margin*. Par exemple, la déclaration XML d'un écran composé d'une première ligne contenant un titre centré sur la largeur et d'une deuxième ligne comprenant deux boutons espacés entre eux de 20dp pourrait se faire ainsi :

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res
/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal"
    android:rowCount="2"
    android:columnCount="3" >
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Un titre"
    android:layout_columnSpan="3"
    android:layout_gravity="center_horizontal" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button 1"
    android:layout_row="1"
    android:layout_column="0" />
<Space
    android:layout_row="1"
    android:layout_rowSpan="1"
    android:layout_column="1"
    android:layout_width="20dp"
    android:layout_height="wrap_content" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button 2"
    android:layout_row="1"
    android:layout_column="2" />
</GridLayout>
```

[Fig.5]

Le nouveau widget TextureView permet, à l'instar de SurfaceView déjà présent dans les précédentes versions, d'afficher des contenus vidéo ou OpenGL. Elle s'utilise en revanche comme une View à part entière qu'il est possible d'intégrer parmi d'autres widgets, et non pas comme une fenêtre distincte. Pour l'utiliser, il est nécessaire d'accéder à l'accélération matérielle du terminal. Celle-ci est désormais activée par défaut, et il n'est plus nécessaire d'utiliser le paramètre android:hardwareAccelerated de la balise <activity> dans le fichier AndroidManifest.xml, à moins de vouloir la désactiver explicitement. Un nouveau composant fait son apparition pour les écrans de préférences : SwitchPreference représente un composant à deux états (on et off). Il offre des possibilités similaires aux checkbox (CheckBoxPreference), les deux classes héritant d'ailleurs de TwoStatePreference. Un widget associé est également disponible avec la classe Switch. On peut déclarer un SwitchPreference sous forme de code :

```
SwitchPreference switchPref = new SwitchPreference(this);
switchPref.setKey("switch_preference");
switchPref.setTitle(R.string.title_switch_preference);
switchPref.setSummary(R.string.summary_switch_preference);
```

Ou bien en XML :

```
<SwitchPreference
  android:key="checkbox_preference"
  android:title="@string/title_switch_preference"
  android:summary="@string/summary_switch_preference_yes_no"
  android:switchTextOn = "YES"
  android:switchTextOff = "NO" />
```

Ces deux exemples sont extraits des nouvelles API Demos fournies avec le SDK.
Honeycomb introduisait l'objet PopupMenu. Celui-ci est désormais disponible également sur smartphones, avec quelques nouveautés mineures : la possibilité de récupérer le contenu du menu depuis un fichier XML via la méthode inflate(), ainsi que l'ajout du listener PopupMenu.OnDismissListener pour capturer l'évènement lorsque l'utilisateur ferme le menu sans sélectionner d'item.
Autre nouveauté issue d'Honeycomb, la classe ActionBar affiche une barre de navigation personnalisable en haut des activités. Il est possible d'y intégrer un logo, des menus, des onglets de navigation, des champs de recherche… Quelques nouveautés font leur apparition, comme la possibilité de séparer la barre en deux sur les petits écrans, soit par défaut l'un sous l'autre, soit en bas de l'écran avec l'option "splitActionBarWhenNarrow" à définir dans le champ android:uiOptions du tag <application> ou <activity>. La classe ActionProvider fournit un composant permettant

d'afficher une liste de choix en fonction des composants installés. Un bon exemple est la classe ShareActionProvider, qui permet d'afficher la liste complète des options de partage (via l'application mail, via Gmail, via SMS, via Google+…). Il est également possible de créer sa propre classe dérivée d'ActionProvider pour d'autres types de choix (navigateur web, application de téléphonie…). Plusieurs exemples sont fournis parmi les API Demos.

Avec la disparition des touches physiques, les tablettes sous Honeycomb ont intégré les boutons HOME et BACK directement sur leur interface via une barre système qui intègre aussi la barre de notification. Sur les nouveaux smartphones sans touches physiques, les boutons seront également intégrés à une barre de navigation en bas de l'écran, mais les notifications resteront en haut. De nouvelles propriétés permettent donc de contrôler l'affichage de cette nouvelle barre. La méthode View.setSystemUiVisibility() peut prendre en paramètre l'un des trois modes suivants :

- SYSTEM_UI_FLAG_VISIBLE, qui laisse visible la barre de navigation.
- SYSTEM_UI_FLAG_LOW_PROFILE, pour conserver la barre de navigation mais en masquer les icônes.
- SYSTEM_UI_FLAG_HIDE_NAVIGATION, pour masquer la barre de navigation, ce qui était impossible sous Honeycomb, mais celle-ci réapparaîtra à la moindre interaction utilisateur. Ce mode est donc plutôt dédié à la lecture de vidéo que pour des jeux en plein écran.

Gestion des entrées

Les méthodes d'entrée ont été étendues pour les stylets et les souris. Toutes les vues (héritant de la classe View) peuvent désormais capturer l'évènement "hover" lorsque le pointeur passe dessus, en définissant une instance de View.OnHoverListener. L'exemple suivant en montre brièvement le fonctionnement. En revanche, celui-ci ne peut actuellement pas être testé, puisqu'aucun appareil sous Ice Cream Sandwich n'est disponible et que l'émulateur n'est pas capable de gérer la souris.

```
public class HoverActivity extends Activity {
 @Override
 public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);

     final TextView text = (TextView)findViewById(R.id.text);
     Button button = (Button)findViewById(R.id.button);

     button.setOnHoverListener(new View.OnHoverListener() {

         @Override
         public boolean onHover(View v, MotionEvent event) {
             switch (event.getAction()) {
             case MotionEvent.ACTION_HOVER_ENTER:
                   text.setText("Entre sur le bouton");
                   break;
             case MotionEvent.ACTION_HOVER_EXIT:
                   text.setText("Sort du bouton");
                   break;
             case MotionEvent.ACTION_HOVER_MOVE:
                   text.setText("Bouge sur le bouton");
                   break;
             }
             return false;
           }
     });
 }
}
```

La classe MotionEvent bénéficie également de nouvelles méthodes. getToolType() permet de connaître le type de pointage utilisé parmi les constantes TOOL_TYPE_UNKNOWN, TOOL_TYPE_FINGER, TOOL_TYPE_MOUSE, TOOL_TYPE_STYLUS et TOOL_TYPE_ERASER. getButtonState() permet de connaître l'état des boutons de la souris, en utilisant un masque binaire entre les constantes BUTTON_PRIMARY, BUTTON_SECONDARY, BUTTON_TERTIARY, BUTTON_BACK et BUTTON_FORWARD. Enfin, la méthode getAxisValue() peut prendre en paramètres les nouvelles constantes AXIS_DISTANCE et AXIS_TILT pour connaître la distance et l'inclinaison du stylet. L'orientation peut être obtenue avec AXIS_ORIENTATION.

Services de correction orthographique et de Text-to-Speech

Il est désormais possible de développer des services de correction orthographique. Il faut pour cela demander la permission BIND_TEXT_SERVICE et dériver la classe SpellCheckerService, en y implémentant sa propre version de SpellCheckerService.Session. Celle-ci pourra alors analyser les mots tapés par l'utilisateur dans n'importe quelle application et proposer sa propre liste de suggestions. Les API de Text-to-Speech ont également été quelque peu revues, la méthode setEngineByPackageName() étant désormais dépréciée au profit d'un nouveau constructeur de la classe TextToSpeech prenant en paramètre le nom du moteur à utiliser. La liste des moteurs est désormais disponible via la méthode getEngines(), qui retourne une liste de TextToSpeech.EngineInfo. Le développement de services de Text-to-Speech est également simplifié grâce à la nouvelle classe TextToSpeechService, qu'il est possible d'hériter pour créer son propre service.

Effets OpenGL

Le package android.media.effect fournit plusieurs effets à appliquer sur des textures OpenGL ES 2.0. La création d'une nouvelle texture en appliquant un effet se fait facilement, comme dans l'exemple suivant :

```
EffectContext effectContext = EffectContext.createWithCurrent
GlContext();
EffectFactory factory = effectContext.getFactory();
if   (EffectFactory.isEffectSupported(EffectFactory.EFFECT_
FILLLIGHT)) {
Effect effect = factory.createEffect(EffectFactory.EFFECT_
FILLLIGHT);
effect.setParameter("strength", new Float(0.5f));
effect.apply(textureId, width, height, newTextureId);
 }
```

Les paramètres applicables varient d'un effet à l'autre et sont documentés avec la classe EffectFactory.

Contrôle des applications multimédias

Les interactions entre les lecteurs médias et les autres applications (widget, écrans de verrouillage…) sont facilitées par la nouvelle classe RemoteControlClient. Pour cela, il faut déclarer dans son application multimédia une instance de la classe, lui indiquer quels évènements seront interceptés avec la méthode setTransportControlFlags(), et l'enregistrer via la méthode MediaManager.registerRemoteControlClient(). Un BroadcastReceiver associé pourra alors intercepter les intent ACTION_MEDIA_BUTTON ainsi qu'un élément de type KeyEvent via un appel à getParcelableExtra(Intent.EXTRA_KEY_EVENT). Mais la communication fonctionne aussi dans l'autre sens, car RemoteControlClient permet également de diffuser des informations sur le média lu en appelant la méthode editMetaData() et en complétant l'objet généré de type RemoteControlClient.MetadataEditor. La nouvelle version de l'exemple RandomMusicPlayer fournie avec le SDK implémente RemoteControlClient.

Accessibilité

La gestion de l'accessibilité (à destination des personnes malvoyantes principalement) a été améliorée dans Ice Cream Sandwich. Il devient notamment possible d'obtenir une description vocale des différents éléments de l'écran en passant le doigt dessus d'où l'intérêt de compléter le champ "android:contentDescription" pour les vues des différents écrans (les widgets de type View).

Les méthodes pour la gestion de l'accessibilité ont été modifiées. Au lieu de redéfinir la méthode sendAccessibilityEvent() et d'y initialiser une instance d'AccessibilityEvent, c'est la méthode onInitializeAccessibilityEvent() qui est à redéfinir, et éventuellement onPopulateAccessibilityEvent() pour les types de vues qui nécessitent des informations textuelles. Il devient également possible d'ajouter à une vue une instance de classe de type View.AccessibilityDelegate via la méthode setAccessibilityDelegate(), dans laquelle seront également redéfinies onInitializeAccessibilityEvent() et onPopulateAccessibilityEvent(). On peut ainsi partager l'initialisation des AccessibilityEvent entre plusieurs vues différentes.

Le développement des services d'accessibilité est également enrichi par de nouvelles classes. Un AccessibilityEvent peut contenir plusieurs éléments de type AccessibilityRecord, qui contiennent les informations sur l'état de la vue associée. Des objets de type AccessibilityNodeInfo peuvent également être extraits des AccessibilityEvent et AccessibilityRecord, et renseignent sur les différents noeuds de la fenêtre. Chaque noeud peut correspondre a une seule vue, mais une vue peut définir son propre arbre de noeuds.

Sécurité

Jusqu'à présent, la gestion des certificats numériques, et notamment l'ajout de certificats racines (Autorité de certification) était un cauchemar dans Android : il était nécessaire de rooter le terminal pour être en mesure (via la modification d'un fichier système) de rajouter des certificats d'autorité au système. Cette limitation était un vrai frein à l'adoption d'Android en entreprise. Ice Cream Sandwich apporte une réponse à cette problématique avec la classe KeyChain. L'arrivée de cette classe dans l'API signifie également, et surtout, qu'il est désormais possible d'ajouter des certificats d'autorité personnels au keystore Android depuis les paramètres Android (Settings _ Security _ Install from SD card) ; pour cela, il est nécessaire de déposer le certificat concerné sur la partition "sdcard" du terminal.

La bonne nouvelle est que le navigateur fourni avec Android utilise les certificats d'autorité ajoutés au "keystore", évitant ainsi les alertes de sécurité lorsqu'on se connecte à un site HTTPS dont le certificat est signé par une autorité de certification concernée.

Gestion du VPN

Ice Cream Sandwich apporte une nouveauté importante pour la gestion des VPN (Virtual Private Network : Réseau Privé Virtuel) : avant ICS, il était indispensable de rooter le terminal Android pour y installer un client VPN tiers. Désormais, la classe android.net.VpnService permet à une application de mettre en place un tunnel VPN sans que le terminal soit rooté.

Cette nouvelle classe permet de procéder aux différentes actions qui demandent normalement de disposer de privilèges sur la plateforme, notamment :

- la création d'une interface réseau virtuelle, de son adressage et du routage associé
- l'établissement d'un tunnel sécurisé
- la gestion des paquets entrant/sortant

Laisser une application intercepter l'ensemble des paquets entrant et sortant du terminal représente un risque évident en termes de sécurité. C'est pourquoi l'implémentation de la classe VpnService prévoit un certain nombre d'actions pour en limiter les effets :

- L'utilisateur doit explicitement accepter la création d'une connexion VPN, ainsi, une application ne peut monter de connexion VPN à l'insu de l'utilisateur.
- Il ne peut y avoir qu'une seule connexion VPN à la fois.
- Une notification système est présente pendant la durée de la connexion. Cela permet à l'utilisateur de savoir qu'il y a une connexion VPN en cours
- Une boîte de dialogue système permet d'afficher les informations concernant la connexion VPN en cours et donne la possibilité de se déconnecter

Une nouvelle permission apparaît pour gérer cette nouvelle fonctionnalité : BIND_VPN_SERVICE

Peu d'applications Android auront besoin de cette fonctionnalité mais cela va certainement permettre de voir apparaître un certain nombre de clients VPN sur le market, par exemple un client OpenVPN, sans qu'il soit nécessaire de rooter le terminal.

Conclusion

Comme nous venons de le voir, l'API d'ICS a subi de nombreuses modifications dont la plupart sont des améliorations significatives autant en termes de nouvelles fonctionnalités que d'ajouts demandés par la communauté. Il est évident que le survol de ces nouveautés n'est pas suffisant pour faire de vous des experts mais nous espérons vous avoir donné l'envie d'investiguer plus avant cette nouvelle API.

- Léo **Cacheux** - *Genymobile*
- Daniel **Fages** - *Genymobile*
- Jérémy **Vagnet** - *Genymobile*
- Paul **Marois** - *Genymobile*

Migrer vos applications vers Android 4.0

Répondant au doux nom d'Ice Cream Sandwich, la nouvelle mouture d'Android est une version majeure de la plateforme, unifiant le développement pour tous les appareils, l'embarquant des smartphones aux tablettes et TV connectées. Afin de tirer pleinement parti des avancées de cette version 4.0, il va être nécessaire de migrer les applications existantes. Cette phase de migration s'avérera plus ou moins lourde suivant le niveau d'intégration que le développeur va viser avec cette nouvelle mouture.

La compatibilité ascendante entre versions est une norme tacite au sein de l'écosystème Java depuis l'apparition des premiers JDK, et Android ne fait pas exception à la règle, puisque la firme de Mountain View a adopté cette approche dès le début de la plateforme. De fait, une application tournant actuellement sur une version antérieure à la 4.0 sera de facto opérationnelle sur Ice Cream Sandwich. Néanmoins, et puisque cette nouvelle mouture est une version unifiée qui vient élargir la gamme des périphériques supportés, il va sans dire qu'une application existante gagnera énormément à être migrée afin d'offrir la meilleure expérience utilisateur possible aux possesseurs d'appareils à écrans extra larges tels les tablettes. La phase de migration d'une application sera plus ou moins lourde suivant le but poursuivi. La possibilité la plus légère aura pour but principal d'optimiser l'application existante pour les tablettes alors qu'une seconde stratégie plus lourde va nécessiter des modifications profondes sur le code existant. Elle s'avère appropriée pour la réalisation de nouveaux développements Android. Quelle que soit la stratégie retenue, les premières étapes de migration vont être communes. Tout d'abord, il faut vérifier que l'application fonctionne toujours correctement sur Ice Cream Sandwich via une série de tests réalisés sur l'émulateur Android par exemple. Une fois cette étape initiale franchie, le développeur va pouvoir bénéficier au sein de son application du nouveau thème "Holographic" ainsi que de la nouvelle police de caractères du système nommée Roboto. Ceci est fait en modifiant au sein du manifest l'attribut android:targetSdkVersion de la balise <uses-sdk> pour y mettre la version 14 du SDK (code d'Android 4.0) afin de signifier au système que l'application utilise la dernière mouture de l'OS. Ce changement n'impacte pas la version minimale du SDK que cible l'application dans le cas d'une migration légère d'un existant.

Supporter des écrans extra larges

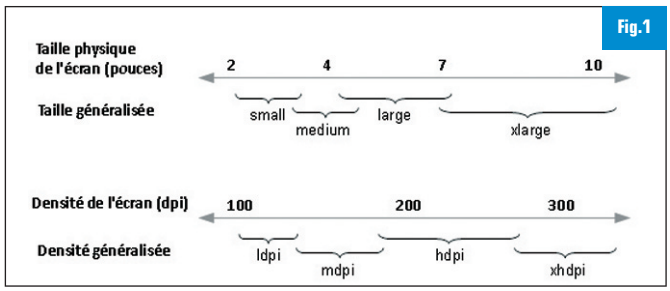
Les nouveaux écrans extra larges équipant les tablettes doivent désormais être au cœur des préoccupations des développeurs. Afin de simplifier le support du multi-écran au sein des applications, Android propose un certain nombre de concepts et d'abstractions au centre desquelles on retrouve la taille des écrans et leur densité. Exprimée en dpi, la densité d'un écran correspond à la quantité de pixels qu'une zone physique d'un écran peut contenir. Au sein du système, ces 2 paramètres sont regroupés dans les ensembles généralisés suivants :

- Taille des écrans : small, normal, large et xlarge
- Densité : ldpi (faible densité), mdpi (moyenne), hdpi (haute), xhdpi (extra)

Le schéma de la [Fig.1] montre la correspondance que le système réalise au runtime entre ces données généralisées et les écrans physiques des appareils. En outre ces tailles d'écrans généralisées ont une résolution minimum associée, elle-aussi exprimée en dp, toujours dans un souci d'indépendance vis-à-vis des données physiques des écrans.

L'arrivée des écrans extra larges s'accompagne de nouvelles catégories pour le support du multi-écran au sein des applications. Les qualificateurs xlarge et xhdpi permettent de cibler ces écrans tout comme l'utilisation désormais autorisée des informations liées à la résolution des écrans. En ce sens, il est bon de noter que la résolution minimum pour les écrans extra larges est d'au moins 960dp par 720dp. Pas toujours supportée sur les applications destinées aux smartphones, l'orientation de l'écran en mode paysage est la norme sur les tablettes. De fait, il devient primordial de définir des layouts adaptés au mode paysage pour chacun des écrans de son application en utilisant le qualificateur xlarge-land à la suite du nom du dossier layout au sein des ressources.

De plus, il s'avère intéressant de cibler les layouts suivant la taille de l'écran. On pourrait ainsi définir pour une activité donnée un



Correspondance données généralisées et physiques pour les écrans

layout à destination des smartphones et un à destination des tablettes en utilisant les qualificateurs comme suit :

```
res/layout/myActivity.xml    # smartphones
res/layout-sw600dp/myActivity.xml    # tablettes
```

Le qualificateur sw600dp spécifiant que le layout est à utiliser pour des écrans ayant une résolution en largeur d'au moins 600 dp. Au niveau des ressources de type drawables, le développeur doit désormais fournir des images suivant le ratio 3/4/6/8 entre les 4 catégories généralisées de densité d'écrans. Ainsi, si une image de votre application est de taille 48*48 en mdpi qui est la référence, alors il faudra fournir une image équivalente en 96*96 dans le dossier res/drawable-hdpi.

Outre ces ressources dédiées aux écrans extra larges à prendre en compte, il convient de rappeler que les bonnes pratiques pour réaliser des interfaces utilisateurs indépendantes et donc multi-écrans sont toujours d'actualité et se résument à 4 points clés :

1. Utiliser wrap_content, fill_parent ou des unités en dp pour les dimensions des layouts
2. Ne pas utiliser de valeurs en pixels dans le code applicatif
3. Ne pas utiliser l'AbsoluteLayout
4. Utiliser au maximum les ressources alternatives pour les drawables

Le respect de ces points permet de réaliser des applications offrant une expérience utilisateur optimisée au plus grande nombre d'appareils. Utilisé avec une large gamme de configurations d'écrans différentes, l'émulateur Android permet de valider tout ceci. Cette phase de tests assure que l'application est optimisée à la fois en mode portrait et en mode paysage, qu'elle tire profit au maximum des écrans larges avec des layouts adaptés utilisant l'espace supplémentaire et enfin fonctionne parfaitement sur des petits écrans. En effet, le support des tablettes ne doit pas faire oublier que la base utilisateur du système est majoritairement sur des appareils dotés d'écrans plus petits.

Les optimisations détaillées jusqu'ici sont applicables sur une application sans gros changements sur le code existant et limitent ainsi le risque de régressions. Les modifications présentées dorénavant concerneront les applications adoptant une stratégie de migration plus importante vers Ice Cream Sandwich.

Utiliser l'Action Bar

Apparue avec la version 3.0 de l'OS, l'Action Bar est un widget qui vient remplacer la traditionnelle barre de titre située en haut de l'écran des applications. Elle offre une meilleure interaction aux utilisateurs entre l'application et les différentes fonctionnalités qu'elle propose. Ainsi, elle facilite et fluidifie la navigation entre celles-ci. En outre, elle s'avère être le remplaçant du menu options utilisé jusqu'ici mais qui est voué à disparaître à terme puisque les futurs appareils Android ne proposeront plus forcément de bouton "Menu". Par défaut, la barre d'action inclut le logo de l'application sur son côté gauche, le titre de l'activité et éventuellement sur son côté droit, une liste proposant les items définis au sein du menu options de l'activité. Certains d'entre eux pouvant être directement intégrés au sein de l'Action Bar via l'attribut android:showAsAction positionné à "ifRoom" lors de la déclaration de l'item au sein du fichier XML du menu. La méthode callback appelée lors du clic sur un de ces items demeure toujours onOptionsItemSelected() de la classe Activity. D'autre part, cette barre d'action est idéale pour afficher des vues plus interactives comme une boîte de recherche par exemple. Pour cela, on rajoutera dans la balise de déclaration de l'item l'attribut android:actionLayout pointant vers le layout utilisé par cette vue action et on associera les listeners adéquats à la création des items :

```
public boolean onCreateOptionsMenu(Menu menu) {
   getMenuInflater().inflate(R.menu.options, menu);
   SearchView searchView = (SearchView) menu.findItem(R.id.
menu_search).getActionView();
   // … association du listener de click pour la recherche …
   return super.onCreateOptionsMenu(menu);
}
```

Enfin, cette barre d'action peut être employée pour afficher une élégante barre de navigation entre onglets pointant chacun vers un fragment d'une activité.

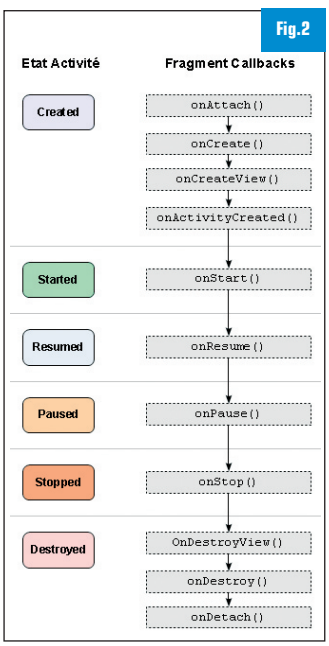
Pour des activités nécessitant une utilisation totale de l'espace offert par l'écran, il est bien évidemment possible de la retirer en passant par la déclaration de l'activité au sein du manifest via l'application d'un thème dédié comme suit :

```
<activity android:theme="@android:style/Theme.Holo.NoActionBar" />
```

Il reste toujours possible de réaliser ce retrait dynamiquement via le code suivant placé au sein de l'activité concernée : getActionBar().hide() .

Diviser les activités en fragments

L'apparition des tablettes Android aura mis en évidence la nécessité de pouvoir découper les activités en plusieurs parties modulaires afin d'offrir des IHM plus réactives et flexibles. Fort de ce constat, les fragments d'activités font leur apparition. Un fragment représente un comportement ou une portion de l'interface utilisateur à l'intérieur d'une activité. N'existant qu'au sein d'une activité, un fragment est un composant modulaire réutilisable entre plusieurs activités. En ce qui concerne son cycle de vie, un fragment va être instancié uniquement lorsqu'il sera attaché à une activité. Devenant alors actif, il n'est détruit que lorsqu'il est détaché de l'activité que ce soit par un retrait de l'interface de cette dernière ou bien sa destruction. Tout au long du cycle de vie de l'activité parent d'un fragment, des méthodes callbacks sont appelées permettant l'interaction entre le fragment et son activité de rattachement [Fig.2].



Lien entre le cycle de vie d'une Activité et d'un Fragment

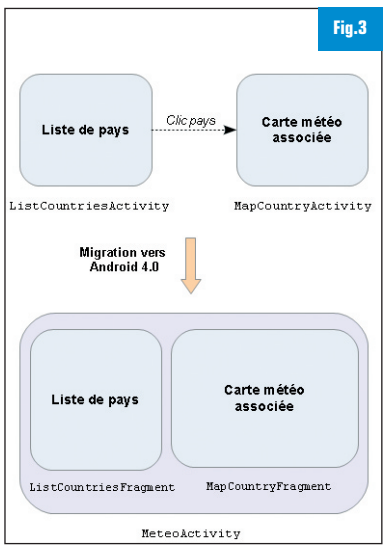
Modélisés au travers de la classe Fragment, les fragments bénéficient d'une API dédiée. Leur gestion se fait via un manager récupérable au sein de chaque activité par un

appel à getFragmentManager(). Les opérations les concernant sont encapsulées au sein de transactions de type FragmentTransaction, ce qui permet d'effectuer d'éventuels rollbacks ou bien d'ajouter des fragments à la pile des fragments encore disponibles, via la méthode addBackStack(), lors de l'appui sur le bouton "Back" d'un appareil Android.

Prenons l'exemple d'une application météo avec une activité listant un ensemble de pays et qui selon le clic réalisé sur l'un d'entre eux affiche la carte météo correspondante, au sein d'une autre activité. La migration vers Android 4.0 va consister à découper le travail réalisé par ces 2 activités au sein de 2 fragments qui pourront éventuellement être affichés dans une même activité sur une tablette afin de profiter au mieux de l'espace offert par l'écran [Fig.3].

Le layout de l'activité MeteoActivity, incluant ces 2 fragments, a l'allure suivante :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/
res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <fragment
        android:id="@+id/listFragment"
        android:layout_width="wrap_content"
        android:layout_weight="0.2"
        android:layout_height="fill_parent"
        class="app.meteo.ListCountriesFragment " />

    <fragment
        android:id="@+id/mapFragment"
        android:layout_width="wrap_content"
        android:layout_weight="0.8"
        android:layout_height="fill_parent"
        class="app.meteo.MapCountryFragment" />

</LinearLayout>
```



Cas d'utilisation de 2 fragments au sein d'une activité

Héritant de la classe Fragment, ces objets vont charger leurs layouts respectifs lors de l'appel au callback onCreateView(). Avec cet exemple, les fragments sont rattachés de manière statique à l'activité mais il est également envisageable de ne rattacher que le fragment ListCountriesFragment dans un premier temps puis, lors d'un clic sur un pays, de charger et d'ajouter dynamiquement le fragment à l'activité MeteoActivity via l'utilisation du FragmentManager.

Utiliser la nouvelle API Animation

Disponible sur la plateforme depuis plusieurs versions, l'animation des vues aura été mise à profit par les développeurs afin de réaliser des effets de transition entre activités notamment. Cependant, cette API a montré ses limites et notamment le fait que seule la manière dont une vue est dessinée est affectée par l'animation et que la valeur de la propriété animée n'est pas mise à jour. Ainsi, si un bouton est animé pour être déplacé au sein d'un écran, ses coordonnées de positionnement ne seront pas mises à jour. Afin de corriger ce problème, le système a été doté d'un framework robuste aux capacités étendues offrant la possibilité d'animer presque avec une mise à jour des valeurs des propriétés animées. Le développeur peut définir plusieurs caractéristiques pour une animation telles que sa durée, son nombre de répétitions ou bien la regrouper avec d'autres animations pour les jouer séquentiellement ou en parallèle.

Le point d'entrée de l'API est la classe ObjectAnimator qui décrit l'animation affectant une propriété. Pour réduire le code à écrire et simplifier l'utilisation de ces animations sur les vues, la classe ViewPropertyAnimator est proposée. Cette dernière offre des méthodes spécifiques pour agir sur les propriétés d'une vue utilisées pour l'animer. Ainsi, réaliser une translation d'une vue en diagonale vers le bord droit de l'écran peut être codé comme suit :

```
int toX = container.getWidth() - myView.getWidth();
int toY = container.getHeight() - myView.getHeight();
ViewPropertyAnimator animator = myView.animate() ;
animator.x(toX).y(toY);
```

On note au passage que l'appel chaîné des méthodes x() et y() permet de jouer en parallèle l'animation de ces 2 propriétés. On le voit, cette API s'avère puissante et concise. Le revers de cette puissance est un niveau de performance en deçà du système d'animation de vues existant. Ainsi, si une application possède déjà des animations et que le développeur ne voit pas d'intérêt de migrer vers la nouvelle API cela peut être une bonne solution de garder l'existant et d'utiliser la nouvelle API uniquement pour des nouveaux besoins ou des besoins non couverts par l'ancienne API. En sus, les animations de propriétés peuvent également être décrites au sein de fichiers XML dédiés placés au sein du dossier res/animator pour les distinguer des animations destinées à l'ancienne API. La définition au sein d'un fichier XML est une bonne solution pour avoir des animations réutilisables entre différentes vues et objets. Le code animant le fragment d'activités MapCountryFragment si celui-ci est ajouté dynamiquement à l'activité MeteoActivity, lors d'un clic sur un pays, est présenté ci-dessous :

```
FragmentTransaction ft = getFragmentManager().beginTransaction();
ft.setCustomAnimations(R.animator.slide_left_enter, R.animator.
slide_left_exit);
ft.add(mapCounttryFragment, MAP_FRAGMENT);
ft.commit();

// Animation slide_left_enter …
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <objectAnimator
        android:interpolator="@android:interpolator/decelerate
```

```
_quint"
       android:valueFrom="0dp" android:valueTo="-100dp"
       android:valueType="floatType"
       android:propertyName="translationX"
       android:duration="@android:integer/config_mediumAnim
Time" />
</set>
```

Activer l'accélération matérielle

La plateforme Android bénéficie dorénavant d'un support de l'accélération matérielle implémenté via le moteur de rendu OpenGL. Elle vient booster les performances de la plupart des opérations graphiques 2D réalisées par le système et profite de fait aux applications. Il en résulte des animations plus fluides, un scrolling plus rapide et de meilleurs temps de réponse aux actions utilisateurs. Néanmoins, l'accélération matérielle n'est pas supportée par certaines opérations 2D et il peut donc s'avérer nécessaire de ne l'activer que pour les parties d'une application pouvant en profiter sans risques. C'est en ce sens que le système offre une granularité sur 4 niveaux pour le contrôle de l'accélération matérielle d'une application :

```
¸ Application, activable via la balise <application android:
hardwareAccelerated ="true" … /> au sein du manifest
¸ Activité, activable à la définition d'une activité comme suit :
 <activity android:hardwareAccelerated="true" … />
¸ Fenêtre, activable dans le code via : getWindow().addFlags
(WindowManager. LayoutParams.FLAG_HARDWARE_ACCELERATED);
¸ Vue, activable dans le code via : myView.setLayerType(View.
 LAYER_TYPE_HARDWARE, null);
```

Le regain de performance et de réactivité apporté par l'accélération matérielle aux applications ne doit pas pour autant faire oublier les bonnes pratiques préconisées pour l'utilisation des opérations 2D telles que la réduction du nombre de vues utilisées par une application ou bien d'éviter de créer des objets Paint ou Path à chaque passage dans les méthodes de rendu, là où une initialisation statique fait l'affaire et soulage le travail du Garbage Collector. Bref, l'optimisation des applications et du rendu des IHM doit demeurer un souci constant du développeur.

Conclusion

La migration des applications existantes vers Android 4.0 s'avère être une tâche plus ou moins lourde selon le niveau d'intégration désiré avec les nouvelles possibilités de la plateforme. Dans tous les cas, l'effort devra être mis sur le support des écrans très larges afin de fournir aux utilisateurs la meilleure expérience possible quel que soit le terminal Android utilisé. En outre, l'arrivée des tablettes aura ouvert la voie à de nombreuses nouveautés au niveau de l'interface et à un nouveau modèle d'interaction utilisateur visant à rendre les IHM plus réactives et flexibles. Comme le souligne cet article, la migration vers cette version d'unification met l'accent sur l'interface utilisateur des applications. Néanmoins, il est bon de rappeler qu'Ice Cream Sandwich ne se limite pas à cela et amène également sont lot de nouvelles fonctionnalités qui promettent d'offrir aux applications un grand nombre de nouvelles possibilités allant du NFC au Wi-Fi direct en mode P2P en passant par Android Beam.

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

Développement d'applications unifiées sous Android 4.0

Les développeurs d'applications Android se sont pendant longtemps heurtés à la variété de produits utilisant la plateforme mobile. En effet, ne pas savoir sur quel support une application est utilisée devient problématique pour fournir un confort d'utilisation optimal.

Android 4.0 apporte un élément de réponse à ce problème en proposant la première version du système unifiant le support des smartphones et des appareils de type tablette. Cette nouvelle version permet donc de créer des applications uniques, réduisant la rupture instaurée depuis le lancement d'Android 3.0 destiné à un usage sur tablettes uniquement. Cette différenciation qui contraignait les développeurs à maintenir de multiples versions du code de leurs applications étant révolue, cet article propose un premier état des lieux du développement d'une application fonctionnant sur mobiles et tablettes.

Sélectionner les appareils autorisés à utiliser l'application et fournir les ressources adéquates

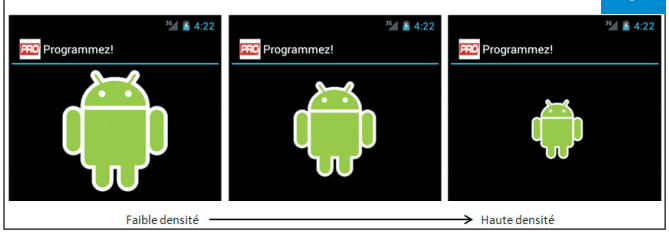
La première étape pour développer une application supportant différents formats d'appareils est de déclarer les tailles d'écran que l'application supportera dans le fichier AndroidManifest.xml. Pour cela, il suffit de renseigner la balise <supports-screens>. Par exemple, pour une application supportant à la fois mobiles et tablettes :

```
<supports-screens android:
android:smallScreens="true"
android:normalScreens="true"
android:largeScreens="true" />
```

Il faut ensuite fournir plusieurs formats pour les ressources affichées sur des écrans de différentes densités (pour une zone de même taille, deux écrans de densité différente ne proposent pas la même quantité de pixels). Par exemple, une image destinée à être utilisée sur un écran de faible densité occupe moins de place pour s'adapter à un écran plus dense. Il est donc recommandé de définir cette ressource pour chaque densité d'écran afin de conserver une taille d'affichage constante [Fig.1].

Fournir ces ressources consiste simplement à créer la structure de dossiers adéquate, le système s'adaptant ensuite automatiquement à l'écran utilisé [Fig.2]. Par défaut, Android redimensionne l'interface de l'application pour qu'elle s'ajuste à l'écran utilisé. Cependant, il est dans certains cas nécessaire d'adapter plus en finesse la taille et la position de certains éléments plutôt que de laisser le système agir de lui-même. Pour ce faire, il faut fournir à l'application des fichiers d'interface (« layouts ») spécifiques aux tailles d'écran déclarées dans le fichier AndroidManifest.xml. Ainsi, pour chaque format, on ajoute un dossier portant un nom spécifique et comportant les ressources associées. Le système choisit alors automatiquement le layout approprié.

Fig.1



Avant l'arrivée d'Android 3.2, quatre catégories permettaient de distinguer les écrans en fonction d'un seuil minimal sur leur résolution exprimée en « dp » (unité permettant de caractériser la résolution de l'écran de manière indépendante de sa densité) :

```
xlarge : 960dp x 720dp
large : 640dp x 480dp
normal : 470dp x 320dp
small : 426dp x 320dp
```

Cette hiérarchie est devenue insuffisante pour spécifier autant de layouts que de formats d'écrans existants. La méthode recommandée depuis Android 3.2 est de répartir ces fichiers selon la largeur minimale nécessaire en « dp ». Par exemple, pour une application proposant des layouts différents pour mobiles, tablettes 7 pouces et tablettes 10 pouces, il faut définir trois dossiers :

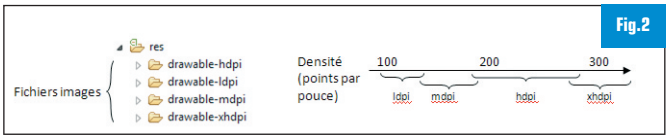
```
res/layout/              # Pour mobiles (par défaut)
res/layout-sw600dp/      # Pour les tablettes 7 pouces
res/layout-sw720dp/      # Pour les tablettes 10 pouces
```

Cette première approche améliore considérablement le rendu d'une application sur plusieurs supports mais ne permet pas d'optimiser totalement le potentiel de chaque type d'appareil : une tablette de par son écran de grande taille permet l'affichage de plus de contenu simultanément alors que ce même contenu doit être réparti sur plusieurs vues pour un mobile par exemple.

Optimiser l'interface selon l'appareil en utilisant les fragments

Instaurés depuis la version 3.0, les fragments permettent de décomposer les fonctionnalités d'une application en plusieurs blocs affichables et limitent ainsi les changements d'activité, notamment pour une application composée de plusieurs volets. Les fragments sont particulièrement utiles pour un usage sur des

Fig.2



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 Diffusion, GLIE - 17 chemin des Boulangers 78926 Yvelines cedex 9 - ou par fax : 01 55 56 70 20

1 an : 50€ *au lieu de 60€, prix au numéro* *(Tarif France métropolitaine)* **- Autres destinations :** *CEE et Suisse : 60€ - Algérie, Maroc, Tunisie : 65€ , Canada : 80€ - Dom : 75€ Tom : 100€*

10 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

appareils disposant d'un grand écran car ils permettent de combiner ou d'interchanger facilement des parties de l'interface.
Afin d'être mis en place efficacement, les fragments doivent être programmés de manière modulaire et réutilisable. Chaque fragment décrit son propre cycle de vie, gère lui même les interventions de l'utilisateur et dispose de ses propres accesseurs.
Sous Android 4.0, ces éléments représentent également la solution pour le développement d'une application s'adaptant au profil d'appareil utilisé : la conception de l'interface doit alors être pensée comme un ensemble de fragments agencés différemment en fonction de la taille d'écran disponible.
Ce comportement s'illustre facilement avec l'exemple d'un lecteur de flux RSS : un appareil disposant d'un petit écran affiche dans un premier temps une vue composée d'une liste d'articles. Lors de la sélection d'un article, l'affichage bascule sur une vue laissant place au contenu de cet article. Avec un grand écran, ces deux fragments sont affichés côte à côte, la sélection d'un élément de la liste ayant pour effet la mise à jour de la vue de détail de l'article.
D'un point de vue technique, il y a deux façons de produire ce comportement :

- Utiliser une activité unique regroupant les deux fragments, et gérer dynamiquement l'affichage de ceux-ci selon la taille de l'écran depuis le code de l'activité. Cette approche, bien que fonctionnelle, peut rapidement devenir complexe à maintenir pour une application proposant de multiples agencements de fragments (afin d'obtenir la taille de l'écran il suffit d'appeler : getResources().getConfiguration() qui renvoie un objet de type Configuration proposant un champ screenlayout).
- Utiliser plusieurs activités et plusieurs fragments : on utilise une activité principale dont le layout inclut les deux fragments pour un grand écran et un seul fragment pour les petites tailles. Le système choisit alors automatiquement le layout approprié. Lorsque l'activité principale est sollicitée elle doit, si le second fragment n'est pas présent, instancier une seconde activité hébergeant celui-ci sinon le mettre à jour.

Dans le cas de l'application proposant un lecteur de flux RSS, voici ce que doit réaliser la seconde méthode : [Fig.3].
Il faut donc définir deux « layouts » pour l'activité principale (activité 1) : le premier dédié aux petits écrans ne comportant que le fragment affichant la vue de liste des articles. Celui-ci est déclaré dans le dossier res/layout/

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/
res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
       <!--Fragment de liste -->
      <fragment class ="com.programmez.lecteurRSS.list.List
Fragment"
          android:id="@+id/list_fragment"
          android:layout_width="match_parent"
          android:layout_height="match_parent">
     </fragment>
</FrameLayout>
```

Le second destiné aux grands écrans, affichant à gauche la liste d'articles et à droite la vue de détail, placée dans le répertoire res/layout-sw600dp/

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
    <!--Fragment de liste -->
    <fragment class ="com.programmez.lecteurRSS.list.List
Fragment"
        android:id="@+id/list_fragment"
        android:layout_height="match_parent"
        android:layout_weight="45">
    </fragment>
    <!--Fragment de détail -->
    <fragment class ="com.programmez.lecteurRSS.list.Viewer
Fragment"
        android:id="@+id/view_fragment"
        android:layout_height="match_parent"
        android:layout_weight="55">
    </fragment>
</LinearLayout>
```

Lorsque l'utilisateur sélectionne un élément de la liste d'articles, le comportement de l'activité principale (activité 1) est le suivant :

- Si le fragment de détail est disponible dans le « layout » courant, l'activité 1 ordonne au fragment de détail de se mettre à jour
- Si le fragment de détail n'est pas présent, l'activité 1 instancie l'activité 2 qui prend alors le relais et affiche la vue de détail.

L'activité 2 n'étant appelée que dans le cas de petits écrans, son

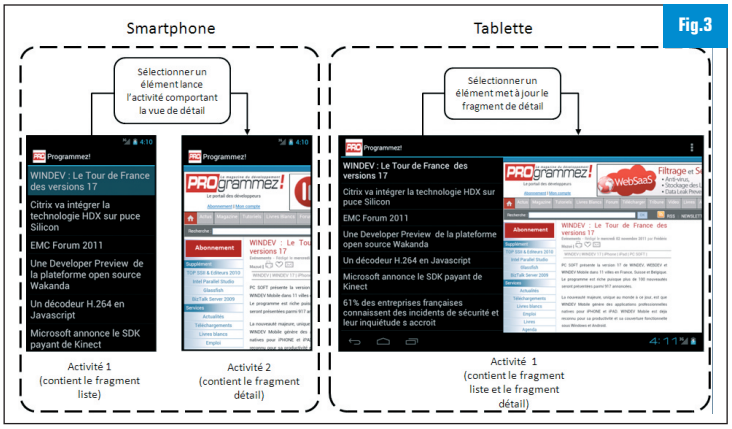


Fig.3

Fig.4

layout ne sera déclaré qu'une seule fois dans le répertoire correspondant. Les interfaces par défaut étant définies, l'étape suivante consiste à déclarer la fonction de l'activité principale (activité 1) qui écoute le signal émis lorsqu'un élément de la liste est sélectionné :

```
public void onItemSelected(String Url) {
// on cherche le fragment permettant l'affichage de la vue de détail
ViewerFragment viewer = (ViewerFragment) getFragmentManager
().findFragmentById(R.id.view_fragment);

if (viewer == null) {
// le fragment n'est pas dans le « layout », on lance l'activité
 2 avec le paramètre (ici l'url de l'article)
Intent showContent = new Intent(this,ViewerActivity.class);
showContent.setData(Uri.parse(Url));
startActivity(showContent);
} else {
// le fragment est disponible dans le « layout » courant : on
 le met à jour avec le paramètre
viewer.updateUrl(Url);
}
}
```

L'utilisation de fragments est essentielle pour produire une application multi-supports car elle permet de proposer une interface ajustée aux différents appareils du marché.

Utilisation de la barre d'actions

Egalement introduite depuis Androïd 3.0, la barre d'actions (« ActionBar ») offre la possibilité d'obtenir aisément une barre de menus s'intégrant à l'interface du système et permettant d'effectuer diverses actions grâce à des boutons, ou de changer le mode d'affichage via un menu d'onglets. Son fonctionnement n'évolue pas dans cette nouvelle mouture de l'environnement. Cependant, quelques ajustements sont de rigueur pour qu'une application supportant plusieurs formats d'écran en tire le meilleur parti.

En ce qui concerne les boutons d'action : une fois définis, ceux-ci sont stockés par défaut dans le menu options à droite de la barre d'actions ou accessibles depuis une touche de l'appareil. Il est également possible de forcer ces boutons à apparaître directement dans la barre d'actions en utilisant lors de leur déclaration l'attribut : android:showAsAction="always ".

Dans le cas d'une application combinant le support des mobiles et des tablettes, il faut éviter au maximum de forcer l'apparition de ces menus dans la barre supérieure car la place est limitée. Pour cela, une option est préférable : android:showAsAction="ifRoom".

Le système ajuste alors, selon la place disponible, les éléments à afficher. Il est également recommandé de fournir une icône pour chaque action en plus de son texte descriptif et de compléter sa déclaration : android:showAsAction="ifRoomIwithText". Ainsi, selon la largeur disponible, le descriptif de l'action est automatiquement masqué [Fig.4]. Viennent ensuite les onglets de navigation qui permettent de basculer entre les modes de l'application. Bien qu'il soit possible de créer ces onglets manuellement, il est recommandé dans le cadre du développement d'une application multi-support d'utiliser ceux standardisés depuis l'arrivée du SDK d'Android 3.0 car ils s'adaptent automatiquement à la taille de l'écran. Sur un mobile, les onglets se placent sous la barre d'actions et il est possible de faire glisser la liste latéralement lorsque leur nombre devient trop important. Sur tablette, les onglets se logent dans la barre d'actions. Ceux-ci sont automatiquement rassemblés si la taille ne permet pas de les afficher sur une seule ligne. Android 4.0 ajoute également une nouveauté permettant de solutionner le problème de la faible quantité de boutons d'action affichables dans la barre supérieure. Il s'agit de l'option uiOptions="splitActionBarWhenNarrow" qui se déclare dans le manifeste de l'application ou de l'activité. Sur de grands écrans, cette option n'a pas d'effets alors que sur des écrans de smartphones, les boutons d'action se retrouvent projetés en bas de l'écran.

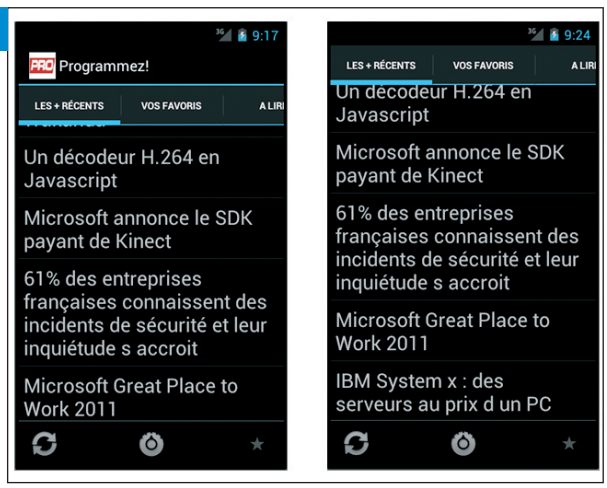
Il est également bon d'appeler la méthode « setDisplayShowHomeEnabled(false) » afin d'enlever la barre de titre et gagner quelques précieux pixels d'affichage lorsque cela est possible.

Android 4.0 est le premier essai de la plateforme visant à rassembler le monde des tablettes et celui des smartphones. Cette mouture reprend en grande partie les éléments instaurés depuis la version 3.0 dédiée aux tablettes. Ceci permet aux développeurs souhaitant proposer des applications uniformisées un travail standardisé et d'harmoniser les interfaces au travers de la multitude de produits supportés par Android. Cependant, la répartition des versions du système d'exploitation est encore très fragmentée et cette nouvelle édition ne sera pas supportée par tous les appareils. Les développeurs devront penser pendant quelque temps encore à rendre leurs applications compatibles avec les appareils plus anciens. Pour cela, Google fournit un package de compatibilité afin d'utiliser les nouveautés telles que les fragments sur les anciennes versions de la plateforme.

■ Pierre-Emmanuel **Pollet**, *consultant Osaxis* (www.osaxis.fr).

Fig.5

Fig.6



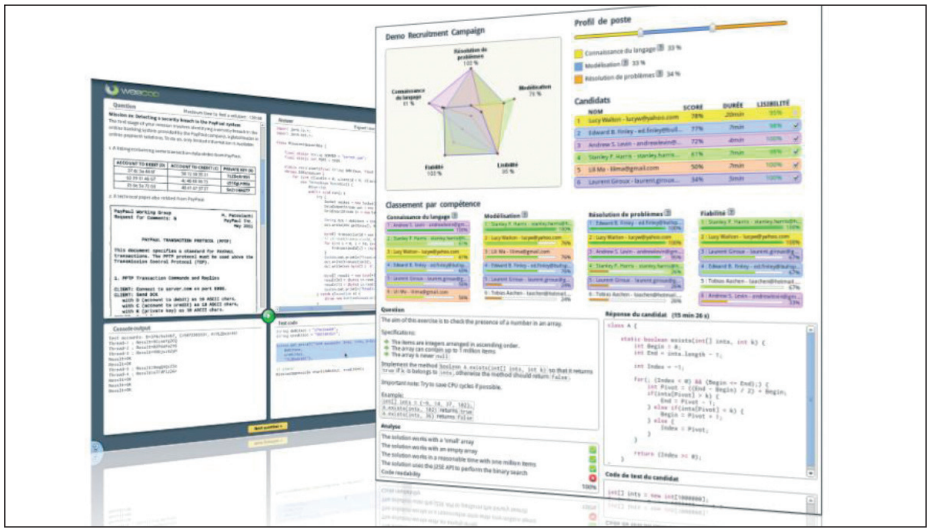
Evaluer en ligne les compétences de programmation des candidats

Weecod est un outil en ligne d'aide à la décision sur la partie technique du recrutement de développeurs en informatique.

Un processus de recrutement de développeurs informatique se déroule classiquement en trois étapes : un pré-filtrage technique est réalisé sur les CV ; les ressources humaines (RH) assurent des entretiens afin d'établir une 'short list' ; les entretiens techniques sont assurés par les services compétents de l'entreprise. Ce processus affiche cependant plusieurs failles : le contenu du CV est théorique et peut se révéler éloigné des compétences réelles des individus ; les RH n'ont pas les connaissances nécessaires pour juger ces mêmes compétences ; les techniciens de l'entreprise n'ont pas le temps d'assurer les entretiens en face à face afin de valider les capacités des candidats qui peuvent être évaluées via des QCM ne disposant généralement que d'exercices théoriques ! Dans ces conditions, un décalage très net subsiste entre les connaissances théoriques, sur lesquelles portent finalement le recrutement standard, et les compétences réelles des candidats et leur capacité à s'adapter à une mission.

Autre exemple classique, le jeune ingénieur diplômé qui sort de l'école se verra bloqué aux portes des entreprises qui recrutent, car son CV n'affiche que ses connaissances théoriques, qui peuvent là encore être validées par un QCM standard, mais n'affiche pas ses compétences acquises à titre personnel et qui en feraient pourtant un candidat directement opérationnel. L'entreprise préférera se tourner vers des profils experts qui affichent quelques années d'expériences…

Les fondateurs de la jeune start-up Cartser ont subi ces contraintes. Ils ont « galéré » de longs mois, diplôme en poche, avant de trouver une entreprise qui leur ouvre ses portes. De leurs expériences professionnelles, ils ont également constaté que souvent ce ne sont pas les personnes qui ont le plus de connaissances qui sont les plus appropriées à être positionnées sur certaines tâches en entreprise. *« Ce sont plutôt les personnes qui ont la tête sur les épaules, une capacité de recul et de décortiquer les problèmes qui sont généralement intéressantes, en particulier pour les tâches de R&D ou qui demandent réflexion »,* constate Frédéric Desmoulin, à l'origine de la création de Cartser. *« Dans une SSII, au quotidien et au delà de la connaissance des standards de type Java ou C++, le développeur informatique peut être amené à travailler sur des problématiques complexes, qui demandent à être rapidement opérationnel. De même chez un éditeur de logiciels, participer à la R&D soulève des problématiques qui demandent réflexion. »*



> *« Les candidats sont invités à montrer ce dont ils sont capables… »*

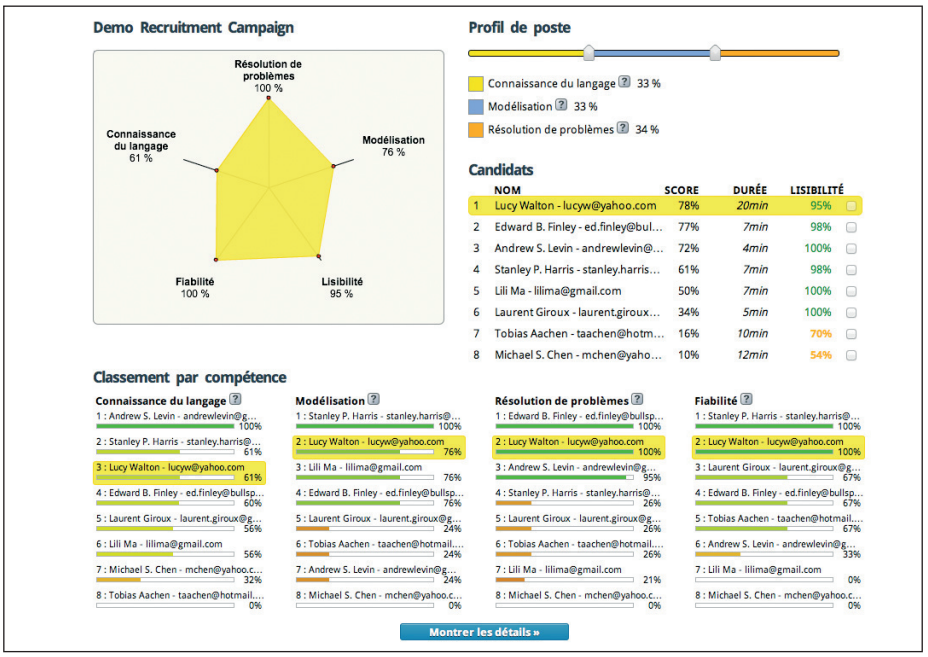
Se posant la question de ce qui est important pour le recrutement d'un développeur informatique, Cartser se propose d'inverser les processus via son offre en ligne Weecod, une plateforme d'évaluation des compétences techniques en programmation qui permet de simuler des problèmes concrets et demande aux candidats d'élaborer une solution en programmant dans un éditeur de code interactif en ligne. Ce dernier permet de coder, de compiler, de tester le code, et de soumettre la solution à un exercice. *« Weecod est orienté vers l'extraction chez le candidat de sa capacité à structurer son raisonnement afin d'essayer de décortiquer un problème et de l'amener à une solution,* décrit **Frédéric Desmoulin**.

Nous avons conservé le volet connaissances, qui est important pour les technologies, les langages, les frameworks, afin de mesurer si le candidat peut être très rapidement opérationnel sur ces thèmes. Mais nous allons plus loin sur sa capacité à structurer un raisonnement. Notre démarche d'inversion du processus a une finalité démocratique : elle n'est pas basée sur un CV souvent incorrect, mais sur l'évaluation des compétences réelles du candidat. »

L'entreprise qui recrute définit une fiche de poste et valorise ses priorités en termes de compétences techniques : CONNAISSANCES, le candidat sera-t-il rapidement opérationnel ? ; RESOLUTION, sera-t-il capable de traiter des algorithmes complexes ? ; MODELISATION, pour qualifier le profil architecture, etc. Le recruteur peut jouer sur les curseurs qui délimitent la valeur de ces compétences selon le profil recherché, par exemple en donnant plus de

Des concours de programmation

Dans ses cartons, Cartser a le projet de proposer dès 2012 des concours de programmation à destination des étudiants. Ce programme ambitieux est destiné à permettre aux entreprises et aux start-up partenaires de se sourcer auprès des jeunes qui ont du talent. Les universités et écoles de formation à l'informatique, d'une part, et les entreprises ou organisations intéressées par ce projet sont invitées à prendre contact avec la jeune entreprise.



valeur aux 'connaissances' pour un programmeur, ou plus de 'modélisation' à un responsable de projet. Une fois les candidats testés sur Weecod, l'entreprise dispose d'un rapport qui les classe selon leurs compétences, ce qui permet d'extraire les bons profils. Les RH n'ont donc pas besoin de disposer d'une expertise technique pour évaluer et valider les compétences, sans entretiens techniques et sans faire appel aux ressources internes de l'entreprise.

Côté langages, Weecod couvre actuellement Java et PHP. C#, C et C++ sont en préparation. Suivront Python et Ruby.

Weecod s'adresse à toutes les entreprises et organisations qui, ne souhaitant pas ou ne disposant pas des ressources pour consacrer leurs moyens techniques au recrutement, cherchent un outil de sélection des candidats qui réduise le traitement individuel. Une démarche d'industrialisation qui séduit les cabinets de recrutement qui n'ont pas d'expertise pour valider les compétences des candidats, ainsi que les SSII qui entendent adopter une approche humaine et qualitative du recrutement. Enfin, comme toute jeune start-up, Cartser se cherche encore sur son modèle économique. La solution est proposée en mode SaaS, avec des formules dont le prix varie en fonction des volumes. Elle devrait évoluer vers des systèmes de forfaits et d'abonnements mensuels. ■ Yves **Grandmontagne**

Soutenir les TPE et PME de l'informatique

Ils sont 50 000 freelances et TPE du numérique en France, porteurs d'innovation et d'agilité, à la recherche de visibilité, de business, voire de légitimité. La CICF Informatique et le MUNCI s'engagent mutuellement pour faire entendre leur voix.

Le partenariat engagé entre la CICF (Chambre de l'Ingénierie et du Conseil en France) Informatique et l'association professionnelle des informaticiens MUNCI, est un rappel sur le positionnement en équilibre instable des freelances, TPE et PME du secteur du numérique en France. Ces entreprises sont au nombre de 50 000, elles représentent plus de 90% des entreprises du secteur, et occupent 63% des emplois informatiques. *« Les TPE et PME du numérique sont mal considérées, voire pas du tout, et subissent le déréférencement,* nous confie **Marie Prat,** présidente du CICF Informatique. *Nous avons la volonté de les fédérer plus largement, afin d'être plus nombreux et de faire entendre nos voix. »*

Marie Prat, présidente du CICF Informatique

Les deux organisations établissent un bilan peu encourageant pour les acteurs du secteur. Farouchement indépendants au point de n'être qu'une minorité à se faire représenter – à peine plus de 500 adhérents et affiliés à la CICF sur 50 000 entreprises, et 2 300 au Munci sur 550 000 informaticiens ! - ils doivent faire face à de nombreuses difficultés. A commencer par la pratique de plus en plus manifeste de la sous-traitance en cascade.

Un mal qui gangrène les ressources des TPE/PME

Comme le fait remarquer Régis Granarolo, président du MUNCI, *« La spécialisation du numérique passe par des petites structures ».* Freelance, consultants indépendants, TPE et PME sont aujourd'hui une source de compétences, d'innovation et d'agilité dans laquelle piochent les grands acteurs du secteur à la recherche d'experts au profit de leurs clients. Le constat est alarmant : trop souvent, c'est l'expertise vendue en bout de chaîne par un spécialiste officiant au sein d'une petite structure qui permet à l'entreprise de grande taille d'honorer le marché négocié au prix fort avec son client. Résultat, selon Marie Prat, *« La problématique du tarif journalier de la prestation intellectuelle, qui en France*

> *« Notre but est que les petites structures soient correctement rémunérées »*
>
> **Marie Prat, présidente du CICF Informatique.**

a chuté à 300, 400 € par jour, est due aux différents niveaux de sous-traitance ». C'est le phénomène de la sous-traitance en cascade.

La solution ? Les deux organisations ont tout d'abord engagé une saisine collective du médiateur national des relations inter-entreprises et de la sous-traitance. Le MUNCI milite depuis 2003 pour un 'Small Business Act' à la française – aux Etats-Unis, 23 à 40% des marchés publics sont réservés aux PME nationales – tandis que la CICF mise sur la sensibilisation des acheteurs.

En communiquant plus efficacement avec eux, en obtenant la possibilité de se faire référencer et de répondre directement aux appels d'offres, et d'être rémunérés à un taux correct. *« C'est le moment d'établir des ponts entre nous et les acheteurs,* milite Marie Prat. *Notre but est de faire en sorte que les petites structures aient du business et soient correctement rémunérées ».*

> *« L'informatique est un secteur où l'individualisme est roi… »*
>
> **Régis Granarolo, président de l'association MUNCI.**

Et de travailler sur les savoirs avec les acheteurs, afin de répondre à leurs demandes et à leurs doutes face aux freelances et TPE.

D'autres actions porteront sur l'accompagnement juridique, ainsi que sur la labélisation avec le développement de partenariats avec les éditeurs, des transferts de compétences, et le développement de certifications techniques. ■ Y.G.

Polémique sur le chômage des informaticiens

Régis Granarolo, président du MUNCI

Quels sont les chiffres réels des emplois informatiques en France ? **Régis Granarolo**, président du MUNCI, pointe les chiffres généralement cités par les autorités nationales et la presse, dont la fourchette est données par le Syntec Numérique : le taux de chômage dans le secteur informatique se situerait à un niveau plancher compris entre 2% et 4%. Des chiffres que le MUNCI qualifie de « fantaisistes »… Pourquoi ? Il serait calculé sur la base des demandeurs d'emploi dont la dernière entreprise appartient à la Branche 'Logiciel, Conseil & Services informatiques'. Y figureraient également les employés non informaticiens – administratifs, commerciaux, managers, etc. - qui représenteraient 20 à 30% des salariés de la Branche. Le Syntec ajouterait les indépendants et les informaticiens de la fonction publique, qui par leur statut sont exclus des statistiques du chômage. Enfin, seraient absents du calcul les demandeurs d'emploi informatique dont le code ROME correspond à la qualification ou au poste recherché, ainsi que les jeunes diplômés.

Conclusion selon le MUNCI, les effectifs salariés du secteur informatique seraient sur-évalués, et donc en contrepartie le taux de chômage sous évalué. Le taux de chômage des informaticiens, calculé sur la base des statistiques publiques Dares/Insee, serait en réalité de 5,2% ! Pourquoi un tel décalage ? Pour Régis Granarolo, le Syntec Informatique - qui représente 80% du chiffre d'affaires du secteur, mais seulement 6% des entreprises – entretiendrait ainsi le mythe des « pénuries d'informaticiens » pour justifier plus particulièrement les pratiques de « flexibilité » et de l'offshore de ses adhérents.

C'est là que les préoccupations du MUNCI rejoignent celles de la CICF Informatique. Entretenir des chiffres de l'emploi informatique erronés profiterait à certaines pratiques, dont des salaires plus bas en France que dans certains pays européens ou en Amérique du Nord et qui progressent peu, ainsi que les tarifs des prestations intellectuelles maintenus bas. Le spectre de la sous-traitance en cascade n'est pas loin, mais doit également affronter celui de l'offshoring… L'influence du Syntec Numérique sur les grandes instances nationales n'est plus à démontrer, et souvent justifiée, ce que ne contestent ni la CICF, son homologue pour les PME, ni le MUNCI, mais ces organisations souhaiteraient obtenir plus de considération pour les TPE et PME du numérique… ■ Y.G.

TENDANCES EMPLOIS INFORMATIQUE

Sur un marché difficile, cela n'aura échappé à personne, la demande en provenance des prestataires informatiques continue d'augmenter, de 2% en octobre et d'une année sur l'autre selon le baromètre Hitechpro. Elle demeure particulièrement forte sur les domaines techniques des nouvelles technologies (Java, Web, GED…) et des systèmes, réseaux et sécurité, qui représentent chacun près de 30% de la demande. On notera également un retour sur le domaine des clients serveurs, en augmentation de 10%. En revanche, Hitechpro a détecté un net repli (-43%) sur les ERP, ainsi que dans une moindre mesure sur l'industrie, l'électronique et la R&D (-10%) et sur la recette et le support utilisateurs (-8%).

Côté secteurs économiques, la finance (36,8%) et l'assurance (12,4%) continuent de dominer largement et pour moitié la demande, suivis loin derrière par les télécoms et l'électronique (7,7%), les services aux entreprises dont le travail temporaire (7,6%), l'administration et les services publics (5,9%), les médias, la communication et les loisirs (5,3%) et le commerce et la distribution (5,1%).

Ces entreprises qui recrutent

Après l'euphorie de la rentrée de septembre - qui a vu exploser les recrutements, en particulier chez les SSII, qui représentent un peu moins des deux tiers des emplois informatiques en France, et sur quasiment tous les types de postes (ingénieurs, informaticiens, techniciens, infogérants, consultants, etc.) - prolongée en octobre avec les salons de recrutements, le marché retombe, malmené par l'actualité économique. Il faudra probablement attendre la fin du premier trimestre 2012 et le visu des entreprises du secteur sur leurs résultats comme sur les tendances du marché et de leurs clients, voire des élections…, pour que la demande du recrutement reparte à la hausse.

Programmez a repéré pour vous ce mois-ci une centaine de postes à pourvoir :

Ingénieurs

vNext, start-up française (créée en 2010) éditrice de logiciels et société consulting IT qui se positionne sur le marché des nouvelles technologies, usages, industrialisations et mode de distribution de type SaaS, recrute une dizaine d'ingénieurs pour ses bureaux basés à Paris. www.vnext.fr

Informaticiens

CSC France, conseil, intégration et externalisation, 2 300 collaborateurs et un chiffre d'affaires de 384 millions d'euros, recrute pour ses trois expertises une vingtaine d'ingénieurs, de consultants, de chefs de projets et de développeurs (Java). www.csc.com/fr

Ingénieurs

Osiatis, SSII en fort développement dans les services aux infrastructures et les développements nouvelles technologies, continue de recruter des ingénieurs systèmes, des administrateurs, des architectes, des consultants, des ingénieurs avant-vente et commerciaux, des techniciens help desk, etc.

www.osiatisrecrute.com

Concevoir des applications aujourd'hui : quels changements ? comment s'adapter ?

Les technologies de développement sont en mutation permanente, boostées par les nouveaux usages : appareils nomades et autres médias sociaux omniprésents dans notre quotidien. Si l'on compare une application développée il y a 10 ans avec une application actuelle, c'est le jour et la nuit, tant du point de vue de son ergonomie que de ses fonctionnalités. Cette évolution déborde du champ technique du projet et de son architecture, touchant également les processus, la méthodologie et même les profils des intervenants.

Lorsque l'on démarre un projet aujourd'hui, il est important de prendre tous ces aspects en considération voire de les revisiter. Par ailleurs, de nouvelles technologies et outils peuvent aussi nous aider à supporter ces nouvelles contraintes.
Evidemment, pas de science exacte autour de ces sujets là, simplement des retours d'expérience illustrés, pour l'exercice, par des technologies Microsoft.

Une application d'aujourd'hui : quels changements ?

Mais qu'entend-on par "le développement d'aujourd'hui" ? C'est simplement l'intégration de nos usages et contraintes actuels.
D'un point de vue fonctionnel :

- des applications disponibles sur le poste de travail, mais aussi sur nos appareils nomades (téléphone, tablette,…)
  des applications qui savent mieux exploiter
- leur environnement de fonctionnement (interaction avec d'autres applications connexes ou communautaires)
- de plus en plus de données en ligne, pour les partager de manière publique ou privée une meilleure réactivité (exploitation de nos machines multi-core)
- une ergonomie "sexy" et pensée pour l'utilisateur des services disponibles depuis n'importe où et tout le temps

D'un point de vue technique :

- l'avènement des applications web
- l'arrivée de plateformes et de services dans les nuages
- les applications naissent/vivent/évoluent/coexistent/meurent mais les données perdurent
- le besoin d'une architecture générique et adaptable (adaptation au développement agile)
- une meilleure lisibilité du code
- une meilleure testabilité
- une amélioration du confort de développement et de la productivité du développeur

Bref, tout un programme !
Parallèlement - ou plutôt transversalement - à ces aspects fonctionnels et techniques se décidera l'architecture de l'application.
Lorsque l'on parle d'architecture logicielle, on mélange souvent 2 niveaux différents de conception :

- L'architecture « physique » au sens client/serveur
- L'architecture « logique » de l'application cliente

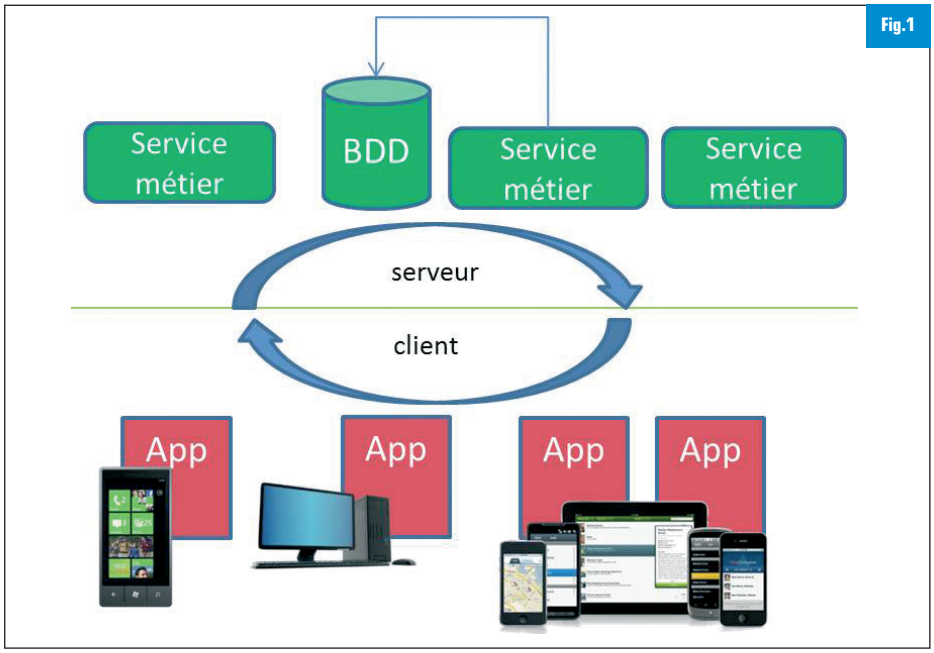


Fig.1

L'architecture « physique » au sens client/serveur

L'architecture « physique » va décomposer l'application de manière fonctionnelle, mais également technique pour répartir les éléments côté client et serveur. L'application se trouvera côté client, et les services métiers ainsi que la base de données et sa publication se trouveront côté serveur. Tout le fonctionnel qui est déporté côté serveur sera factorisé pour les différents clients potentiels [Fig.1].

Combien de scénarios métiers sont couverts par mon projet ?

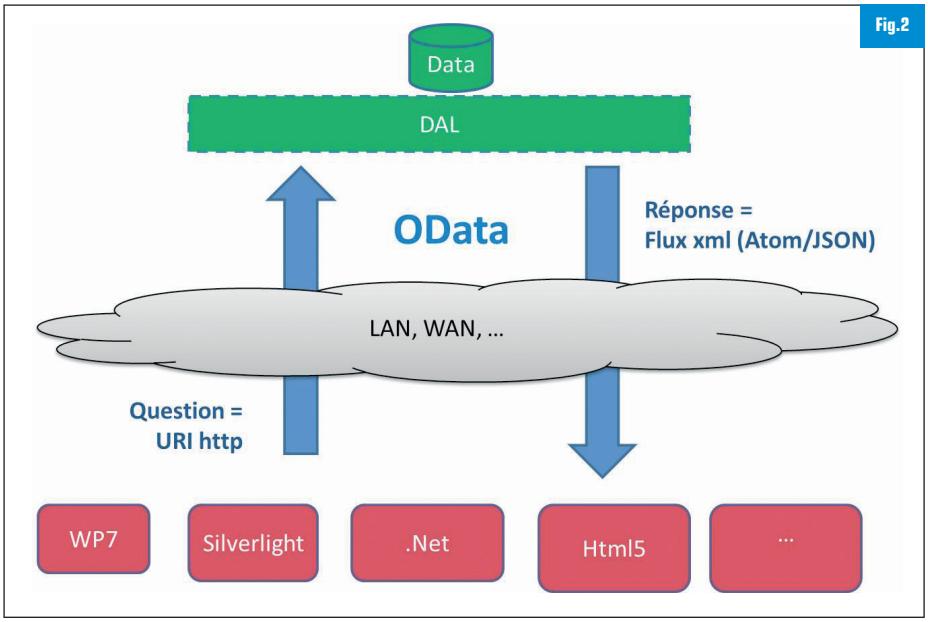
Un projet donne souvent lieu à plusieurs applications. D'autant plus qu'aujourd'hui il est très fréquent de développer une version d'application par smartphone et/ou tablette. Quand ce n'est pas l'application complète qui se décline sur plusieurs plateformes, cela peut être un service métier qui est utilisé par différents clients. Par exemple c'est souvent le cas pour l'accès aux données : il est très fréquent que plusieurs clients (applications ou services) utilisent les mêmes données, ne serait-ce que pour des scéna-

rios internes de maintenance ou de debug. La factorisation des services métiers côté serveur est une manière de minimiser le coût de développement des applications clientes. Encore faut-il qu'ils puissent être techniquement accessibles par ces plateformes.

Quelles sont les plateformes cibles de mes applications ?

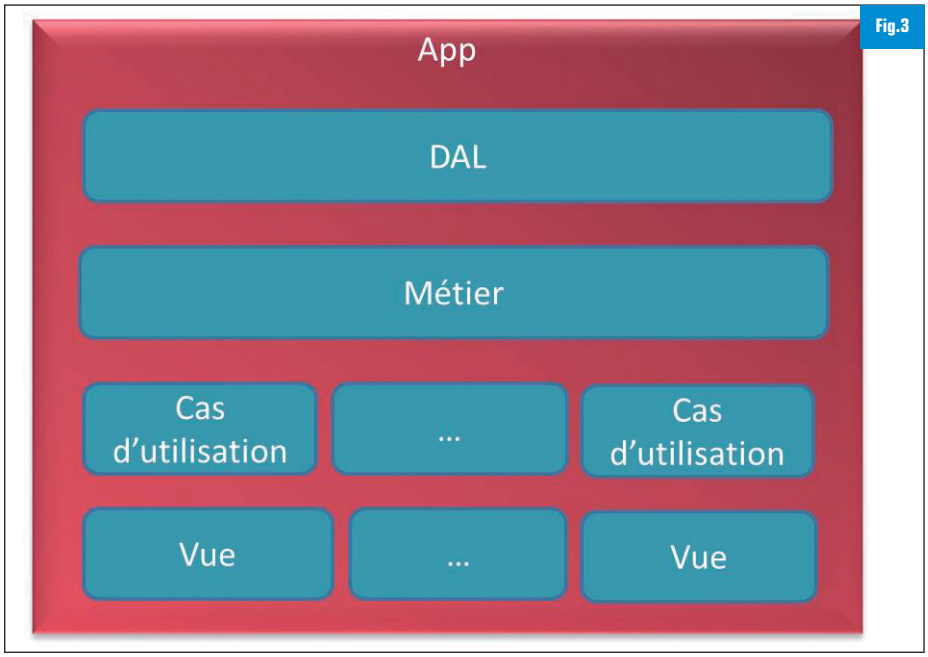
Avec l'évolution des usages, il est difficile de prévoir à l'avance toutes les plateformes cibles potentielles pour nos applications. D'où l'intérêt de choisir une technique de publication très interopérable.

Par exemple dans le cas du service de publication des données, le protocole OData est un choix intéressant. En effet, il permet de publier des données filtrables à la source et modifiables, à travers une simple Uri http. N'importe quelle plateforme cliente qui sait communiquer en http et manipuler un flux xml - en gros « tout le monde » - sera en mesure de manipuler ces données [Fig.2].



L'architecture « logique » de l'application cliente

On entend par là le découpage en couches d'une application : vue, cas d'utilisation, métier, accès aux données (DAL)…ou appelez-les comme vous voudrez ? [Fig.3].



Architecture 4-tiers

Le découpage en couches va permettre une bonne évolutivité de l'application et une bonne réutilisabilité des composants côté client. Mais a-t-on toujours intérêt à mettre en place une architecture fortement découplée ?

On pourrait débattre des heures sur l'intérêt ou non de mettre en place de bout en bout une architecture 4-tiers ultra découplée avec utilisation massive d'interfaces et de l'injection de dépendances. Dans les faits, il n'y a évidemment pas une seule bonne manière de procéder et il faut mettre la rapidité de développement en balance avec la souplesse. L'architecture dépendra des besoins fonctionnels, mais aussi - et surtout - des besoins d'évolutivité et donc de la durée de vie potentielle de l'application.

Ça peut paraitre tout bête mais c'est peut-être la première question à se poser.

Quelle est la durée de vie de mon application ?

L'évolutivité de l'application est très souvent corrélée à sa durée de vie : de nouveaux besoins émergeront forcément au fil du temps, voire même un changement de fonctionnement de l'existant.

Plus une application doit durer, plus son architecture devrait être souple et évolutive. Ainsi une application à but évènementiel pourrait être conçue différemment d'une application métier pour l'industrie.

Le nombre d'intervenants sur le projet est lui aussi fortement lié à la durée de vie de l'application : entre les sous-traitants qui se succèdent, les priorités qui changent et le turn over naturel des collaborateurs, il est fréquent que les contributeurs initiaux aient complètement disparu du projet après quelques mois ou quelques années.

Il est donc primordial – surtout si le projet ou l'application doit perdurer dans le temps – de définir une architecture claire et bien documentée pour que le projet puisse être appréhendé le plus rapidement possible. Attention, un projet facile à prendre en main ne signifie pas qu'il faille écrire beaucoup de documents ni qu'il faille noyer le code sous les commentaires !

Plus une architecture est claire, moins le code nécessite d'explications. D'ailleurs plus un code comporte de commentaires, moins ceux-ci seront à jour et moins le développeur sera enclin à les lire. Rien de neuf sous le soleil me direz-vous, mais les méthodologies actuelles sont plus ouvertes à cette vision des choses que ce n'était le cas il y a quelques années.

De la même manière, les documents produits lors de la conception d'un projet sont rarement tous utilisés par ses protagonistes car ils sont eux aussi assez rarement à jour. S'ils le sont, cela signifie qu'une énergie importante a été mise en œuvre pour le permettre. Il peut être judicieux de se poser la question du nombre de fois où ils ont été consultés. Ceux qui servent le moins pourraient être abandonnés pour passer plus de temps sur d'autres composantes du projet comme les tests qui sont souvent laissés pour compte en fin de chaîne.

D'autres alternatives intéressantes existent pour définir et faire respecter une architecture, comme le propose par-exemple Visual Studio avec son diagramme de couches. Chaque module est associé à une couche et pour chaque couche, on définit les relations autorisées vers ses homologues [Fig.4].

Diagramme de couches

A chaque compilation (ou manuellement), les appels et donc les dépendances entre les différents modules sont évalués et contrôlés. Pour imposer et faire perdurer une architecture N-tiers tout au long d'un projet, c'est parfait ! Le développeur constate lui-même ses erreurs en temps réel lors du codage, ce qui est plus pédagogique et moins frustrant que de s'entendre devoir modifier son code suite à une relecture du chef de projet. Avec Visual Studio, il est également possible de déterminer a posteriori les dépendances entre les modules pour comprendre le découpage du code, ou de générer des diagrammes de séquence [Fig.5].

Fig.4

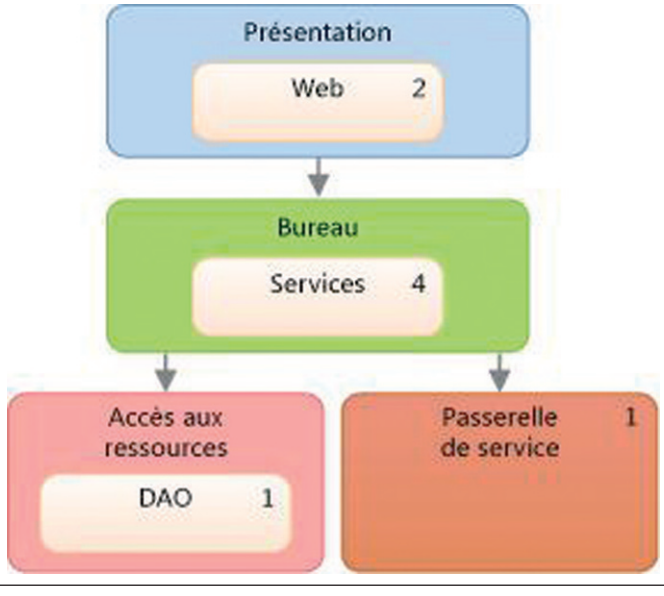
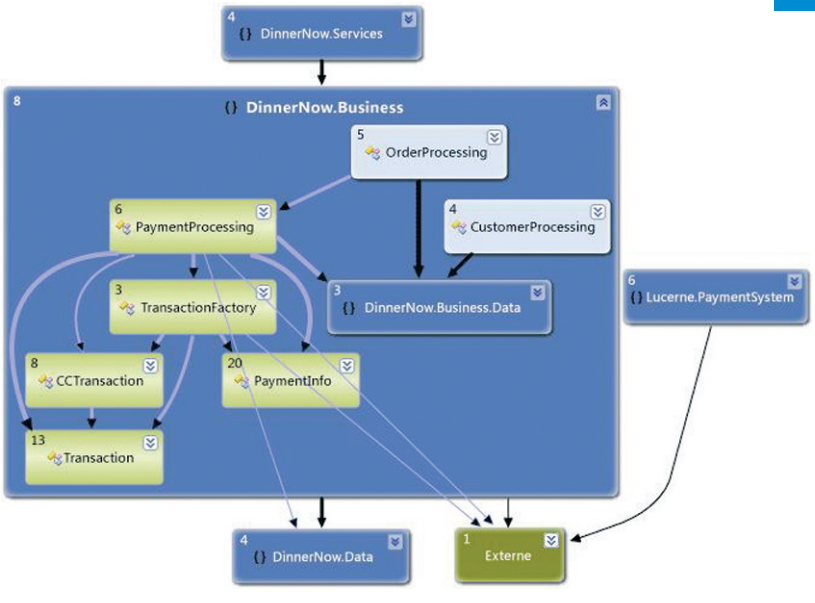


Fig.5



Graphique de dépendance [Fig.6].

Diagramme de séquence

L'avantage est que ces diagrammes sont produits en temps réel et sont donc à jour, ce qui est rarement le cas avec des documents ou schémas produits manuellement.

Vers plus d'agilité

Il peut être délicat d'abandonner la production de certains documents, tout simplement parce que ceux-ci font partie intégrante du processus de conception ou oserais-je dire « processus qualité » (quel est donc ce frisson d'effroi qui vient de vous parcourir l'échine ?). En tout cas, c'est pourquoi (pas à cause du frisson, mais du processus) de plus en plus d'équipes de développement se tournent vers des méthodologies dites agiles : moins de production de documents, une intégration du client dans le processus de conception avec des revues régulières pour corriger le tir au fur et à mesure de l'avancement du projet, bref moins de rigidité avec au final une meilleure réactivité et moins de surprises à la livraison. En effet, plus le client est intégré au processus de conception et ses revues successives, plus il est conscient et solidaire du résultat, d'où moins de réserves lors de la recette.

De l'intérêt du designer

L'expérience utilisateur est un élément décisif dans le succès d'une application. L'intervention de designers professionnels dans les équipes de développement n'est pourtant pas encore une pratique très répandue. Un bon développeur n'est (généralement) pas un bon designer. En tout cas, à la base, il n'a pas été formé pour cela. Ce n'est pas parce qu'on sait faire des logos en flamme et des dégradés de toutes les couleurs qu'on sait concevoir une interface pratique, agréable à regarder et à utiliser. Le design, l'ergonomie et la définition de chartes graphiques, c'est un métier ! Et ça ne s'apprend généralement pas en codant.

On préfère souvent recruter un technicien de plus plutôt que de doter l'équipe d'un designer, mais c'est faire un mauvais calcul. En effet, les développeurs passent énormément de temps à travailler sur leurs pages et l'interface utilisateur de manière générale, pour un résultat assez médiocre il faut bien l'avouer. La productivité d'un designer pour effectuer ce travail - pour lequel il a été formé, et qui est donc son métier ! - est évidemment bien meilleure.

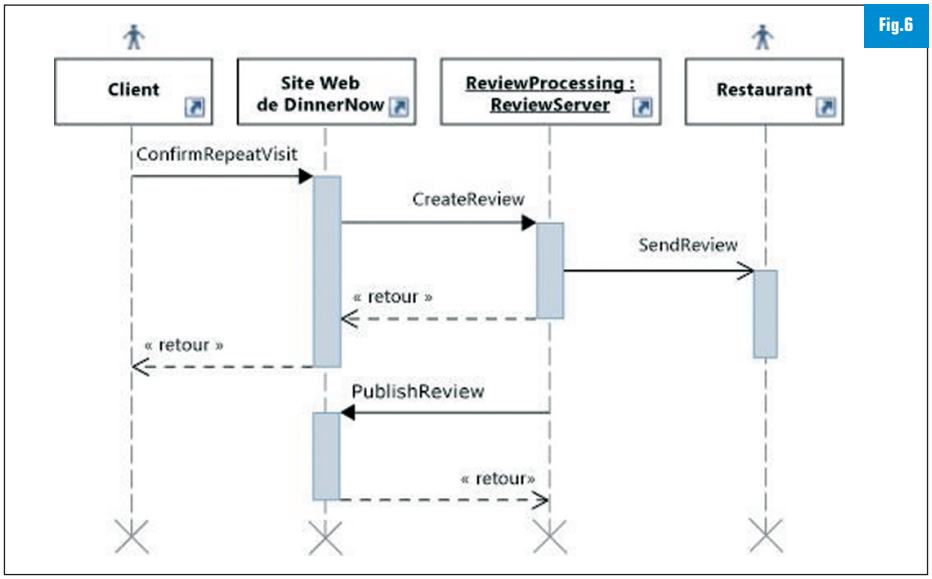


Fig.6

Cette charge dont les développeurs sont libérés sera réaffectée au développement des fonctionnalités métiers et techniques où leur productivité est maximale.

C'est du gagnant-gagnant : une interface plus réussie et pensée pour l'utilisateur ainsi qu'un gain de productivité général pour le projet : pourquoi s'en priver ? Surtout que les techniques de développement et outils actuels permettent et encouragent de plus en plus cette séparation fonctionnel/design (Visual Studio/Expression Blend chez Microsoft). Ainsi de nouveaux métiers émergent depuis quelques années : les intégrateurs. Ces profils se situent à mi-chemin entre les développeurs et les designers et permettent une communication aisée avec le reste de l'équipe technique (puisqu'à l'aise avec le jargon des développeurs) ainsi qu'une bonne connaissance des outils de développement et de design.

Hébergement des applications, services et données

La question de l'hébergement est indirectement liée à la notion d'architecture du projet, dans la mesure où les services doivent être capables de s'adapter à la charge des connexions et requêtes clientes.

Nous avons besoin de serveurs mettant à disposition les données et services, en faisant en sorte qu'ils soient disponibles tout le temps. Tout ce petit monde se doit de se porter comme un charme, peu importe le nombre d'utilisateurs. Au lancement, il y aura certainement peu de clients, mais au fil du temps et du succès de l'application, notre système doit pouvoir supporter la montée en charge. Le développeur n'a aucune idée de la manière dont il doit dimensionner ses serveurs. En fait, l'idéal serait de ne pas avoir de serveurs à installer, maintenir, configurer. Après tout, l'hébergement n'est pas directement lié au projet au sens métier du terme, et l'on préfèrerait se concentrer sur les applications.

C'est le principe d'une plateforme cloud (Azure chez Microsoft). Qui dit plateforme, dit que toute la partie infrastructure est déjà prête à l'emploi, mettant à disposition un environnement, un framework de développement et des services.

Par rapport à un développement qui est hébergé sur un serveur au sens classique du terme, on utilisera des éléments inhérents à la plateforme Azure, en termes de stockage et de publication. Vous me direz, la belle affaire, mais pourquoi y aurait-il des différences entre mon développement classique et un développement Azure ?

Eh bien justement, pour pouvoir garantir la possibilité au système de monter en charge, de supporter les pannes matérielles, sans impacter sa disponibilité. Avec un développement classique hébergé sur un serveur, vos données sont stockées localement. Il faut prévoir au minimum des mécanismes de redondance pour supporter une panne matérielle. Si vous avez besoin de plus de ressources et décidez d'ajouter un serveur, comment allez-vous répartir la charge sur ces serveurs, partager les données, sans interrompre l'accès à vos services ?

Les nuages sans les perturbations

Avec Azure, le principe est de ne pas faire de stockage local sur une machine (on parle en fait d'instance), mais sur des éléments persistants fournis par la plateforme Cloud, qui seront accessibles par les différentes instances quel qu'en soit le nombre. Si une instance redémarre, le système fonctionne toujours et c'est transparent pour l'utilisateur. Finalement, on s'abstrait de la notion de machine et on se concentre sur les applications et les services ainsi que les systèmes de stockage. Les questions d'hébergement, de redondance, d'installation et de maintenance de serveurs n'existent plus en tant que telles. On a intégré ce besoin d'élasticité et de disponibilité directement au niveau de la conception des applications. Notre application et nos services sont prêts une fois pour toute à fonctionner aussi bien au ralenti qu'à pleine charge, en fonction du nombre et de la taille des instances qui leur auront été attribuées. La tarification se fait en fonction des critères de volume, bande passante, nombre et taille des instances.

Autre avantage : l'environnement Azure est intégré dans Visual Studio, ce qui permet au développeur de travailler dans un seul et même outil, que ce soit pour le développement, le déploiement ou la maintenance de ses applications. Aucun dépaysement pour le développeur en ce qui concerne le stockage de données puisque la base de données « cloud » de Microsoft : SQL Azure, s'utilise exactement comme son cousin SQL Server. On peut utiliser les outils classiques de SQL Server pour administrer une base de données "dans les nuages", simplement en précisant la chaîne de connexion SQL Azure correspondante.

En conclusion

Etre conscient des nouveaux usages, cibler davantage de plateformes en essayant de rester interopérable, penser aux technologies Cloud qui supportent nativement la montée en charge, faire de l'expérience utilisateur une priorité, évaluer de nouveaux outils qui peuvent faire gagner en productivité, vérifier la pertinence de la méthodologie et du processus de conception, …

Pas vraiment de révolution dans tout cela me direz-vous. Mais l'air de rien, la prise en compte de ces éléments peut réellement aider l'équipe à travailler avec plus d'efficacité et à sortir une application de meilleure qualité, bref contribuer concrètement au succès d'un projet.

■ Stéphanie **Hertrich**
Relation Technique Développeur
Microsoft france

Installer GIT en un clin d'œil en mode SaaS, sur le cloud

Git est un SCM (Source Control Manager) décentralisé. Il ne repose pas sur une base de données unique (dépôt). Celle-ci peut être disponible sur plusieurs serveurs, et chaque développeur en possède une copie complète sur son propre poste. GIT en mode SaaS est l'outil idéal pour un développeur Agile, nomade. Il permet au développeur de travailler n'importe où dans le monde.

Dans un premier temps nous allons installer GIT sur la solution AWS (Amazone Web Service). A cet effet nous allons utiliser Amazon Simple Storage Service (S3) comme dépôt central Git. Amazon S3 est un service de stockage pour Internet. Il est pensé pour simplifier l'accès à un système de stockage à l'échelle du web, pour les développeurs. Pour l'installation de GIT (JGit), il faut dans un premier temps installer Git depuis http://git-scm.com/ Puis télécharger le script Shell JGit depuis le lien http://www.eclipse.org/jgit/download/ et ajouter au répertoire $HOME/bin en le renommant jgit. Puis il faut ajouter un fichier de configuration dans votre $HOME/.jgit pour y ajouter les clés d'accès qui permettent une forte authentification au compte Amazon Web Service.

```
accesskey: aws access key
secretkey: aws secret access key
```

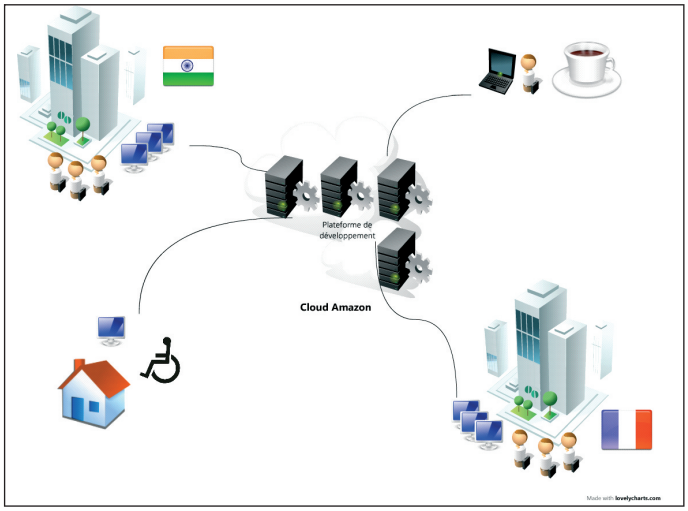
Ces clés sont disponibles dans l'administration du compte AWS sur lequel vous voulez créer le dépôt. Elles vous font donc créer un bucket dans la section S3. Pour trouver ces clés, une fois connecté à votre compte allez dans Compte, puis dans Identifiants de sécurité, dans le paragraphe « Accédez aux justificatifs d'identité » puis récupérez l'identifiant de clé d'accès et la clé d'accès secrète et ajoutez-les au fichier de configuration créé ci-dessus. Ces clés ne doivent en aucun cas être communiquées. Pour plus d'information sur l'authentification forte sur AWS, rendez-vous sur le site d'Amazon AWS. Dans un deuxième temps il faut créer son dépôt en local en utilisant les commandes de Git et ajouter le serveur à la liste des serveurs connus grâce à la commande :

```
git remote add origin amazon-s3://.jgit@nom-du-bucket/le-dépôt.git
```

Il est ensuite possible d'utiliser la surcouche JGit à l'aide à la commande jgit qui possède quasiment les mêmes options que la commande git. Pour donner accès au dépôt central Git sur le Cloud aux autres développeurs, Amazon propose une interface graphique pour gérer les droits d'accès selon le « Bucket S3 ». Il est possible de choisir d'affecter à un autre compte AWS les droits de consultation, d'ajout et de suppression, de visualisation des droits et de modification des droits. Les coûts de GIT en SaaS sur AWS S3 défient toutes les solutions concurrentes du marché !

Pour évaluer, le coût de GIT en SaaS sur S3, il faut connaître l'espace de stockage nécessaire sur S3, le type de requête sur S3 (PUT, COPY, POST ou LIST) et le transfert de données sortantes.

Pour connaître la masse de données sortantes nous avons dû faire une estimation : selon le nombre d'utilisateurs. Pour cela il faut comprendre le fonctionnement de Git. Git utilise la commande clone pour récupérer l'ensemble d'un dépôt, le téléchargement depuis le serveur sera donc de la taille du dépôt (ici 100 Mo). Mais une fois le clone fait, la commande pull permet de récupérer uniquement les derniers ajouts ou modifications faits par d'autres développeurs depuis un autre clone. On peut se rendre compte que l'action de cloner peut engendrer un téléchargement assez important depuis le serveur, tandis que chaque pull représentera au maximum quelques dizaines de Ko. On notera que le nombre de clones est à peu près équivalent au nombre de développeurs.



Estimation de l'utilisation d'Amazon S3
(attention, les tarifs évoluent régulièrement)

Espace utilisé	500 Mo	1 Go	2 Go	1 Go	2 Go	5 Go	13 Go
Prix Stockage ($)	0,07	0,14	0,28	0,14	0,28	0,7	1,82
Trafic sortant	5 Go	10 Go	20 Go	10 Go	20 Go	50 Go	130 Go
Prix du trafic sortant ($)	0,6	1,2	2,4	1,2	2,4	6	15,6
Totaux ($/mois)	0,67	1,34	2,68	1,34	2,68	6,7	17,42

Prix de stockage = 0,14$ * espace utilisé (Go). Prix du trafic sortant = 0,12$ * trafic sortant (Go).

Ce tableau est une estimation du premier mois d'utilisation, c'est-à-dire en comptant une dizaine de clones par dépôts. Il faut bien comprendre qu'une fois ces clones effectués pour mettre à jour le dépôt chaque développeur fera un pull ou deux pulls par jour mais la quantité transférée pour un pull sera presque nulle.

Nous constatons que l'utilisation de GIT en SaaS sur AWS a un ROI très avantageux, remplaçant aisément une solution interne ou externe clé en main. Cette solution SaaS offre une infrastructure de stockage hautement durable et totalement sécurisée, offrant une disponibilité de service de 99,99 %.

■ Cyril **Agoin** & Hervé **Desaunois**
Consultants seniors
■ Robin **Collet**, Damien **Flament**, Thierry **Weissbeck**, Lois **Collet**
Ingénieurs stagiaires
Valtech Consulting Toulouse

A la découverte du SDK de l'AR. Drone

2e partie

D.R.

Après avoir fait une présentation de l'AR. Drone, et vu ce qu'il est capable de faire, dans le numéro précédent, nous allons maintenant passer à un exercice un peu plus technique. Nous avions évoqué l'existence d'un SDK (Software Developpement Kit), mais sans nous y attarder plus que cela. Il est temps de combler ce manque et de vous faire découvrir comment tirer parti de l'AR. Drone au sein de votre propre application. Le but étant de créer un client « home-made » permettant de faire décoller et contrôler un AR. Drone.

Nous allons dans un premier temps étudier l'API fournie directement par Parrot (l'entreprise ayant créé et distribuant l'AR. Drone) qui est dans un langage dit « bas-niveau » car reposant seulement sur du C. Ce choix est tout à fait logique pour une entreprise comme Parrot qui commercialise énormément d'appareils de type embarqué, ou fonctionnant avec ce type d'appareil. Être bas niveau permet d'avoir accès à l'ensemble des ressources de la machine host, que ce soit un appareil mobile ou bien un mini-hélicoptère. Alors oui, cela peut sembler un peu verbeux pour quelqu'un qui ne pratique que des langages haut niveau tels que C# ou bien Java, mais chacun a ses avantages. Nous parlerons aujourd'hui seulement de l'API fournie par Parrot en C, mais notez qu'il existe des surcouches dans à peu près tous les langages existants, et même haut niveau, pour ceux qui sont allergiques au bas niveau. Le SDK de Parrot est multiplateforme et fonctionne tout aussi bien sur Linux, Mac ou bien Windows, mais nous allons opter pour la plateforme Windows pour les exemples de cet article, choix arbitraire, mais classique.

Rappel

Nous avions vu que l'AR. Drone c'est une coque, quatre hélices, mais surtout un « cerveau » reposant sur un noyau Linux. Ce dernier lui permet de gérer de nombreuses informations, mais également d'effectuer des calculs qui peuvent être directement faits par l'appareil sans avoir besoin d'être déportés sur chacun des clients, on pourra citer par exemple la détection de Tags.

Le SDK est là pour communiquer avec le Linux embarqué dans votre AR. Drone, l'ensemble des communications passe bien sûr par la connexion Wifi qui relie l'appareil à votre PC (ou téléphone), mais le protocole en lui-même est une API complexe que Parrot a mise en place avec son expérience de ce type d'appareil.

Nous allons dans cet article nous intéresser principalement à l'API en elle-même et peu au grand principe réseau existant, permettant à l'ensemble de communiquer correctement. Pour nous, développeurs, nous noterons juste qu'il ne faut pas oublier de lier les deux appareils en mode ad hoc avant de faire nos tests, sinon peu d'espoir que cela décolle !

Installation

Comme tout SDK, il faut noter une phase d'installation et surtout de prérequis. Par chance, Parrot à plutôt bien fait les choses, avec une documentation d'une taille que nous pourrions qualifier de légère, mais suffisante dans la majorité des cas. La communauté autour de ce SDK commence à prendre de l'ampleur, donc même si vous vous retrouvez dans une situation de blocage, vous devriez pouvoir vous en sortir assez rapidement.

Voici donc la liste des prérequis que vous devez installer pour commencer le développement sous Windows :

1 Visual Studio 2008/2010 express ou plus
2 https://projects.ardrone.org/ pour récupérer le SDK AR. Drone
3 http://www.microsoft.com/downloads/en/details.aspx?FamilyID=6b6c21d2-2006-4afa-9702-529fa782d63b&displaylang=en pour le SDK Windows
4 http://www.microsoft.com/downloads/en/details.aspx?displaylang=en&FamilyID=3021d52b-514e-41d3-ad02-438a3ba730ba pour le SDK DirectX
5 http://www.libsdl.org/release/SDL-devel-1.2.14-VC8.zip pour la SDL (bibliothèque graphique)

Les deux SDK de Microsoft sont à installer, tandis que pour les deux autres prérequis, je vous invite à les mettre dans un dossier à la racine de votre disque, cela sera plus simple pour la configuration de votre projet.

Dans l'archive contenant le SDK de l'AR. Drone, vous trouverez une documentation complète décrivant l'installation dans le dossier Examples\Win32\VCProjects, ce n'est pas forcément la partie la plus passionnante de la découverte du SDK mais elle n'en reste pas moins vitale, donc suivez-là à la lettre pour avoir un environnement de travail stable. Une fois l'installation faite, vous devriez être en mesure d'exécuter la démonstration Win32 permettant de prendre

le contrôle de votre AR. Drone. Dans le cas contraire, vérifiez point par point le processus d'installation jusqu'à obtenir une solution

Explication

Nous allons maintenant passer à l'explication du code et surtout comment refaire votre propre client permettant de contrôler un AR. Drone. La première étape est bien sûr l'initialisation de l'API AR. Drone. Celle-ci passant par le réseau réclame deux choses, que le PC soit connecté en Wifi avec l'AR. Drone mais également que votre binaire ouvre une communication réseau avec ce dernier. Nous allons pour ce faire utiliser les API Windows. Si on regarde de plus près notre Main on devra y trouver :

```
int main(int argc, char **argv)
{
    WSADATA wsaData;
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
    {
        printf("Erreur d'initialisation du systeme de socket\n");
        getchar();
        exit(-1);
    }
```

Ce *main* est pour le moment très simple et sans rapport direct avec l'AR. Drone, mais nécessaire pour indiquer à Windows que l'on va travailler avec du réseau. Une fois cette initialisation faite, on peut passer à l'étape suivante qui est la détection d'un AR. Drone au bout de la connexion Wifi. Pour cela, nous allons utiliser la même fonction qui est fournie dans le sample de Parrot, celle-ci active une connexion sur le FTP interne à l'AR. Drone (eh oui !) pour vérifier qu'il répond bien, une sorte de ping amélioré, permettant en même temps de récupérer certaines informations sur l'appareil telles que le numéro de version du firmware par exemple.

```
int main(int argc, char **argv)
{
    WSADATA wsaData;
    if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
    {
        printf("Erreur d'initialisation du systeme de socket\n");
        getchar();
        exit(-1);
    }
    if (test_drone_connection() != 0)
    {
        printf("Impossible de detecter l'Ardrone\n");
        getchar();
        WSACleanup();
        exit(-1);
    }
```

Si ces deux fonctions passent avec succès alors nous pouvons affirmer que notre logiciel est maintenant connecté à l'AR. Drone, ce n'est que le début, mais il faut bien commencer par les bases. On notera que la fonction de test est bloquante dans la version du sample, il faudra donc penser à modifier ce comportement par la suite pour rendre l'utilisation de votre contrôleur plus agréable, avec l'ajout d'une interface graphique par exemple. Mais restons pour le moment en console et regardons de plus près l'étape suivante qui est celle de l'initialisation.

Le test de connexion à vérifié que l'appareil était bien présent sur le réseau et en a profité pour récupérer son numéro de version, mais par le biais du protocole FTP. Pour la suite nous devons mettre en place une communication fixe entre les deux appareils, nous faisons donc appel à la fonction qui s'occupe d'établir ce lien pour que nous puissions ensuite y faire transiter nos trames réseaux, comprendre par-là nos commandes à effectuer.

```
if (ardrone_tool_setup_com(NULL) != 0)
{
    printf("Erreur d'initialisation de la liaison wifi\n");
    getchar();
    exit(-1);
}
```

Une fois la connexion établie de manière fixe entre les deux appareils, il faut encore configurer l'API, celle-ci comprend de nombreux éléments tels que la gestion des entrées permettant de contrôler votre appareil, mais également le flux vidéo ou même simplement la gestion de ses « informations vitales » comme le niveau de batterie, son altitude ou bien son orientation par exemple.

La démonstration fournie contient déjà deux éléments qui nous seront vraiment utiles dans un premier temps, une gestion du clavier en se reposant sur DirectX pour surveiller quelles touches sont pressées, mais également la création d'une fenêtre permettant de voir le flux vidéo de la camera frontale de l'appareil.

Nous verrons ensuite comment modifier voire ajouter un périphérique de contrôle, libre à vous ensuite d'imaginer quels types d'entrées vous souhaitez utiliser. Une souris ? Une Kinect ? Un casque capable de lire les flux nerveux pour jouer avec la « force » ?

Nous allons donc pour le moment simplement faire appel à la fonction *ardrone_tool_init(int, char**)* qui s'occupe d'initialiser un ensemble d'éléments et d'en avertir l'AR. Drone par le biais d'une première commande contenant sa configuration. Nous irons voir ensuite comment ajouter notre propre initialisation dedans.

```
if (ardrone_tool_init(argc, argv) != 0)
{
    printf("Erreur d'initialisation du drone\n");
    getchar();
    exit(-1);
}

while ( ardrone_tool_exit() == FALSE )
{
    res = ardrone_tool_update();
}
```

Une fois cela fait, l'AR. Drone est connecté et configuré, autant dire prêt à décoller. Nous avons besoin de mettre en place une boucle dite « infini » permettant de garder le contact avec notre AR. Drone une fois celui-ci activé. Cette boucle a plusieurs rôles, le premier étant de maintenir la connexion ouverte par le biais d'un appel à la commande de l'API et de faire une sorte de ping régulier vers l'AR. Drone indiquant que nous sommes encore actif.

Si ce dernier ne le reçoit pas, il va automatiquement activer le mode emergency permettant de se poser rapidement et dans la mesure du possible en douceur.

Le deuxième rôle est d'appeler nos contrôleurs, nous avons évoqué leur existence plus haut, mais sans nous attarder plus que cela. Il est temps maintenant d'étudier la question de plus près. Il existe déjà deux contrôleurs dans le code fourni au départ, un pour le clavier et un pour une manette, nous pouvons voir qu'ils sont tous les deux enregistrés auprès de l'API dans la méthode d'initialisation, plus exactement dans une sous-fonction se nommant : *ardrone_tool_init_custom(int, char**)* qui comme son nom l'indique permet d'ajouter facilement des éléments à l'API sans interférer avec le reste du code.

Si on regarde de plus près le code de cette fonction, on peut y voir ces deux lignes :

```
/* Registering for a new device of game controller */
    ardrone_tool_input_add( &dx_keyboard );
    ardrone_tool_input_add( &dx_gamepad );
```

On y retrouve bien le clavier et la manette et si on regarde d'encore plus près le type de ces deux paramètres, on comprend comment ajouter nos propres « input manager ». On déclare une structure de type *input_device_t* qui comprend un nom et trois pointeurs sur fonction. Une pour l'initialisation du périphérique, une pour l'update et une pour la fermeture. À partir de là, il est assez facile d'ajouter son propre gestionnaire de périphérique qui va venir s'ajouter comme un plugin dans la phase d'initialisation de l'API.

```
input_device_t dx_keyboard = {
  "DirectX Keyboard",
  open_dx_keyboard,
  update_dx_keyboard,
  close_dx_keyboard
};
```

```
typedef struct _input_device_t {
  char name[MAX_NAME_LENGTH];

  C_RESULT (*init)(void);
  C_RESULT (*update)(void);
  C_RESULT (*shutdown)(void);
} input_device_t;
```

Contrôleur « Home-Made »

Voici une implémentation minimaliste d'un contrôle personnalisé que nous pouvons ajouter dans le code de l'API. La méthode Start est donc appelée une seule fois au début, Close à la fin du programme et Update quant à elle sera appelée à intervalle régulier durant la boucle infinie de l'API permettant de donner des instructions à l'AR. Drone en suivant l'API, nous allons voir cela ensuite.

```
input_device_t input_Keyboard = {
  "Input HomeMade",
  Start,
  Update,
  Close
};


C_RESULT Start(void)
{
    return C_OK;
}

C_RESULT Update(void)
{
    return C_OK;
}

C_RESULT Close(void)
{
  return C_OK;
}
```

Imaginons ensemble une implémentation « bateau » que nous pourrions faire de ce nouveau contrôleur pour notre AR. Drone. Les fonctions Start et Close vont rester vides pour le moment, car nous n'avons pas vraiment de besoin, mais elles seront utiles pour la suite pour, par exemple, ouvrir un socket réseau et le fermer dans le cas où nous voudrions exposer notre contrôleur sur le réseau par exemple, mais encore une fois, libre à vous d'implémenter la gestion qui vous intéresse.

Pour le moment nous allons donc nous contenter de remplir un peu la fonction update qui pour mémoire est appelée en boucle par l'API. C'est tout à fait comparable à une boucle de jeu pour les connaisseurs, ou bien un callback appelé en boucle pour les amateurs de programmation évènementielle. Après avoir étudié un peu plus la documentation de l'API fournie par Parrot, on constate qu'à notre niveau, donner vie à l'AR. Drone n'a rien de bien compliqué, il faut utiliser principalement deux fonctions de l'API, ni plus ni moins ! Ces deux fonctions font ensuite appel à la couche réseau de l'API qui se charge d'envoyer la bonne commande à l'appareil pour qu'il soit en mesure de la comprendre, mais surtout de l'exécuter.

- *ardrone_tool_set_ui_pad_start(int)* qui prend en paramètre un entier ayant pour valeur 1 pour décoller et 0 pour atterrir
- *ardrone_at_set_progress_cmd(int, float, float, float, float)* qui prend en paramètre le nécessaire pour faire voler l'ArDrone
  - int : 0 pour le mode « hover » permettant de stabiliser le vol, 1 pour passer en mode « free »
  - float : une valeur en -1 et 1 permettant de contrôler le roll
  - float : une valeur en -1 et 1 permettant de contrôler le pitch
  - float : une valeur en -1 et 1 permettant de contrôler les gaz
  - float : une valeur en -1 et 1 permettant de contrôler le yaw

Nous pourrons tout de même citer la fonction *ardrone_tool_set_ui_pad_select(int)* permettant d'activer le mode Emergency en cas de problème avec votre pilotage, une valeur de 1 pour l'activer et 0 pour le désactiver.

Une fois cela compris, nous pouvons imaginer une version de notre contrôleur proche de celle-ci :

```
C_RESULT Start(void)
{
    ardrone_tool_set_ui_pad_start(1);
    return C_OK;
}

C_RESULT Update(void)
{
    // récupération des informations de gaz sur une pédale
    float gaz = GetGaz();
    ardrone_at_set_progress_cmd(0, 0, 0, gaz, 0);
    return C_OK;
}

C_RESULT Close(void)
{
    ardrone_tool_set_ui_pad_start(0);
  return C_OK;
}
```

Cela reste basique comme implémentation, mais permet déjà de contrôler l'altitude de notre AR. Drone, libre à vous ensuite d'ajouter le code nécessaire pour lui donner les bonnes impulsions pour avancer ou bien tourner en fonction de vos envies.

Et ce n'est qu'un début

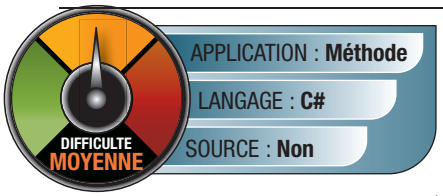
Vous êtes maintenant capable de rapidement mettre en place votre propre client pour piloter votre AR. Drone et cela directement en C, nous n'avons pas passé en revue la manière de manipuler le flux vidéo, le but étant surtout de comprendre comment contrôler l'appareil, mais il faut noter que l'on récupère directement un flux vidéo, donc il est aisé ensuite d'y appliquer tout traitement possible. La démonstration fournie avec le SDK vous permettra de mieux comprendre comment lire ce flux et ensuite à vous de vous en servir.

Toute l'API repose sur un principe d'extensibilité, il est facile d'ajouter nos propres comportements et contrôleurs à l'API. ce qui permet de se l'approprier, mais surtout de la rendre utilisable par tout type d'appareil, que ce soit sur un téléphone (il vous suffit d'une connexion réseau) ou bien sur un PC, ce qui ouvre un énorme champ de possibilités en matière d'appareil permettant de contrôler un AR. Drone.

■ Niels **Freier** – *Consultant chez Wygwam* (www.wygwam.com)
www.stumpy.fr

Design By Contract : Code Contracts avec C# 4.0

Dans l'article S.O.L.I.D paru il y a quelques mois, nous vous avons présenté les grands principes qui ont pour objectif d'améliorer la qualité des programmes. Cet article approfondit le principe de Ségrégation de Liskov (LSP) et présente comment Microsoft, par le biais de l'API Code Contracts, permet de l'implémenter de manière efficace et élégante.



Bertrand Meyer, en 1985, est le premier à introduire la technique de programmation par contrat (Design By Contract) permettant de répondre techniquement au Principe de Liskov. Cette technique propose de vérifier que certaines conditions formant un contrat sont vraies à un moment donné d'un programme. Pour cela, trois types de conditions sont définies :

- Pré-condition : condition devant être vraie en amont du traitement
- Post-condition : condition devant être vraie en aval du traitement
- Invariant : condition devant toujours être vraie

1 CODE CONTRACTS

En tant que développeur, il est recommandé de valider tout entrant externe à une méthode. Nombreux sont les développeurs qui passent du temps à implémenter leur propre logique de validation qui vérifie une certaine condition et lève une exception si elle n'est pas respectée. Cette approche peut devenir rapidement illisible et lourde à maintenir. Voici un exemple de validation sans Code Contracts :

```
public double ComputeInterest(double amount, double rate)
{
  double result = 0;

  //Pre-Condition
  if (rate < 1)
    throw new ArgumentOutOfRangeException(
      «Interest rate must be less than 1.»);

  result = amount + amount * rate;

  //Post-Condition
  if (result < amount)
    throw new ArgumentOutOfRangeException(
      «Amount after interest must be superior to initial amount.»);

  return result;
}
```

Même si le code peut être optimisé en encapsulant la logique de validation dans une seule classe, il reste l'exemple de ce qu'il ne faut plus faire car chaque méthode a besoin de sa propre logique de validation. Afin de répondre à cette problématique, Microsoft propose une implémentation du principe « Design by Contract » à travers une API très complète et simple à utiliser : Code Contracts. Tout ce qui est nécessaire à l'utilisation de cette API est déjà présent dans le .NET Framework 4.0 (dans la BCL). Il est aussi possible d'utiliser Code Contracts avec le .NET Framework 3.5, ***en téléchargeant les binaires*** sur le site de Microsoft http://research.microsoft.com/en-us/projects/contracts/. Les différents types de contrats disponibles ainsi que la manière de les utiliser vont être présentés par le biais d'un exemple simple. Un point à remarquer : Pour pouvoir utiliser Code Contracts, l'activation se trouve dans les paramètres de projet dans l'onglet Code Contracts [Fig.1].

2 API CODE CONTRACTS

Voici comment implémenter les différents types de contrats disponibles dans l'API Code Contracts.

Les Pré-conditions

Ce sont les conditions qui doivent être vraies en début de traitement. En général, il faut s'assurer que certains paramètres ne sont pas nuls, ou encore qu'ils ont des valeurs valides. De cette manière, on est capable de savoir que le module est appelé dans des conditions normales de fonctionnement. Voici un exemple de pré-conditions :

```
//Pre-Condition
Contract.Requires(rate > 0, «Rate must be higher than 0.»);
Contract.Requires(amount > 0, «Amount must be higher than 0.»);
Contract.Requires(rate < 1, «Rate must be less than 1.»);
```

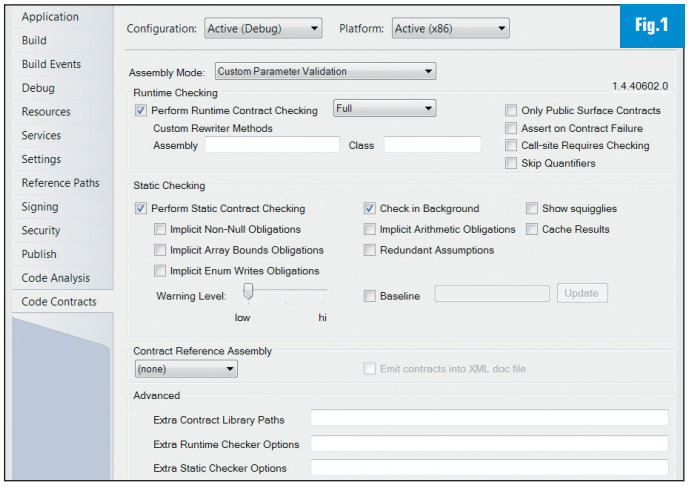


Fig.1

La méthode Requires(…) de la classe statique Contract est utilisée. Les conditions sont positionnées en début de méthode et permettent de spécifier le message d'erreur associé à chaque non-respect de conditions. A l'exécution de la méthode avec des paramètres d'entrée qui ne respectent pas les pré-conditions, une exception est levée au runtime [Fig.2].

Les Post-conditions

Les post-conditions sont les conditions qui doivent être vraies à la fin d'un traitement. Les post-conditions sont utilisées pour valider que le résultat renvoyé est fonctionnellement valide : on va par exemple vérifier que le total d'une facture est supérieur à zéro, que la liste des articles commandés contient bien plusieurs articles…
Pour cela, la classe statique Contract est à nouveau utilisée, cette fois-ci, c'est la méthode Ensure(…) qui est appelée. Comme pour les pré-conditions, les post-conditions sont généralement positionnées en début de méthode. Voici un exemple de post-condition :

```
//Post-Condition
Contract.Ensures(Contract.Result<double>() > amount,
  «Amount after interest must be superior to initial amount.»);
```

A l'exécution de la méthode avec un résultat qui ne respecte pas les post-conditions, une exception est levée au runtime [Fig.3].

Les Invariants

Les invariants sont des conditions qui doivent toujours être respectées. Au lieu d'insérer la validation de ces conditions de multiples fois dans le code, Code Contracts permet de les centraliser. Pour cela, il faut regrouper toutes les conditions invariantes dans une même méthode identifiée avec l'attribut [ContractInvariantMethod]. L'API injecte du code qui permet l'exécution de la validation à chaque appel d'une méthode ou quand une propriété de la classe est modifiée. Attention, car cela peut avoir des impacts sur la performance. Prenons l'exemple suivant : un contrat d'invariance est défini sur une classe, la méthode ContractInvariants(…) contient des conditions sur la propriété ContractID et sur le champ privé _minimum.

```
private double _minimum = 1;
public string ContractID { get; private set; }

public CEL(string contractID)

{
 ContractID = contractID;
}

[ContractInvariantMethod]
private void ContractInvariants()
{
 Contract.Invariant(_minimum >= 1);
 Contract.Invariant(ContractID.Length == 4);
}
public virtual double ComputeInterest(double amount, double rate)
{
 //Post-Condition
 Contract.Ensures(Contract.Result<double>() > amount,
   «Amount after interest must be superior to initial amount.»);

 return amount + amount * rate + _minimum;
}
```

Regardons de plus près ce qui a été injecté lors de la compilation avec un outil de décompilation tel que .NET Reflector.

```
public virtual double ComputeInterest(double amount, double rate)
{
 double Contract.Old(amount);
 try
 {
   Contract.Old(amount) = amount;
 }
 catch (Exception exception1)
 {
   if (exception1 == null)
   {
     throw;
   }
 }

 double CS$1$0000 = (amount + (amount * rate)) + this._minimum;
 double Contract.Result = CS$1$0000;
 __ContractsRuntime.Ensures(Contract.Result > Contract.Old(amount),
   «Amount after interest must be superior to initial amount.»,
   «Contract.Result<double>() > amount»);
 this.$InvariantMethod$();
 return Contract.Result;
}
```

L'API injecte un appel à la méthode ContractInvariants(…) à la fin de la méthode ComputeInterest(…). Voici ce qui a été injecté dans le construc-

Fig.2

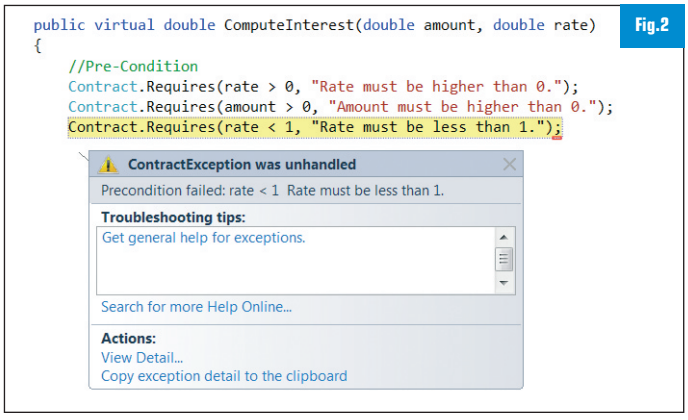
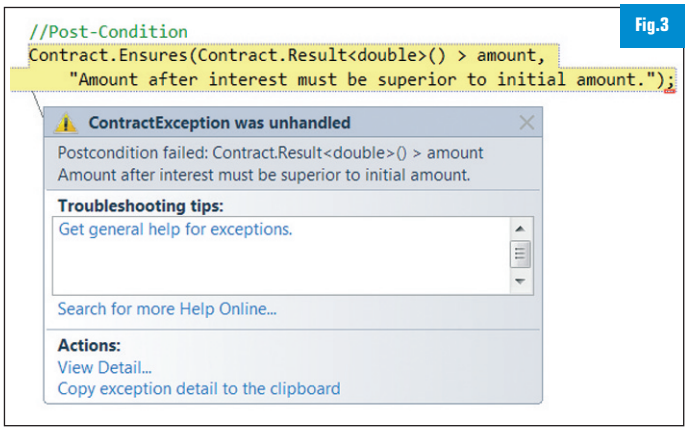


Fig.3



teur de la classe CEL et dans la propriété ContractID (get et set) :

```
public CEL(string contractID)
{
 bool flag = this.$evaluatingInvariant$;
 this.$evaluatingInvariant$ = true;
 this._minimum = 1.0;
 this.ContractID = contractID;
 this.$evaluatingInvariant$ = flag;
 this.$InvariantMethod$();
}

public string ContractID
{
 [CompilerGenerated]
 get
 {
   string Contract.Result = this.<ContractID>k__BackingField;
   if (!this.$evaluatingInvariant$)
   {
     __ContractsRuntime.Ensures(Contract.Result.Length == 4,
       null, «ContractID.Length == 4»);
   }
   return Contract.Result;
 }
 [CompilerGenerated]
 set
 {
   if (!this.$evaluatingInvariant$)
   {
     __ContractsRuntime.Requires(value.Length == 4,
       null, «ContractID.Length == 4»);
   }
   this.<ContractID>k__BackingField = value;
 }
}
```

L'API injecte seulement la vérification des conditions nécessaires en fonction du contexte.

Le fonctionnement des contrats d'invariance se résume donc ainsi :

- Exécution en fin de méthode, donc pas de vérification en début ou au milieu de la méthode.
- Prise en compte des propriétés en injectant les conditions liées aussi bien dans le set que dans le get.

Comme déjà évoqué, les performances de la classe peuvent être dégradées, il est donc nécessaire de bien évaluer ce qui doit être traité comme invariant afin de trouver un juste milieu entre validation fonctionnelle et performance.

3 CODE CONTRACTS ET LE MODÈLE OBJET

Nous avons parcouru les principaux types de contrats. Maintenant, il nous faut voir comment les intégrer à un modèle objet et les combiner avec les principes de la programmation orientée objet tels que l'héritage, le polymorphisme et l'utilisation des interfaces ?

Héritage & polymorphisme de méthode

Lorsqu'on dérive une classe qui possède des Contrats, la classe dérivée hérite automatiquement des contrats définis dans la classe de base. L'héritage des Contrats dans les sous-types est justement ce qui permet de respecter le principe de Liskov : on maîtrise les différentes modifications faites au type de base en s'assurant que le fonctionnement nominal est modifiable dans la limite des contrats définis. Il est également possible de modifier ou de compléter le contrat en ajoutant des conditions en début de méthode. Dans l'exemple suivant, la classe CEL contient la méthode ComputeInterest(…) avec des contrats.

La classe LivretA hérite des contrats et rajoute un nouveau Contrat avec une nouvelle pré-condition.

```
public class CEL
{
 private double _minimum = 1;
 public string ContractID { get; private set; }

 public CEL(string contractID)
 {
   ContractID = contractID;
 }

 [ContractInvariantMethod]
 private void ContractInvariants()
 {
   Contract.Invariant(_minimum >= 1);
   Contract.Invariant(ContractID.Length == 4);
 }

 public virtual double ComputeInterest(double amount, double rate)
 {
   //Pre-Condition
   Contract.Requires(rate < 1, «Rate must be less than 1.»);

   //Post-Condition
   Contract.Ensures(Contract.Result<double>() > amount,
     «Amount after interest must be superior to initial amount.»);

   return amount + amount * rate + _minimum;
 }

 public virtual double ComputeInterest(double amount, double rate,
 double minimum)
 {
   _minimum = minimum;
   return ComputeInterest(amount, rate);
 }
}

public class LivretA : CEL
{
 public LivretA() : base(«9999») { }

 public override double ComputeInterest(double amount, double rate)
 {
   //Additional Pre-Condition
   Contract.Requires(rate > 0, «The interest rate must be higher
than 0.»);
```

```
    return base.ComputeInterest(amount, rate);
  }
}
```

Contrat par interface

Une autre approche de conception consiste en l'association de contrats à une interface, l'idée étant que chaque classe qui implémente cette interface se voit automatiquement imposer ses contrats.

Cela est particulièrement important quand on utilise la programmation basée sur les interfaces (Interface Based Programming). En effet, des contraintes fonctionnelles sont ajoutées aux interfaces en plus des contraintes purement techniques : on évite ainsi les implémentations techniquement valides mais incohérentes.

L'exemple suivant présente une interface et la classe associée qui porte la définition des contrats.

La classe doit respecter les conditions suivantes :

- La classe doit implémenter l'interface
- La classe doit être abstraite (il n'y a aucun sens à vouloir créer une instance de ce type)
- La classe doit porter l'attribut [ContractClassFor(…)]
- Chaque méthode doit porter l'ensemble des contrats

```
[ContractClass(typeof(DefaultOperationClass))]
public interface IInterest
{
  double ComputeInterest(double amount, double rate);
}

[ContractClassFor(typeof(IInterest))]
internal abstract class DefaultOperationClass : IInterest
{
  public double ComputeInterest(double amount, double rate)
  {
    //Pre-Condition
    Contract.Requires(rate < 1, «Rate must be less than 1.»);

    //Post-Condition
    Contract.Ensures(Contract.Result<double>() > amount,
      «Amount after interest must be superior to initial amount.»);

    //Dummy Value, this method only serves to add additional
  contracts
    return 0;
  }
}
```

4 POUR ALLER PLUS LOIN AVEC CODE CONTRACTS

L'utilisation seule de Code Contracts est puissante, comme vous avez déjà pu le remarquer ; Code Contracts peut également être utilisé avec d'autres outils pour devenir un petit bijou au quotidien.

Pex & Code Contracts

Pex est une API de Microsoft Research proposant de générer automatiquement des tests unitaires (http://research.microsoft.com/en-us/projects/pex/). L'utilisation de Code Contracts permet à Pex de proposer des jeux de tests encore plus pertinents et plus complets. Ceci est effectué par lecture des contrats existants pour aboutir à une couverture maximale du code. La combinaison de ces 2 outils est donc très efficace.

SandCastle & Code Contracts

SandCastle est un outil de génération automatique de documentation de code (http://www.codeplex.com/Sandcastle/). Sandcastle est gratuit et peut être étendu pour fonctionner avec Code Contracts. Les fichiers de documentation XML contiennent ainsi des informations supplémentaires sur les contrats. De plus, des feuilles de style peuvent être utilisées avec Sandcastle pour que les pages de documentation générées incluent des sections de contrat.

L'analyse statique du Code

L'analyse statique permet d'évaluer de manière automatique l'exactitude du code source sans l'exécuter. Il existe plusieurs approches pour la mettre en pratique. La plus simple est l'utilisation du compilateur. Static Checker est une autre approche qui est un exécutable s'intégrant dans le processus de build. Le Checker analyse les faits et montre les éventuelles erreurs ainsi que les violations des contrats. L'analyse statique est donc un bon moyen de réduire les temps de développement.

L'attribut [ContractAbbreviator]

Imaginons que votre code contienne des méthodes dans une classe qui a les mêmes paramètres et/ou les mêmes contraintes qui doivent être vérifiées. Il faudrait normalement copier les contrats dans chacune de ces méthodes. Contract abbreviators permet de définir une seule méthode qui contiendra la validation pour les autres méthodes via l'utilisation de l'attribut [ContractAbbreviator].

C'est un attribut assez pratique qui ne se trouve pas dans les binaires contenant l'API Code Contracts. Il faut référencer la classe contenant la définition directement dans les projets (elle se trouve dans le répertoire « %ProgramFiles%/Microsoft/Contracts/Languages/ContractAbbreviator.cs »). Cet attribut permet donc de regrouper les contrats dans une méthode à part et ainsi de les factoriser à un seul endroit.

CONCLUSION

L'API Code Contracts apparaît donc comme une amélioration majeure du framework .NET. Elle permet de gérer en interne les exceptions liées à la validation du code, de positionner les contrats au niveau d'abstraction souhaité sans nuire à la lisibilité du code. Et bien sûr, vous pouvez personnaliser vos contrats et vous assurez de leur respect avant même l'exécution, grâce à l'analyse statique ! L'utilisation de PEX et de SandCastle va permettre d'augmenter la qualité de vos tests et de votre documentation et cela de manière automatisée. Il ne vous reste plus qu'une chose à faire : l'adopter !

Retrouvez le code des exemples sur http://designbycontract.codeplex.com !

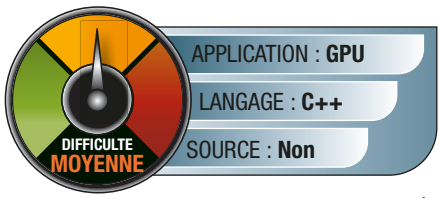
■ Jason **De Oliveira**
Practice Manager & Solutions Architect / MVP C#
.Net Rangers by Sogeti
Blog : http://jasondeoliveira.com

■ Fathi **Bellahcene**
Software Architect
.Net Rangers by Sogeti
Blog : http://blogs.codes-sources.com/fathi

Introduction à la programmation GPU

1re partie

La programmation GPU a souvent mauvaise réputation auprès des développeurs car elle est considérée comme difficile et réservée à des spécialistes techniques souvent répartis dans des domaines de niches réclamant des performances accrues : simulation en calcul scientifique, imagerie médicale, dynamiques des fluides, protection de l'environnement, calculs financiers en salle de marché ...



L'arrivé de la nouvelle librairie **Microsoft C++ Accelerated Massive Parallelism** (C++ AMP) est sans aucun doute un évènement important pour la communauté des développeurs Microsoft. Naturellement, les développeurs .NET sont un peu déçus, car pour l'instant, il n'y a pas d'implémentation GPU pour le Framework .NET planifiée, mais les équipes y réfléchissent. Cependant, si vous souhaitez développer en .NET sur GPU, la société **Tidepowerd** (http://www.tidepowerd.com/) fournit une belle implémentation .NET, avec sa librairie **GPU.NET** (http:/www.tidepowerd.com/product).

Pour une grande majorité des acteurs concernés par la programmation GPU, l'annonce de la nouvelle librairie **Microsoft C++ AMP** a suscité à la fois curiosité et étonnement. Pourquoi Microsoft annonce-t-il une nouvelle librairie pour traiter du calcul sur GPU alors que **nVIDIA CUDA** et **OpenCL** occupent déjà ce territoire ? La programmation GPU n'étant pas très populaire, je souhaitais vous présenter ces technologies à travers une série d'articles. L'objectif est de vous initier aux rudiments de ces technologies en utilisant systématiquement le même algorithme. Dans cette introduction je vous présenterai dans un premier temps les différentes technologies GPU que nous allons étudier : **nVIDIA CUDA C, OpenCL** et Microsoft C++ AMP. J'ai éliminé l'illustration avec l'API **Microsoft Direct Compute** (http://en.wikipedia.org/wiki/DirectCompute), car elle m'a semblé moins pertinente au regard de l'arrivée de la librairie Microsoft C++ AMP. Dans un second temps, je vous présenterai l'algorithme sélectionné pour toutes nos implémentations. Puis nous mesurerons sa performance en mode séquentiel. Enfin, nous terminerons cette introduction par l'utilisation de la librairie Microsoft **Parallel Patterns Library** (http://msdn.microsoft. com/en-us/library/dd492418.aspx), pour paralléliser notre code exemple sur CPU. Pour éviter tout malentendu, la programmation GPU est particulièrement bien adaptée au parallélisme orienté données et non orienté tâches. En d'autres termes, si votre problème est dominé par un volume de données conséquent qui doit être traité par une unique méthode, alors vous pouvez tirer parti d'une solution orientée GPU. Sur la plateforme .NET la librairie PLINQ répond à cette caractéristique, mais en général toutes les boucles parallèles sont du parallélisme orienté données.

PRÉSENTATION DES TECHNOLOGIES

nVIDIA CUDA C

Le framework nVIDIA CUDA (http://www.nvidia.com/object/cuda_home_new.html) connaît un succès grandissant depuis sa sortie en 2007. Il fait aujourd'hui office de standard dans de nombreuses entreprises. Par exemple, l'offre Microsoft HPC Server arrive avec une distribution nVIDIA CUDA afin de permettre aux développeurs d'implémenter des solutions hybrides, mélangeant par exemple les technologies MPI (http://en.wikipedia.org/wiki/Message_Passing_Interface), OpenMP (http://en.wikipedia.org/wiki/OpenMP) et nVIDIA CUDA C.

L'objectif est naturellement de tirer le maximum de puissance des matériels disponibles sur chacun des nœuds d'une grille HPC.

Aujourd'hui, il existe un véritable écosystème nVIDIA CUDA permettant de profiter de nombreuses librairies optimisées et décliné sur différents langages.

On y trouve aussi de l'outillage sophistiqué afin de profiler et de déboguer graphiquement sous Visual Studio, portant le développement GPU à une maturité proche du développement sur CPU.

Par nature, le code nVIDIA CUDA s'exécute exclusivement sur des matériels nVIDIA supportant l'architecture CUDA à la fois sous Windows, UNIX et MAC.

OpenCL

Plus récemment, le standard ouvert OpenCL (Open Computing Language : http://www.khronos.org/opencl) supporté par le groupe **khronos**, a commencé à rencontrer un franc succès, car son indépendance vis-à-vis du matériel et de la plateforme d'exécution présente un atout que **nVIDIA CUDA** n'offre pas.

En effet, **OpenCL** permet de produire un code source portable (**OpenCL** s'exprime en langage C), tout en acceptant de paralléliser des traitements sur CPU ou sur GPU ou bien les deux. Portable sur différents systèmes d'exploitation, différents types de cartes graphiques, **OpenCL** a de quoi séduire les plus exigeants.

Microsoft C++ AMP

La librairie Microsoft C++ AMP s'inscrit sur un autre terrain : elle est écrite nativement en C++ et supporte aussi la programmation parallèle sur GPU et CPU.
En interne, elle repose sur DirectX (http://en.wikipedia.org/wiki/DirectX) et de ce fait, elle supporte un grand nombre de matériels. Cependant, sa vocation est de s'exécuter sur des systèmes d'exploitation Microsoft Windows à l'instar des librairies parallèles Microsoft C++ **Parallel Patterns Library** (PPL) et **Asynchronous Agents Library** (AAL) de Visual Studio 2010. Contrairement aux technologies précédentes, **Microsoft C++ AMP**, n'est pas encore finalisé.

Cette nouvelle librairie viendra épouser l'écosystème de la prochaine version de Visual Studio. L'intention affichée par Microsoft est de démocratiser la programmation GPU en proposant un écosystème de développement simple et efficace.
L'accent n'est donc pas placé sur « la performance à tout prix », mais sur « la facilité d'utilisation ».

PRÉSENTATION DE L'ALGORITHME

S'il existe une version « Hello world » pour la programmation GPU, c'est sans aucun doute l'algorithme du produit matriciel (http://en.wikipedia.org/wiki/Matrix_multiplication). Cet exemple est très souvent repris en introduction des ouvrages ou des cours abordant la programmation GPU. Voici une déclinaison en langage C d'un produit matriciel engageant une matrice A et une matrice B carrées de taille *size*, engendrant la matrice C.

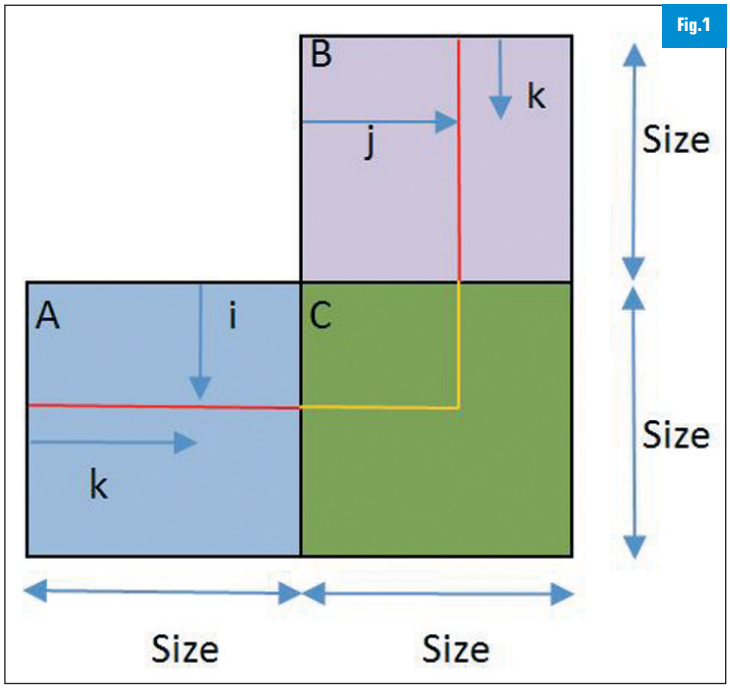


Fig.1

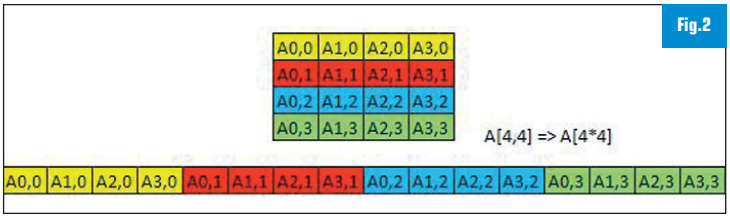


Fig.2

```
void MatrixMultiplyOrig(float** A, float** B, float** C, int size)
{
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            float sum = 0;
            for (int k = 0; k < size; ++k) {
                sum +=  A[i][k]*B[k][j];
            }
            C[i][j] = sum;
        }
    }
}
```

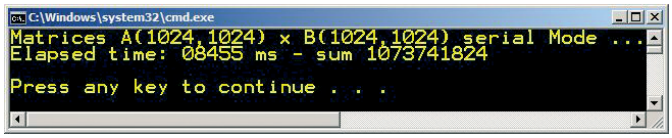
Implémentation du produit matriciel version séquentiel

Dans notre exemple nous disposons de deux matrices carrées *A* et *B* de taille *size* que nous allons parcourir à travers 3 boucles imbriquées où la plus interne va itérer la variable *k* sur les lignes de la matrice A et sur les colonnes de la matrice *B*, la variable *i* itère les colonnes de la matrice *A* et la variable *j* itère les lignes de la matrice *B*. Ainsi nous pouvons calculer le produit de chacune des valeurs de *A* et de *B* et engendrer toutes les valeurs dans *C* [Fig.1].
Bien que parfaitement correct, le code présenté ci-dessus n'est pas celui que l'on retrouve dans la littérature GPU. En effet, la carte graphique préfère manipuler des vecteurs plutôt que des structures comme des tableaux. Alors comment passer d'une matrice à deux dimensions à un simple vecteur ?
Par exemple, prenons le cas de la matrice A, où nous avons colorié chaque ligne en une couleur différente [Fig.2].
Nous pouvons aligner chaque ligne de la matrice A[4,4] de manière à obtenir un long vecteur de 16.
Appliquez ce principe à toutes matrices et modifiez légèrement l'algorithme initial, vous obtenez un code parfaitement compatible avec les technologies parallèles sur GPU.
Le code ci-dessous représente le code adapté pour calculer sur des vecteurs.

```
void MatrixMultiply(float* A, float* B, float* C, int size)
{
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            float sum = 0;
            for (int k = 0; k < size; ++k) {
                sum += A[i * size + k] * B[k * size + j];
            }
            C[i * size + j] = sum;
  }
    }
}
```

Implémentation du produit matriciel version séquentiel compatible CPU

Pour nous bâtir une référence vis-à-vis de nos prochaines adaptations parallèles, je vous propose le résultat du temps d'exécution de cet algorithme pour des matrices de tailles 1024 * 1024, initialisées avec la valeur 1 pour chaque cellule.
Le programme s'exécute en x64 en mode Release sur un Intel Core i7 2820 QM.



Le programme a tourné en 8 secondes et 455 centièmes, ce qui constitue notre valeur de référence.

MICROSOFT PPL ET LE PRODUIT MATRICIEL

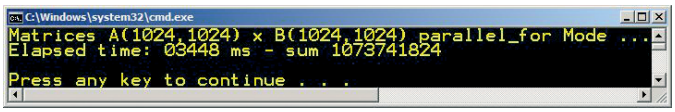
Afin de mieux apprécier les gains de performances de nos futures versions sur GPU, il est très tentant d'évaluer le gain obtenu en utilisant la librairie Microsoft **Parallel Patterns Library** (PPL).

La librairie **PPL** est finalement très proche de la librairie **Intel Threading Building Blocks** (http://threadingbuildingblocks.org), mais si vous utilisez Visual Studio 2010, vous disposez par défaut de cette librairie. Par exemple, si nous utilisons la méthode parallel_for (http://msdn.microsoft.com/en-us/library/dd505035.aspx), pour obtenir une version parallèle de notre algorithme, nous obtenons le code ci-dessous.

Si la syntaxe des lambdas expressions ne vous est pas familière, vous trouverez quelques explications de leurs usages sur ce lien : http://blogs.msdn.com/b/vcblog/archive/2008/10/28/lambdas-auto-and-static-assert-c-0x-features-in-vc10-part-1.aspx.

```
void MatrixMultiplyParallelFor(float* A, float* B, float* C, int size)
{
    for (int y = 0; y < size; ++y) {
 parallel_for(0, size,[=](int x) {
            float sum = 0;
            for (int i = 0; i < size; ++i) {
   sum += A[y * size + i] * B[i * size + x];
            }
            C[y * size + x] = sum;
 });
    }
}
```

Implémentation du produit matriciel version parallèle sur CPU avec la librairie Microsoft PPL et la méthode parallel_for

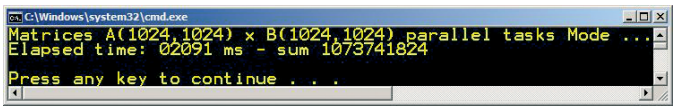


Le programme a tourné en 3 secondes et 448 centièmes, ce qui constitue un résultat raisonnable sur une machine contenant un processeur 4 cœurs physiques.

Nous pouvons tenter de gagner en performance en n'utilisant pas la méthode *parallel_for*, mais le type *task_group* de base de la librairie **PPL**. Le code est un peu plus complexe, mais reste lisible.

```
void MatrixMultiplyTasks(float* A, float* B, float* C, int size)
{
    task_group* task_groups = new task_group[size];

    for (int i = 0; i < size; ++i) {
        task_groups[i].run([=]() {
            for (int j = 0; j < size; ++j) {
                float sum = 0;
                for (int k = 0; k < size; ++k) {
                    sum += A[i * size + k] * B[k * size + j];
        }
        C[i * size + j] = sum;
            }
         });
    }

    for (int i = 0; i < size; ++i) task_groups[i].wait();
    delete [] task_groups;
}
```

Implémentation du produit matriciel version parallèle sur CPU avec la librairie Microsoft PPL et le type task_group.



Le programme a tourné en 2 secondes et 91 centièmes, ce qui est excellent, mais légèrement plus compliqué à implémenter. Les différences de performances entre les deux versions s'expliquent par la simplicité de la seconde implémentation au regard de la méthode parallel_for qui est beaucoup plus complète sur la gestion d'erreurs potentielles ou la capacité à répartir la charge.

EN CONCLUSION

La programmation parallèle orientée données sur CPU est simple à mettre œuvre, mais peut être rapidement limitée par le nombre de cœurs disponibles.

Avec cette dernière implémentation, nous sommes sans doute proches des performances maximum sur un processeur quad-cœurs. Et même si 2 secondes de temps d'exécution, peuvent sembler négligeables vis-à-vis d'une seule exécution, que peut-on dire si notre application engendre des millions de calculs à la fois ?

Sur des machines de type serveur, le nombre de cœurs peut être bien plus élevé, mais le coût des threads sous-jacents peut devenir rapidement pénalisant.

Les architectures massivement multi-cœurs reposent sur des architectures matérielles complexes où des ilots de processeurs/cœurs se partagent la mémoire de la machine, on parle alors d'architecture matérielle type **NUMA** (http://en.wikipedia.org/wiki/Non-Uniform_Memory_Access). Le parallélisme orienté données est souvent touché par le phénomène appelé **False Sharing** (http:/en.wikipedia.org/wiki/False_sharing) qui peut engendrer des pertes de performances non négligeables. Nous sommes alors face à un problème que seules les GPU savent résoudre.

La prochaine fois, nous rentrerons dans le vif du sujet avec une solution reposant sur le Framework **nVIDIA CUDA C**. Nous verrons comment installer les outils **CUDA C** avec Visual Studio 2010. Puis nous étudierons l'architecture CUDA, vis-à-vis des capacités des cartes graphiques et des conséquences sur vos programmes.

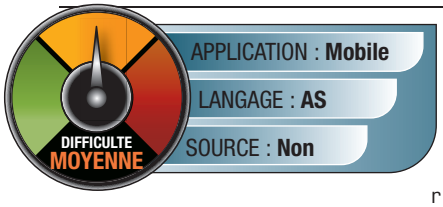
Nous verrons comment calculer le nombre de threads GPU utiles pour notre calcul. Enfin, nous mesurerons les performances de notre algorithme avec la technologie **CUDA**.

La suite le mois prochain.

■ Bruno **Boucard**
Expert en C++ et parallélisme

FLEX Mobile :
Flex sait aussi se faire mobile

Une entreprise qui souhaite cibler un maximum de plateformes mobiles se doit d'investir dans différents frameworks. Effectivement, la programmation mobile est aujourd'hui dépendante de la nécessité de développer plusieurs versions, à destination des différentes plateformes (iPhone, Android…). Une solution pour éviter cette fragmentation est FLEX Mobile.



Flex est une solution de développement créée en 2004, éditée par Adobe. Elle permet de développer des applications Internet riches (RIA) basées sur la technologie Flash, et donc multiplateformes. L'objectif de FLEX est de permettre à des développeurs « classiques » de créer facilement et rapidement des applications Flash. Le développement en FLEX se fait principalement grâce à :

- un langage descriptif : le MXML, qui permet de créer toute l'interface et une partie du comportement de l'application.
- un langage impératif (« traditionnel ») : l'ActionScript, inspiré du JavaScript.

L'exemple ci-dessous présente à la fois la partie descriptive et la partie impérative. On remarque l'utilisation de deux librairies : FlashX (ou fx) pour inclure de l'ActionScript, et Spark (ou s) pour les composants graphiques [Fig.1 et 2].

Fig.1

```
<?xml version="1.0" encoding="utf-8"?>
<s:View xmlns:fx="http://ns.adobe.com/mxml/2009"
        xmlns:s="library://ns.adobe.com/flex/spark" title="HomeView">

    <fx:Script>
        <![CDATA[
            protected function button1_clickHandler(event:MouseEvent):void
            {
                label.text = "Hello, World !";
            }
        ]]>
    </fx:Script>

    <s:VGroup horizontalAlign="center" height="100%" width="100%" gap="30">
        <s:Spacer height="50"/>
        <s:Label id="label"/>
        <s:Button label="Hello !" click="button1_clickHandler(event)"/>
    </s:VGroup>
</s:View>
```

Fig.2

En mai dernier, Adobe a sorti la version 4.5 du SDK FLEX, dont la principale nouveauté est FLEX Mobile. Ce dernier permet de développer des applications FLEX compatibles avec les plateformes Android, iOS (iPhone, iPad, iTouch) et BlackBerry Tablet (Playbook). Outre les facilités offertes par cette nouvelle version, on y trouve ainsi toute une série de librairies spécifiques à nos smartphones et tablettes, notamment pour la gestion du multitouch, de l'accéléromètre, du GPS, etc.

Les applications nécessitent le runtime Air pour fonctionner. Sur un terminal Android, le lancement d'une application FLEX proposera le téléchargement de Air si celui-ci n'est pas installé. Sous iOS, Air n'est pas disponible, et sera inclus directement à l'application lors du déploiement. Enfin, sur la PlayBook de RIM, Air est intégré en natif (une grosse partie de l'interface de l'OS est d'ailleurs basée sur FLEX). Nous allons donc découvrir ici FLEX Mobile, au travers du développement, puis du déploiement, d'une application simple donnant la liste des pays de l'Union Européenne et quelques informations sur ceux-ci. Les données sur les pays seront stockées dans un fichier XML qui sera, avec les images des drapeaux, intégré dans l'application [Fig.3].

Fig.3

```
<europe27>
   <country>
      <name>Allemagne</name>
      <population>83 536 115</population>
      <government>République Fédérale</government>
      <cap>Berlin</cap>
      <flag>assets/de.png</flag>
   </country>
   ...
</europe27>
```

VOTRE PREMIÈRE APPLICATION FLEX

Création du projet

Avant de commencer, vous devez télécharger Flash Builder 4.5 Premium, l'environnement de développement FLEX, basé sur Eclipse. Vous le trouverez sur le site d'Adobe en version d'évaluation de 60 jours : http://www.adobe.com/go/try_flashbuilder/.

Une fois celui-ci installé et lancé, vous vous retrouvez devant une fenêtre très similaire à Eclipse, et une « Start Page ». Celle-ci vous envoie vers différents tutoriaux et exemples de code très utiles pour aller plus loin [Fig.4]. Pour notre projet, sélectionnez File -> New -> Flex Mobile Project. Un assistant vous demande le nom de l'application :

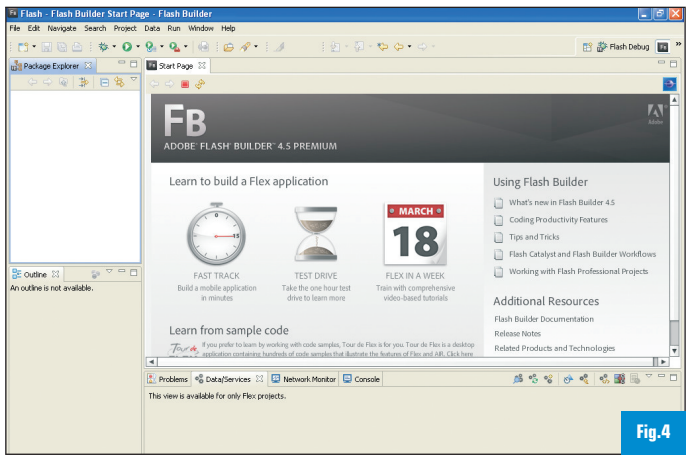


Fig.4

nous l'appellerons « Europe ». Vous devez ensuite choisir les plateformes de destination (pour nous, les trois), le type d'interface (nous choisirons View-Based Application), et les paramètres d'affichage de l'application. Concernant ces derniers, nous choisirons l'orientation automatique (Automatically reorient) et la mise à l'échelle automatique (Automatically scale…, avec une densité de base de 160dpi). Vous avez aussi la possibilité de gérer les permissions selon les plateformes, les terminaux visés, mais nous n'en avons pas besoin ici. Vous pouvez maintenant cliquer sur « Finish » [Fig.5 et 6].

Connexion aux données

Le projet est créé. Vous vous retrouvez avec deux fichiers MXML automatiquement créés : Main.mxml qui est le lanceur de l'application, et MainHomeView.mxml, qui est la fenêtre (ou « View ») principale. Tout d'abord, il faut inclure les données dans l'application. Faites File -> New -> Package. Appelez-le « assets »[Fig.7].

Ensuite, il vous faut créer un fichier XML (nommé europe27.xml par exemple) sur un modèle similaire à celui de la figure 3. Récupérez aussi quelques images de drapeaux. Le champ <flag> du fichier XML doit contenir le chemin du fichier image, sous la forme « assets/nom-du-fichier-image ».

Vous l'avez compris, le fichier XML et les fichiers images doivent être placés dans le package « assets », grâce à un simple glisser-déposer sur le dossier dans l'interface de Flash Builder.

Attaquons maintenant le développement proprement dit, avec la connexion aux données : dans le cadre du bas de Flash Builder, vous trouverez un onglet « Data/Services ». Cliquez sur « Connect to Data/Service… » : un assistant vous demande quel type de données vous souhaitez utiliser. Dans notre cas, il s'agit d'un service XML. Dans la fenêtre suivante, donnez l'emplacement du fichier XML (pour nous, « assets/europe27.xml »). L'assistant parcourt le fichier, affiche les erreurs éventuelles dans le cas d'un XML malformé, et résout automatiquement le nom du service, etc [Fig.8 et 9].

Une fois validé, le service apparaît dans le cadre « Data/Services ». Ouvrez maintenant Main.mxml, puis dans « Data/Services », clic droit sur « getData() : Country[] », puis « Generate Service Call ». Un bloc <fx:Script> contenant une fonction getData() est généré automatiquement : c'est la fonction de récupération des données [Fig.10 et 11].

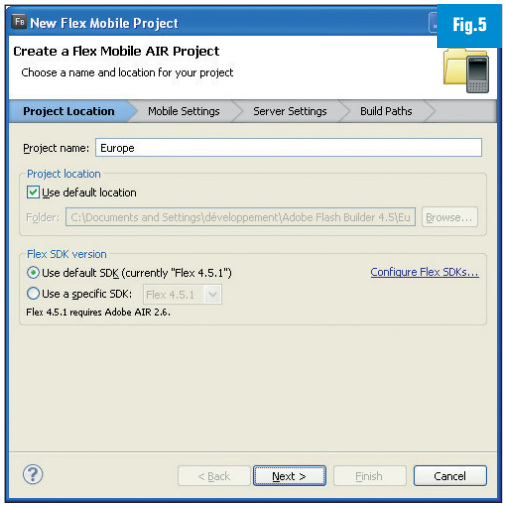
Et maintenant, on code !

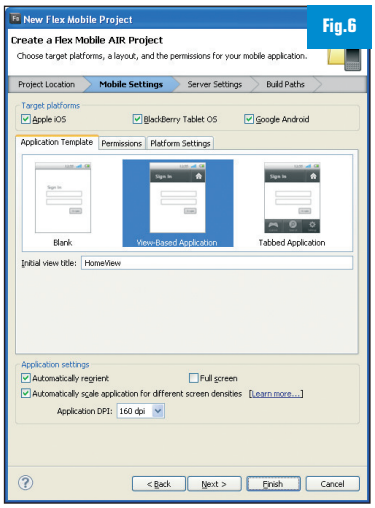
Note : *Comme sous Eclipse, le raccourci Ctrl+Espace affiche l'auto-complétion, et même, comme ici, l'auto-génération de certaines fonctions. Pensez à l'utiliser !*

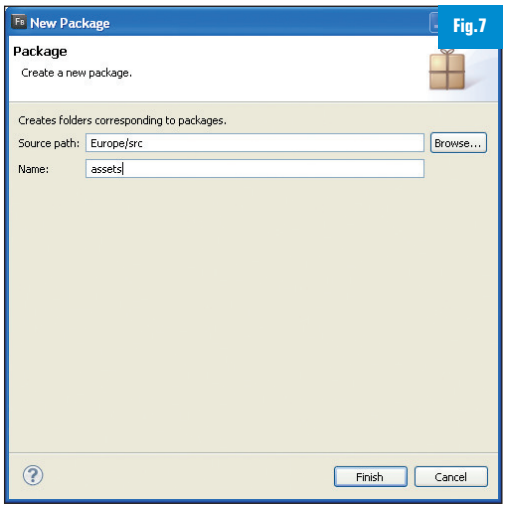
Main.mxml

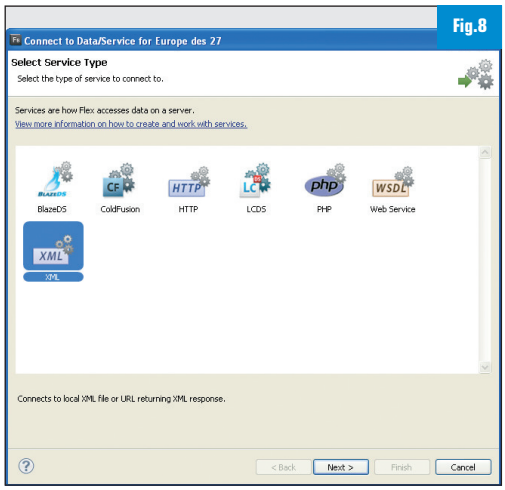
Premièrement, nous voulons que les données provenant du XML soient récupérées avant l'affichage de la fenêtre principale. Dans le fichier Main.mxml, supprimez : « «firstView=»views.MainHomeView» ». A la place, entrez creationComplete=»getData()», ce qui entraîne l'appel de getData() une fois le chargement terminé, au lieu de lancer tout de suite la View principale.

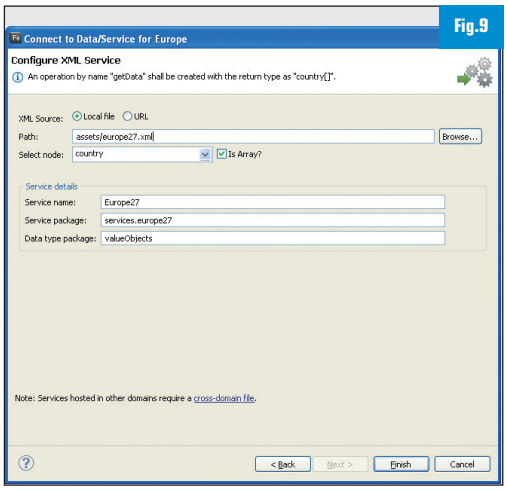
Plus bas dans le code, à l'intérieur du bloc <europe27:Europe27>, juste après « id=»europe27» », rajoutez le paramètre : result=»»,

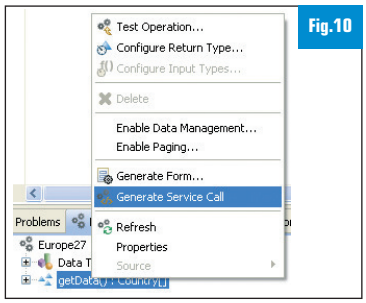


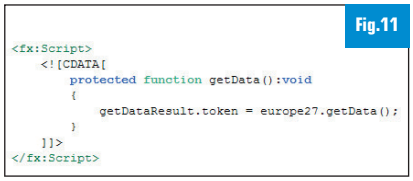












puis placez-vous entre les guillemets, faites Ctrl+Espace pour afficher l'auto-complétion, et choisissez « Generate Result Handler ». Cela vous génère la fonction « europe27_resultHandler(event) ».
Dans cette fonction, nous allons appeler la fenêtre principale, en lui transmettant les données provenant du XML. Pour cela, vous devez saisir « import views.MainHomeView; » en dessous de « import mx.rpc.events.ResultEvent; », et « navigator.pushView(MainHomeView, event.result); » dans la fonction « europe27_resultHandler ».
Voyons cela plus en détail :
La fonction navigator.push(View, Data) « pousse » la View demandée, en lui passant les données spécifiées. A l'inverse, la fonction navigator.popView() détruit la vue courante, et réaffiche la précédente. C'est une gestion par pile, et ces deux fonctions permettent d'empiler et de dépiler les Views. Voici le Main.mxml après ces apports : [Fig.12].

Fig.12

```
<?xml version="1.0" encoding="utf-8"?>
<s:ViewNavigatorApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                            xmlns:s="library://ns.adobe.com/flex/spark"
                            xmlns:europe27="services.europe27.*"
                            creationComplete="getData()">

    <fx:Script>
        <![CDATA[
            import mx.rpc.events.ResultEvent;
            import views.MainHomeView;

            protected function getData():void
            {
                getDataResult.token = europe27.getData();
            }

            protected function europe27_resultHandler(event:ResultEvent):void
            {
                navigator.pushView(MainHomeView, event.result);
            }

        ]]>
    </fx:Script>

    <fx:Declarations>
        <s:CallResponder id="getDataResult"/>
        <europe27:Europe27 id="europe27" result="europe27_resultHandler(event)"/>
    </fx:Declarations>
</s:ViewNavigatorApplication>
```

MainHomeView.mxml

Passons maintenant à MainHomeView.mxml. Cette fenêtre est lancée une fois les données récupérées. Commençons par changer son titre : remplacez « title=»HomeView» » par « title=»Europe des 27» ». Nous allons maintenant lister les pays, en affichant leur drapeau et le nom de la capitale. Pour cela, il faut afficher une liste. Après la ligne « </fx:Declarations> », saisissez la ligne :
« <s:List id=»list» height=»100%» width=»100%» dataProvider=»{data}» itemRenderer=»» /> »
« s:List » représente un composant Spark de type List.
« {data} » représente les données passées à la fenêtre.
Maintenant, placez le curseur entre les guillemets de « itemRenderer=»» » et faites Ctrl+Espace : la liste d'auto-complétion vous propose « Create Item Renderer… ». En validant avec Entrée, un assistant s'ouvre pour la création de l'Item Renderer [Fig.13 et 14]. Entrez le nom choisi, puis indiquez les champs XML en vous aidant de ce schéma : [Fig.15]. Vous êtes maintenant prêt à tester votre application ! Pour ce faire, allez dans Run -> Run. Spécifiez une plateforme à lancer. En choisissant l'option « On Desktop », vous pouvez émuler un terminal sur votre ordinateur. Sélectionnez maintenant le terminal que vous voulez simuler. Votre application se lance, et la liste s'affiche comme prévu, malgré le peu de code écrit. Magique, n'est-ce pas ? [Fig.16]. Et voici le contenu de MainHomeView.mxml : [Fig.17]. Mais nous allons aller plus loin, et faire en sorte qu'un clic sur un pays ouvre une nouvelle fenêtre affichant les détails de ce pays.

DetailsPays.mxml

Créons donc une nouvelle View. Faites File -> New -> MXML Component. Dans l'assistant qui s'affiche, saisissez « views » dans la zone « Package », puis donnez un nom à cette View (ici : « DetailsPays »). Validez, le nouveau fichier DetailsPays.mxml s'ouvre dans l'éditeur [Fig.18]. Commençons par rendre le titre dynamique, lié au pays dont on veut afficher les détails, en remplaçant « title=»DetailPays» » par « title=»{data.name}» ». Encore une fois, l'accès aux données est totalement transparent. Il nous faut maintenant ajouter, à la suite de « </fx:Declarations> », les composants d'affichage : [Fig.19].
Nous voulons à présent afficher le drapeau à droite de la barre de titre, et un bouton Back à gauche. Cette barre de titre est appelée ActionBar, et est divisée en trois parties comme suit : [Fig.20].
Pour le bouton Back, qui sera situé dans le « navigationContent », il faut insérer ceci à la suite du bloc VGroup : [Fig.21].
Ainsi, comme nous l'avons vu plus haut, la View est « dépilée ». Le drapeau, situé dans le « actionContent », nécessitera l'insertion de : [Fig.22]. Le composant « Spacer » permet de laisser un espace entre le drapeau et la limite gauche de l'écran, pour plus d'esthétisme. Voici le contenu du fichier DetailsPays.mxml après nos modifications : [Fig.23]. Et voilà ! L'application est presque terminée. La dernière étape est de lier le clic sur un élément de la liste à la View que l'on vient de créer. Pour cela, dans le fichier MainHomeView.mxml, dans le bloc « <s:List> », ajoutez le paramètre « click=»navigator.pushView (DetailPays, list.selectedItem)» ». Ce code « empile » la View DetailsPays lors d'un clic sur un élément de la liste, en lui passant le nœud de données. C'est tout ! Voici le contenu final de MainHome-

Fig.17

```
<?xml version="1.0" encoding="utf-8"?>
<s:View xmlns:fx="http://ns.adobe.com/mxml/2009"
        xmlns:s="library://ns.adobe.com/flex/spark" title="Europe des 27">
    <fx:Declarations>
    </fx:Declarations>
    <s:List id="list" height="100%" width="100%"
            dataProvider="{data}" itemRenderer="MyRenderer"/>
</s:View>
```

Fig.13

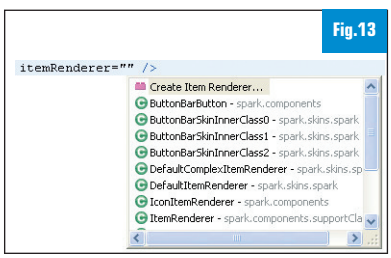


Fig.14

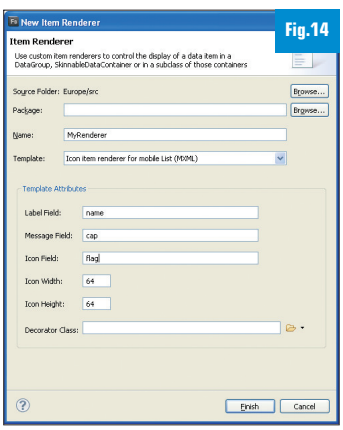


Fig.15

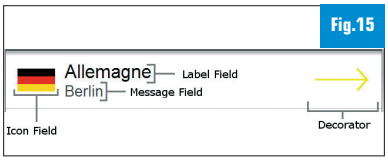
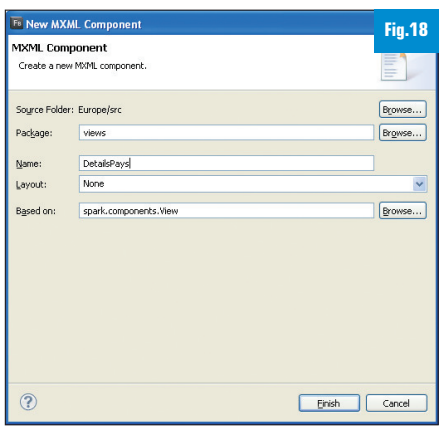


Fig.16

Fig.18



View.mxml : [Fig.24]. Vous pouvez maintenant tester votre application, sous toutes les plateformes que vous souhaitez [Fig.25 et 26].
Nous arrivons à la dernière partie de ce tutoriel : le déploiement.

Déploiement

Note : *Pour déployer sur iOS, il vous faut un certificat Apple iOS Developer.*

Note 2 : *Pour déployer sur Playbook, il vous faut télécharger le BlackBerry Tablet SDK sur* http://us.blackberry.com/developers/tablet/adobe.jsp.

Pour déployer votre application, allez dans Project -> Export Release Build. Dans l'assistant, vous pouvez choisir quelle(s) version(s) déployer, le chemin de destination des exécutables, et si vous voulez les signer immédiatement ou pas. [Fig.26 et 27].
Une fois ces informations saisies et validées, vous pouvez modifier certaines options selon la plateforme cible, et vous êtes invité à fournir le(s) certificat(s) de signature (facultatif pour BlackBerry Tablet, obligatoire pour iOS et Android, avec la possibilité de le créer directement pour ce dernier). Une fois toutes ces informations fournies, cliquez sur Finish, et les exécutables seront générés dans le dossier choisi. Il ne vous reste plus qu'à les installer sur un terminal Android, iOS ou BlackBerry Tablet.

CONCLUSION

Comme nous l'avons vu ici, FLEX Mobile est une technologie puissante, et vraiment simple d'utilisation. On retiendra notamment la transparence de la manipulation des données, que ce soit pour les transmettre à une nouvelle page via « pushView » ou les utiliser via « {data} ». Etant un développeur plutôt « classique » (JAVA, C, etc.), la principale difficulté que j'ai eu lors de ma découverte de FLEX a été de comprendre la logique d'un développement « descriptif » en MXML. Cependant, après une ou deux semaines d'adaptation, le problème avait totalement disparu, éclipsé par la simplicité d'usage. Le développement descriptif est, tout du moins pour l'interface d'une application, extrêmement simple et puissant.
En parlant d'interface, il y a une fonctionnalité de Flash Builder dont je n'ai pas parlé : la possibilité de construire le visuel des Views par simple glisser-déposer. Pour accéder à cette fonctionnalité, il suffit de cliquer sur « Design », en haut du cadre d'édition. Cependant, cette manière de faire a un inconvénient de taille. En effet, les composants ajoutés par glisser-déposer sont placés de manière « absolue » sur la View, c'est-à-dire qu'ils sont situés selon deux axes X et Y, et que leur taille est fixée. Si votre application est censée être déployée sur des appareils de résolutions différentes, ce placement absolu risque de la rendre « mal proportionnée », et peu agréable à l'œil. Par contre, utiliser le mode « Design » pour insérer vos éléments, puis réarranger le code généré pour rendre le placement « relatif » peut être un gain de temps très appréciable.
Pour finir, parlons du potentiel de FLEX Mobile. C'est une solution très complète, associant MXML pour le design et les fonctionnalités de base, et ActionScript pour un comportement plus complexe, de l'algorithmique, ou autre. Les possibilités sont donc très larges, allant de la simple application « donneuse d'infos statiques » que nous avons développée, jusqu'à des systèmes client-serveur complets, en passant par des jeux vidéo. Cette puissance associée à l'indépendance des plateformes font de FLEX Mobile une technologie de développement très intéressante.

■ Christophe **Boucrot** - *Bull Formation*

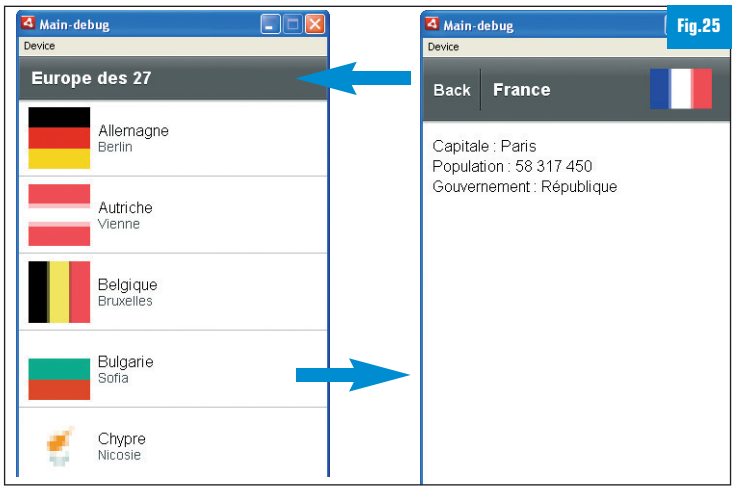


Fig.25

Fig.19

```
<s:VGroup x="10" y="20" width="100%" height="100%">
    <s:Label text="Capitale : {data.cap}" />
    <s:Label text="Population : {data.population}" />
    <s:Label text="Gouvernement : {data.government}" />
</s:VGroup>
```

Fig.20

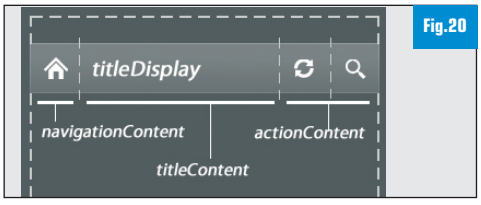


Fig.21

```
<s:navigationContent>
    <s:Button label="Back" click="navigator.popView()"/>
</s:navigationContent>
```

Fig.22

```
<s:actionContent>
    <s:Image source="{data.flag}" height="64" width="64"/>
    <s:Spacer width="20"/>
</s:actionContent>
```

Fig.23

```
<?xml version="1.0" encoding="utf-8"?>
<s:View xmlns:fx="http://ns.adobe.com/mxml/2009"
        xmlns:s="library://ns.adobe.com/flex/spark" title="{data.name}">
    <fx:Declarations>
    </fx:Declarations>

    <s:VGroup x="10" y="20" height="100%" width="100%">
        <s:Label text="{data.cap}"/>
        <s:Label text="{data.population}"/>
        <s:Label text="{data.government}"/>
    </s:VGroup>

    <s:navigationContent>
        <s:Button label="Back" click="navigator.popView()"/>
    </s:navigationContent>

    <s:actionContent>
        <s:Image source="{data.flag}" height="64" width="64"/>
        <s:Spacer width="20"/>
    </s:actionContent>

</s:View>
```

Fig.24

```
<?xml version="1.0" encoding="utf-8"?>
<s:View xmlns:fx="http://ns.adobe.com/mxml/2009"
        xmlns:s="library://ns.adobe.com/flex/spark" title="Europe des 27">
    <fx:Declarations>
    </fx:Declarations>
    <s:List id="list" height="100%" width="100%"
            dataProvider="{data}" itemRenderer="MyRenderer"
            click="navigator.pushView(DetailsPays, list.selectedItem)"/>
</s:View>
```

Fig.26

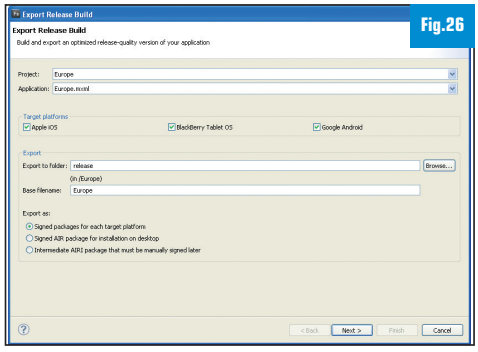
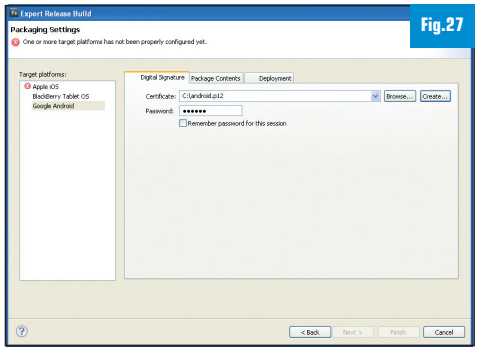


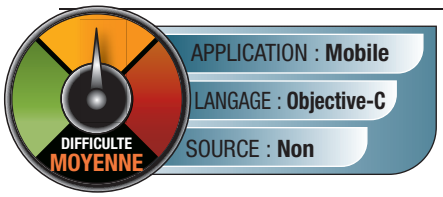
Fig.27



Devenir un développeur iOS

4e partie

Dans nos trois précédents articles, nous avons créé tous les éléments fondamentaux à notre application de gestion de contacts mais il manque encore l'élément le plus important, à savoir le stockage des informations saisies par l'utilisateur dans une base de données.



Ce quatrième et dernier article va donc aborder ce sujet et présenter une des possibilités offerte par iOS en matière de stockage, à savoir la brique « **Core Data** ». D'autres solutions telles que « **SQLite** » existent sur iOS et peuvent être utilisées au sein de vos applications pour stocker des données, mais nous nous concentrerons uniquement sur l'utilisation de *Core Data* pour finaliser notre application de gestion de contacts.

© Apple

PRÉSENTATION DE CORE DATA

Core Data est un framework proposé par Apple aux développeurs iOS et MacOS afin de leur permettre de pouvoir gérer simplement et de manière efficace des données au sein de leur application.
Core Data peut être rangé dans la catégorie des ORM (Object Relational Mapping) et est totalement intégré aux outils Apple, que ce soit avec les systèmes d'exploitation (iOS et MacOS) ou l'outil de développement Xcode. Comme tout outil ORM, Core Data va reposer sur différents concepts importants :

- Modèle de données
- Contexte d'exécution
- Système de persistance

Le modèle de données correspond aux objets métiers (les classes) qui seront manipulées par l'application. Les objets peuvent être autonomes mais il peut exister des relations entre ceux-ci et c'est dans ce cas le framework qui va se charger de gérer celles-ci si vous souhaitez récupérer un objet et ses enfants.
Le contexte d'exécution sera utilisé par les différentes vues/écrans de l'application afin de pouvoir interagir avec les données (sélection, insertion, mise à jour, suppression) mais également gérer les potentiels conflits qui pourraient subvenir lors de la sauvegarde des données. Le système de persistance enfin permet de déterminer sous quelle forme seront sauvegardées les données manipulées par l'application. Core Data supporte des systèmes de persistance en mémoire, sous une forme binaire ou bien au format SQLite.

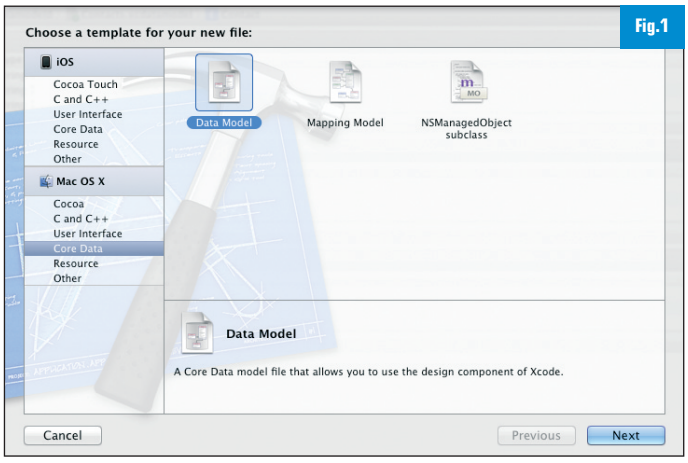


Fig.1

CRÉATION DU MODÈLE DE DONNÉES

La première étape pour utiliser Core Data au sein de notre application va donc être de créer notre modèle de données. Pour ce faire, ajoutons un nouveau fichier à notre projet et sélectionnons le type « **Data Model** » dans la section « **iOS > Core Data** » [Fig.1].
Une fois le fichier nommé, l'éditeur de modèle intégré à Xcode vous est présenté afin que vous puissiez créer vos objets métiers et que vous définissiez les propriétés de ceux-ci. Dans le langage Core Data, un objet s'appelle une « **Entité** », donc commençons par en ajouter une au modèle via le bouton présent en bas à gauche de l'éditeur (**Add Entity**) et nommons celle-ci « **Contact** ». Maintenant que notre entité est créée, il ne reste plus qu'à définir les propriétés de celle-ci. Dans notre application, un contact possède 3 propriétés à savoir un prénom, un nom de famille et un numéro de téléphone, toutes ces propriétés étant de type « **String** ». Ajoutons ces propriété-

tés dans l'éditeur via le bouton « + » présent dans la section « **Attributes** ». Une fois les attributs créés, vous devriez arriver au résultat suivant [Fig.2]. Afin de faciliter la manipulation de notre entité au sein de notre code, il serait utile d'avoir une classe qui correspond en tous points aux propriétés déclarées dans notre entité. Pour ce faire, il suffit de sélectionner une entité et de choisir l'option « **Create NSManagedObject Subclass** » présente dans le menu « **Editor** ». Après avoir nommé votre classe (Contact dans notre cas), les fichiers **Contact.h** et **Contact.m** (dont vous trouverez des exemples ci-dessous) sont automatiquement ajoutés à votre projet.

Le fichier « *Contact.h* »

```
#import <Foundation/Foundation.h>
#import <CoreData/CoreData.h>

@interface Contact : NSManagedObject

@property (nonatomic, retain) NSString *firstName;
@property (nonatomic, retain) NSString *lastName;
@property (nonatomic, retain) NSString *phoneNumber;

@end
```

Le fichier « *Contact.m* »

```
#import "Contact.h"

@implementation Contact

@dynamic firstName;
@dynamic lastName;
@dynamic phoneNumber;

@end
```

DÉCLARATION ET INITIALISATION DES COUCHES COREDATA

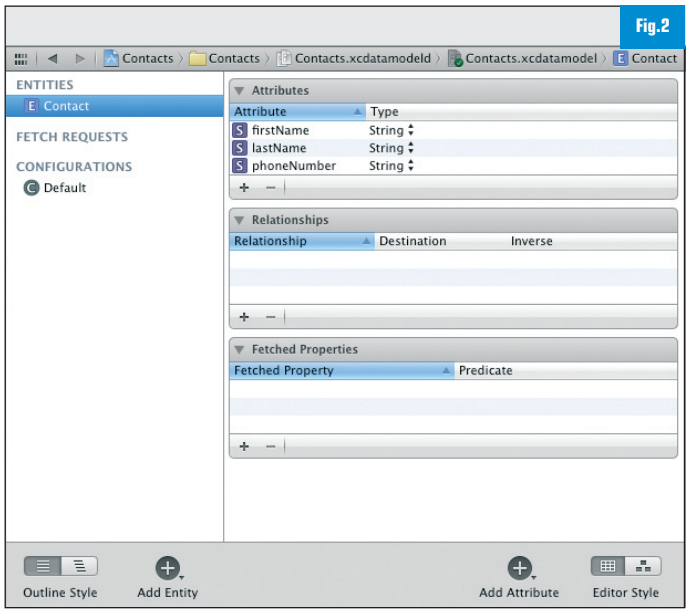
Nous sommes maintenant prêts pour compléter notre application. La première étape sera d'ajouter une référence vers « **CoreData.framework** » au sein des propriétés de notre application [Fig.3].

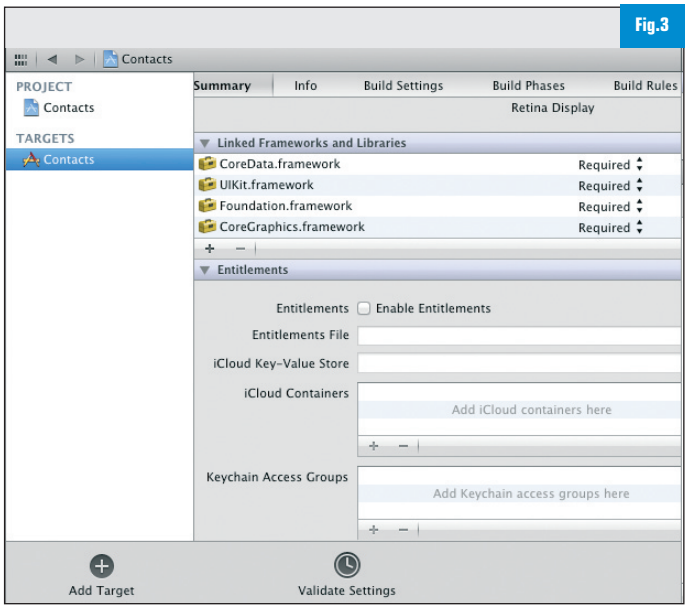
Comme nous l'avons évoqué précédemment, *Core Data* repose sur trois concepts (modèle, contexte et persistance) que nous devons donc déclarer et instancier au sein de notre application. Pour cela, nous allons commencer par ajouter les éléments nécessaires au sein du fichier « **ContactsAppDelegate.h** » afin de pouvoir instancier les objets dès le lancement de notre application.

```
#import <UIKit/UIKit.h>
#import <CoreData/CoreData.h>

@interface ContactsAppDelegate : NSObject <UIApplicationDelegate>

@property (nonatomic, retain) IBOutlet UIWindow *window;
@property (nonatomic, retain) IBOutlet UINavigationController
*navigationController;

@property (retain, nonatomic) NSManagedObjectContext *managed
ObjectContext;
@property (retain, nonatomic) NSManagedObjectModel *managed
ObjectModel;
@property (retain, nonatomic) NSPersistentStoreCoordinator
*persistentStoreCoordinator;

- (IBAction)addContact:(id)sender;

@end
```

Comme nous le voyons ci-dessus, nous déclarons trois propriétés correspondant aux trois concepts de *Core Data*. Reste maintenant à instancier ces objets au lancement de l'application, ce qui se fait dans la méthode « **didFinishLaunchingWithOptions** ».

```
- (BOOL)application:(UIApplication *)application didFinish
LaunchingWithOptions:(NSDictionary *)launchOptions
{
 NSError *error = nil;

 NSURL *documentsDirectory = [[[NSFileManager defaultManager]
URLsForDirectory:NSDocumentDirectory inDomains:NSUserDomain
```





```
Mask] lastObject];
 NSURL *storeURL = [documentsDirectory URLByAppendingPath
Component:@"Contacts.sqlite"];

 NSURL *modelURL = [[NSBundle mainBundle] URLForResource:@
"Contacts" withExtension:@"momd"];
 m_ManagedObjectModel = [[NSManagedObjectModel alloc] initWith
ContentsOfURL:modelURL];
 m_PersistentStoreCoordinator = [[NSPersistentStoreCoordinator
alloc] initWithManagedObjectModel:m_ManagedObjectModel];

 if (![m_PersistentStoreCoordinator addPersistentStoreWithType:
NSSQLiteStoreType configuration:nil URL:storeURL options:nil
error:&error])
 {
   NSLog(@"Unresolved error %@ - %@", error, [error userInfo]);
   abort();
 }

 m_ManagedObjectContext = [[NSManagedObjectContext alloc] init];
 [m_ManagedObjectContext setPersistentStoreCoordinator:m_
PersistentStoreCoordinator];

 RootViewController *rvc = [self.navigationController.view
Controllers objectAtIndex:0];
 rvc.managedObjectContext = m_ManagedObjectContext;

 self.window.rootViewController = self.navigationController;
 [self.window makeKeyAndVisible];

 return YES;
}
```

Nous commençons par construire une URL correspondant au chemin dans lequel sera stocké le fichier dans lequel seront écrites les données de notre application. Nous utiliserons le format SQLite comme système de persistance, d'où le nom donné au fichier et qui sera stocké dans le répertoire des documents de l'application.
Nous récupérons ensuite le chemin de notre modèle et nous instancions celui-ci ainsi que le système de persistance en lui spécifiant que nous souhaitons utiliser notre modèle.
A noter que nous avons mis en place une gestion d'erreur sommaire qui fait que si une erreur se produit (pour quelque raison que ce soit), nous quittons simplement l'application. Dans la pratique, une gestion d'erreur plus adaptée serait à mettre en œuvre. Nous finissons enfin par instancier le contexte qui sera utilisé par nos vues pour manipuler les données de notre application. Nous affectons ce contexte à une propriété du contrôleur de notre vue principale (propriété que nous avons ajouté à cet effet au préalable) afin que celle-ci puisse interroger les données et puisse en afficher la liste. Nous avons adapté un peu la méthode « addContact » qui est appelée lors de l'ajout d'un contact, afin de pouvoir manipuler le contexte dans la vue de création d'un contact et nous indiquons également que nous passons en mode édition à l'aide d'une variable booléenne (pour faire la distinction avec le mode consultation).

```
- (IBAction)addContact:(id)sender
{
 ViewContactController *vcc = [[ViewContactController alloc] init];
 vcc.managedObjectContext = m_ManagedObjectContext;
 vcc.editMode = YES;
 [self.navigationController pushViewController:vcc animated:YES];
 [vcc release];
}
```

RÉCUPÉRATION ET AFFICHAGE DE LA LISTE DES CONTACTS

Passons maintenant dans la vue principale de notre application qui affichera la liste des contacts. Commençons par modifier la déclaration du contrôleur en ajoutant un tableau qui sera en charge de stocker la liste des contacts, liste récupérée depuis *Core Data*. Nous ajoutons également une méthode « refreshData » qui sera en charge d'interroger Core Data pour remplir le tableau créé juste au-dessus.

```
@interface RootViewController : UITableViewController
{
 NSArray *contactsArray;
}

@property (retain, nonatomic) NSManagedObjectContext *managed
ObjectContext;

- (void)refreshData;

@end
```

Passons à l'implémentation de la méthode « refreshData » afin d'interroger nos données et pour cela utilisons les outils mis à notre disposition par *Core Data* et plus particulièrement la classe « **NSFetchRequest** ». Cette classe permet d'interroger notre source de données et de faire tous les traitements de base nécessaires tels que les filtres ou les tris mais pour cela il est d'abord nécessaire de savoir quelle entité nous désirons manipuler. Nous commençons donc par demander de récupérer la description de l'entité Contact que nous avons créée dans notre modèle à travers le contexte d'exécution courant (accessible via la propriété que nous avons déclarée dans notre classe).
Nous instancions ensuite un objet de type NSFetchRequest auquel nous demandons d'effectuer un tri sur les données, en se basant sur la propriété « firstName » et en classant par ordre croissant (de A à Z). Nous finissons par exécuter la requête et nous stockons le résultat dans le tableau que nous avons déclaré au sein de notre contrôleur.

```
- (void)refreshData
{
 NSError *error = nil;
 NSEntityDescription *entityDescription = [NSEntityDescription
entityForName:@"Contact" inManagedObjectContext:m_ManagedObject
Context];
 NSFetchRequest *request = [[[NSFetchRequest alloc] init] autorelease];
 [request setEntity:entityDescription];
 NSSortDescriptor *sortDescriptor = [[NSSortDescriptor alloc]
initWithKey:@"firstName" ascending:YES];
 [request setSortDescriptors:[NSArray arrayWithObject:sort
Descriptor]];
```

```
 [sortDescriptor release];
 contactsArray = [[m_ManagedObjectContext executeFetchRequest:
request error:&error] retain];
}
```

Maintenant que nous sommes en mesure de récupérer les données et que celles-ci sont stockées dans un tableau, il nous faut faire en sorte que notre tableau aille chercher les données à afficher au sein de ce tableau (nous affichions des données statiques jusqu'ici).
Pour cela, les modifications à apporter à notre code sont mineures et vont porter uniquement sur les méthodes « **numberOfRowsIn-Section** » et « **cellForRowAtIndexPath** ». Dans la première, nous renvoyons simplement le nombre de lignes présentes dans le tableau. Pour la seconde, nous remplaçons le code qui affichait le texte statique pour récupérer dans le tableau, la ligne devant être affichée, puis nous concaténons le prénom et le nom de famille pour les afficher dans la cellule.

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRows
InSection:(NSInteger)section
{
 return [contactsArray count];
}

- (UITableViewCell *)tableView:(UITableView *)tableView cell
ForRowAtIndexPath:(NSIndexPath *)indexPath
{
 static NSString *CellIdentifier = @"myCell";

 UITableViewCell *cell = [tableView dequeueReusableCellWith
Identifier:CellIdentifier];
 if (cell == nil)
 {
   cell = [[[UITableViewCell alloc] initWithStyle:UITableView
CellStyleDefault reuseIdentifier:CellIdentifier] autorelease];
 }

 Contact *aContact = [contactsArray objectAtIndex:indexPath.row];
 cell.textLabel.text = [NSString stringWithFormat:@"%@ %@", a
Contact.firstName, aContact.lastName];

 return cell;
}
```

Il nous faut également modifier de la même manière la méthode permettant d'afficher les détails d'un contact afin de récupérer la ligne ayant été sélectionnée par l'utilisateur pour initialiser les champs de texte de la vue en conséquence.

```
- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:
(NSIndexPath *)indexPath
{
 Contact *aContact = [contactsArray objectAtIndex:indexPath.row];

 ViewContactController *vcc = [[ViewContactController alloc] init];
 vcc.managedObjectContext = m_ManagedObjectContext;
 [self.navigationController pushViewController:vcc animated:YES];
 vcc.firstNameLabel.borderStyle = UITextBorderStyleNone;
 vcc.firstNameLabel.text = aContact.firstName;
 vcc.firstNameLabel.enabled = NO;

 vcc.lastNameLabel.borderStyle = UITextBorderStyleNone;
 vcc.lastNameLabel.text = aContact.lastName;
 vcc.lastNameLabel.enabled = NO;

 vcc.phoneNumberLabel.borderStyle = UITextBorderStyleNone;
 vcc.phoneNumberLabel.text = aContact.phoneNumber;
 vcc.phoneNumberLabel.enabled = NO;

 [vcc release];
}
```

Dernier détail avant de pouvoir exécuter notre application, faire en sorte que les données soient chargées lors de l'affichage initial de notre vue. Pour cela il suffit d'appeler la méthode « **refreshData** » dans la méthode « **viewDidLoad** » de notre contrôleur.

```
- (void)viewDidLoad
{
 [super viewDidLoad];
 [self refreshData];
}
```

AJOUT D'UN NOUVEAU CONTACT

Notre application, bien que déjà fonctionnelle n'affichera aucune donnée puisque notre base de contacts est vide. Voyons comment gérer l'ajout de nouveaux contacts au sein de celle-ci.
Commençons par effectuer quelques modifications au sein de la vue en charge de créer/afficher les contacts. Ajoutons deux propriétés pour récupérer le contexte Core Data (qui nous permettra d'insérer des données) et savoir si nous sommes en mode de création ou de visualisation de données, ainsi qu'une méthode qui sera chargée de créer les données dans notre base.

```
@interface ViewContactController : UIViewController

@property (retain, nonatomic) IBOutlet UITextField *firstName
Label;
@property (retain, nonatomic) IBOutlet UITextField *lastName
Label;
@property (retain, nonatomic) IBOutlet UITextField *phoneNumber
Label;

@property (retain, nonatomic) NSManagedObjectContext *managed
ObjectContext;
@property BOOL editMode;

- (void)addContact;

@end
```

Si nous sommes en mode de création de données, nous aurons besoin d'un bouton permettant de sauvegarder celles-ci. Pour cela, ajoutons un bouton à notre barre de navigation (le classique bouton bleu avec le texte « **Done** » à l'intérieur). Lorsque l'utilisateur cliquera

sur ce bouton, la méthode « addContact » déclarée précédemment sera exécutée.

```
- (void)viewDidLoad
{
 [super viewDidLoad];

 if(editMode)
 {
   UIBarButtonItem *rightButton = [[[UIBarButtonItem alloc] init
WithBarButtonSystemItem:UIBarButtonSystemItemDone target:self
action:@selector(addContact)] autorelease];
   self.navigationItem.rightBarButtonItem = rightButton;
 }
}
```

Voyons maintenant comment ajouter un contact au sein de notre base. Pour cela rien de plus simple via *Core Data*, où nous commençons par récupérer une description pour un nouvel objet correspondant à l'entité « **Contact** » de notre modèle.
Nous complétons ensuite les données de notre contact à partir des données saisies dans les champs de texte et nous sauvegardons les données (toujours avec une gestion d'erreur minimale et devant être adaptée aux besoins). Pour finir, nous demandons de rafraîchir les données de tableau et nous faisons disparaître la vue de création d'un nouveau contact en appelant la méthode « popViewControllerAnimated » de notre contrôleur de navigation.

```
- (void)addContact
{
 Contact *newContact = [NSEntityDescription insertNewObjectFor
EntityForName:@"Contact" inManagedObjectContext:managedObject
Context];
 newContact.firstName = firstNameLabel.text;
 newContact.lastName = lastNameLabel.text;
 newContact.phoneNumber = phoneNumberLabel.text;

 NSError *error = nil;
 if(![managedObjectContext save:&error])
 {
   NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
   abort();
 }

 RootViewController *rvc = [self.navigationController.view
Controllers objectAtIndex:0];
 [rvc refreshData];
 [rvc.tableView reloadData];

 [self.navigationController popViewControllerAnimated:YES];
}
```

SUPPRESSION D'UN CONTACT

Dernière étape dans la réalisation de notre application, la gestion de la suppression d'un contact avec la méthode classique sur les tableaux au sein des applications iOS (faire un swipe sur la ligne voulue). Pour cela nous devons implémenter deux méthodes au sein de notre contrôleur à savoir « canEditRowAtIndexPath » qui permet de déterminer si une ligne peut être éditée ou non, et « commitEditingStyle » qui permet de déterminer ce qui se passe lorsque l'utilisateur valide ses modifications.
Pour la première méthode, nous autorisons la modification de toutes les lignes en renvoyant systématiquement la valeur YES.
Pour la seconde méthode, s'il s'agit d'une opération de suppression, alors nous déterminons quel contact doit être supprimé avant de faire appel à Core Data pour supprimer de manière définitive l'élément de la base de données.
Nous finissons par rafraîchir les données une fois la suppression terminée.

```
- (BOOL)tableView:(UITableView *)tableView canEditRowAtIndex
Path:(NSIndexPath *)indexPath
{
 return YES;
}

- (void)tableView:(UITableView *)tableView commitEditingStyle:
(UITableViewCellEditingStyle)editingStyle forRowAtIndexPath:
(NSIndexPath *)indexPath
{
 if (editingStyle == UITableViewCellEditingStyleDelete)
 {
   Contact *aContact = [contactsArray objectAtIndex:indexPath.row];
   [m_ManagedObjectContext deleteObject:aContact];

   NSError *error = nil;
   if(![m_ManagedObjectContext save:&error])
   {
     NSLog(@"Unresolved error %@, %@", error, [error userInfo]);
     abort();
   }

   [self refreshData];
   [tableView reloadData];
 }
}
```

CONCLUSION

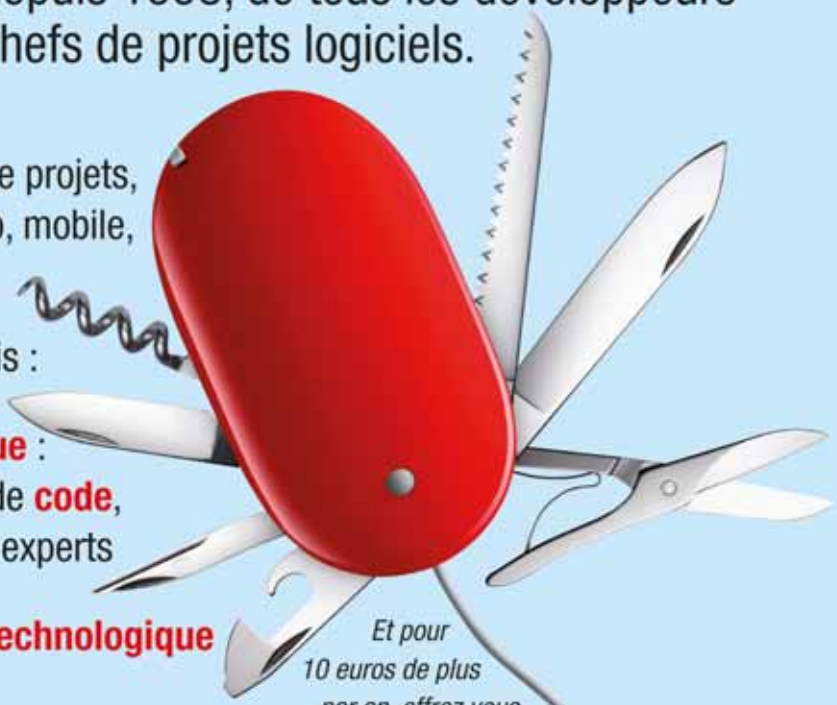
Nous voilà arrivés au terme de notre application qui permet désormais de gérer une liste de contacts (ajout/suppression), et de stocker ces données dans une base en utilisant le framework Core Data.
De nombreuses améliorations peuvent évidemment être apportées à cette application (gestion des erreurs, éviter un rechargement systématique de l'ensemble des données…) mais le but ici était surtout d'introduire les concepts importants pour vous permettre de réaliser vos propres applications iOS.

■ Stéphane **Cordonnier**
Directeur – iTouch
http://www.itouchconsulting.com

TOUT SAVOIR POUR TOUT FAIRE

Développez votre savoir-faire !

Programmez ! est le magazine de référence,
depuis 1998, de tous les développeurs
et chefs de projets logiciels.



Code, gestion de projets,
développement web, mobile,

Programmez ! est à la fois :

- votre **outil pratique** :
  articles de **code**,
  par les meilleurs experts
- votre **veille technologique**

Et pour 10 euros de plus par an, offrez vous l'accès illimité à toutes les archives et numéros en format électronique.

Abonnez-vous à partir de 4€ seulement par mois

ABONNEZ-VOUS EN LIGNE

www.programmez.com

Toutes les offres sont en ligne

☐ OUI, je m'abonne

à retourner, avec votre règlement à :
Groupe GLI, 17 chemin des Boulangers 78926 Yvelines cedex 9

☐ *Abonnement 1 an : 49€ 11 numéros par an au lieu de 65,45€, prix au numéro (*)*

☐ *Abonnement 2 ans au magazine : 79€ (*)*

☐ *Abonnement intégral :1 an au magazine + archives : 59€ (*)*

(Tarif France métropolitaine)*

☐ M. ☐ Mme ☐ Mlle Société .....

Titre : ..... Fonction : ..... Adresse mail .....

NOM ..... Prénom .....

N° ..... rue .....

Complément .....

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

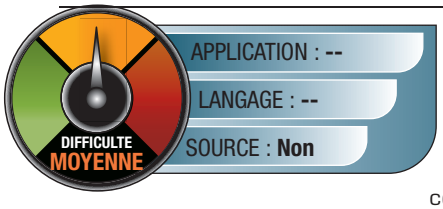
☐ *Je joins mon réglement par chèque à l'ordre de PROGRAMMEZ* ☐ *Je souhaite régler à réception de facture* (écrire en lettres capitales)

Offres valables jusqu'au 31/01/2012

Git : Boostez votre gestion de configuration

2e partie

Dans la première partie de cet article consacré à Git, nous avons mis en avant son architecture et le modèle distribué sur lequel il repose. Dans ce second volet, nous allons passer à la pratique afin de mieux percevoir la puissance et la flexibilité qui font de Git aujourd'hui l'outil en vogue dans le domaine de la gestion de configuration.



A la fin du premier article, nous avions créé notre premier dépôt local pour notre projet au sein duquel nous avions réalisé un premier commit contenant un fichier README. A partir de ce projet, nous continuons notre découverte de l'utilisation de Git en modifiant le contenu du projet.

TRAVAIL LOCAL

Comme expliqué précédemment, 95% des opérations réalisées par Git sont exécutées en local. De fait, le travail au quotidien avec Git se concentre essentiellement sur le poste du développeur et les échanges réseaux sont restreints. Au sein de notre projet, nous allons rajouter un dossier src/ qui contiendra les sources du projet ainsi qu'un dossier conf/ qui contiendra les fichiers de configuration nécessaires au projet. Au sein de ces répertoires, nous ajoutons les fichiers src/Main.java qui constituera une simple classe Java affichant un message sur la sortie standard et conf/myProject.properties contenant le numéro de version du projet. Ces modifications réalisées, nous exécutons la commande *git status* afin de voir les différences introduites au sein du projet par rapport au dernier commit qui est pointé par la référence symbolique HEAD. En sortie de console, Git nous indique que sur la branche courante master sur laquelle nous travaillons, il n'y a aucun fichier dans l'index mais que des fichiers non suivis ont été ajoutés au projet.

AJOUT DE FICHIERS

La notion d'état d'un fichier est essentielle pour bien comprendre le fonctionnement de Git dans le travail quotidien que l'on effectue avec. Globalement, Git considère que chaque fichier présent dans votre répertoire de travail peut avoir 2 états : sous suivi de version ou non suivi. Les fichiers suivis sont ceux qui appartenaient au dernier snapshot commité depuis la zone d'index. Ceux-ci peuvent être inchangés, modifiés ou bien indexés, c'est-à-dire qu'ils feront partie du prochain snapshot que vous commiterez. Ainsi, tous les fichiers présents au sein du répertoire de travail mais non présents dans le dernier snapshot et non indexés sont considérés comme non suivis. Une fois un fichier suivi modifié, il est donc nécessaire de l'ajouter à l'index afin que son snapshot au moment de l'ajout à l'index soit inclus dans le prochain commit. Le cycle de vie d'un fichier du point de vue de Git que nous venons de détailler est illustré à la [Fig.1].

Pour notre projet, il est donc nécessaire d'exécuter la commande *git add conf/ src/* ce qui a pour effet d'ajouter ces répertoires et leur contenu au sein de l'index et va conduire Git à considérer que de nouveaux fichiers ont été ajoutés au sein de notre projet. Une fois le fichier Main.java compilé, le répertoire contient un fichier Main.class produit par le compilateur Java. Ces fichiers n'ont pas vraiment d'intérêt dans le cadre de la gestion de version de notre projet, il est donc préférable d'indiquer à Git de les ignorer. Pour ce faire, il suffit de créer un fichier .gitignore à la racine du projet au sein duquel nous détaillons les noms des fichiers que Git doit ignorer. L'utilisation d'expression régulière étant supportée, notre fichier contient l'entrée suivante : *.class. L'étape suivante consiste à commiter le snapshot du projet indexé au sein du dépôt local à l'aide de la commande suivante :

```
git commit -m 'Ajout des fichiers Main.java et myProject.properties'
```

ANNULER UNE ACTION

Un commit effectué, il se peut que vous réalisiez que celui-ci ne contient pas un fichier précis que vous avez omis. Si votre commit n'a pas été publié, il reste possible de rajouter ce fichier pour qu'il soit inclus dans ce commit. Pour ce faire, il faut ajouter ledit fichier à l'index avant d'utiliser la commande *git commit –amend*. D'autre part, il est également possible d'annuler les modifications apportées par un commit non publié en utilisant la commande revert et en précisant l'identifiant SHA-1 de ce dernier. Dans notre cas pour annuler le dernier commit, il faut donc exécuter la ligne suivante :

```
git revert HEAD
```

VISUALISATION DES DIFFÉRENCES

Ajoutons maintenant un nouveau fichier à la racine du projet, que nous indexons, et modifions le contenu du fichier src/Main.java. Ces modifications faites, l'utilisation de la commande *git status* per-

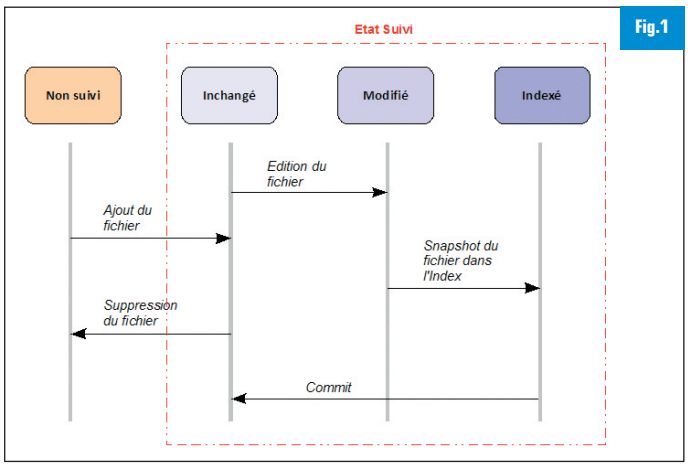


Fig.1

Cycle de vie du statut d'un fichier

met d'avoir un aperçu de l'état de notre répertoire de travail. Cependant, cela se cantonne au niveau des fichiers et non de leur contenu. Il peut donc s'avérer intéressant d'inspecter le contenu des modifications indexées et non indexées de ces fichiers. Pour ce faire, Git met à disposition la commande *git diff*. Celle-ci va permettre dans un premier temps de visualiser ce qui a été modifié mais non encore indexé au sein du répertoire de travail de manière détaillée [Fig.2].

La commande produit un détail visuel des ajouts (en vert) et des suppressions (en rouge) au sein des fichiers concernés. Il est important de noter que *git diff* montre seulement les modifications non indexées et non les modifications réalisées depuis le dernier commit. Ainsi, si tous les fichiers modifiés ont été indexés, *git diff* n'indiquera aucun changement. Afin de visualiser les changements indexés [Fig.3], il faut utiliser cette même commande avec l'option –cached comme suit : *git diff –cached*.

Il est également bon de garder à l'esprit qu'un fichier indexé qui serait ensuite modifié apparaîtra 2 fois au sein du résultat produit par git status par exemple, et donc à la fois dans les résultats de *git diff* et *git diff –cached* respectivement pour les parties des fichiers concernées par ces commandes.

SUPPRESSION DE FICHIERS

Au cours du travail avec Git et des diverses modifications apportées aux fichiers du projet en préparation du futur commit, il se peut qu'un fichier indexé ne doive finalement pas être intégré au sein du snasphot qui fera partie du prochain commit. Le retrait de ce fichier de l'index se fait à l'aide de la commande git reset *<nom du fichier>*. Plus généralement, afin de vider l'index de son contenu, il faudra utiliser cette commande sans préciser de nom de fichier spécifique. Cette commande fait partie des rares commandes Git pouvant potentiellement vous faire perdre des fichiers ou du contenu au sein d'un projet. Enfin, pour écraser toutes les modifications locales de l'espace de travail et y restaurer le contenu d'un commit particulier l'option –hard est nécessaire. Ainsi, la restauration du contenu du HEAD depuis le dépôt local va être réalisée comme suit :

```
git reset --hard HEAD
```

Pour supprimer un fichier d'un projet, la simple suppression de celui-ci du répertoire de travail n'est pas suffisante. En effet, dans ce cas-là cette suppression ne sera pas considérée comme indexée et donc non commitée au sein du dépôt local. Afin que la suppression soit prise en compte, il est nécessaire de supprimer le fichier du répertoire de travail mais également des fichiers *suivi* et donc de la zone d'index avant la réalisation du commit. Ce travail est dévolu à la commande *git rm <nom du fichier>*. En outre, il peut être utile d'abandonner le suivi d'un fichier tout en le conservant dans le répertoire de travail. Ce cas peut se produire lorsque l'on a réalisé un commit englobant des fichiers indésirables dont on a oublié de préciser à Git de les ignorer. Pour ce faire, on utilisera la commande *git rm – cached <nom du fichier>*. Quant au déplacement de fichier, il bénéficie également d'une commande dédiée utilisée comme suit : *git mv <fichier src> <fichier dest>*.

Fig.2

```
$ git diff
diff --git a/src/Main.java b/src/Main.java
index 9ab4b54..0db2bbc 100644
--- a/src/Main.java
+++ b/src/Main.java
@@ -2,7 +2,8 @@
 public class Main {

         public static void main(String[] args) {
-                System.out.println("Hello World !");
+                System.out.print("Hello World !");
+                System.out.println(" Sylvain SAUREL");
         }

 }
```

Visualisation du contenu modifié mais non indexé

Fig.3

```
$ git diff --cached
diff --git a/build.xml b/build.xml
new file mode 100644
index 0000000..af47e28
--- /dev/null
+++ b/build.xml
@@ -0,0 +1,2 @@
+<project name="My First Project">
+</project>
```

Visualisation du contenu modifié et indexé

CONSULTATION DE L'HISTORIQUE

Une fois plusieurs commits créés, il devient souvent intéressant de pouvoir revoir les différentes actions effectuées sur le dépôt du projet. La consultation de l'historique des commits se fait grâce à la commande *git log* qui possède un grand nombre d'options facilitant son usage. Ainsi, *git log* listera l'ensemble des commits réalisés en ordre chronologique inversé sans détailler le contenu présent dans ces commits. Le rajout de l'option -p met en évidence le détail des différences introduites entre chaque commit. Cependant, lister l'ensemble des commits n'est pas forcément utile et dans ce cas l'option -<n> limite le résultat produit aux n derniers commits. Dans le même ordre d'idée, l'option –since permet de réduire l'historique aux commits réalisés depuis une date précise ou une durée. Les statistiques liées à chaque commit sont accessibles via l'option –stat. Enfin, Git est également flexible quant à la visualisation de l'historique du dépôt puisque l'option –pretty offre des possibilités de mise en forme avancées pour la sortie de la commande *git log*. Enfin, notons la commande *git blame* qui détaille pour chaque ligne d'un fichier donné dans quel commit elle a été introduite et qui est son auteur.

COLLABORER AVEC DES DÉPÔTS DISTANTS

Pour pouvoir collaborer au sein d'un projet géré par Git, il est nécessaire d'échanger avec des dépôts distants. Ces derniers peuvent être considérés comme des versions du projet sur le réseau. La collaboration entre dépôts distants peut être réalisée en lecture seule ou bien en lecture / écriture. Echanger avec ces dépôts nécessite de savoir comment les gérer au sein de votre dépôt local pour publier vos modifications ou bien récupérer les leurs. Dans un premier temps, nous allons créer un premier dépôt distant via le clone du dépôt de notre projet. A ce titre, il faut créer un répertoire clonedProject/ en dehors du répertoire de notre projet au sein duquel la commande *git clone myFirstProject/* est lancée. Au sein du clone de notre projet initial, nous allons pouvoir lister les dépôts distants grâce à la commande *git remote*. Celle-ci affiche le dépôt nommé origin qui correspond au dépôt origine, celui à partir duquel le projet a été cloné. L'ajout d'un dépôt distant se fait à l'aide de cette même commande en précisant le nom et l'url du dépôt comme suit : *git remote add <nom> <url>*. Dans le cadre de notre projet, la création d'un troisième dépôt peut se faire en clonant le second que nous venons de créer puis en l'ajoutant aux dépôts distants du dépôt du projet initial. Pour avoir des informations à propos d'un dépôt distant particulier, il suffit d'utiliser la commande git remote show *<nom du depot distant>*. Enfin, il reste possible d'arrêter de suivre les modifications d'un dépôt distant en le retirant de la liste des dépôts suivis par votre dépôt local en appliquant la commande *git rm <nom du depot distant>*.

RÉCUPÉRER DES MODIFICATIONS

La collaboration sur un projet implique bien évidemment de récupérer plus ou moins régulièrement les modifications effectuées sur le projet sur les dépôts distants. Dans ce but, nous modifions le contenu du fichier src/Main.java et ajoutons un nouveau fichier source Message.java au sein du projet sur le dépôt que nous avons cloné en second. Ces fichiers sont ensuite commités directement en sautant la zone d'index grâce à l'ajout de l'option -a comme suit :

```
git commit -a -m 'Rajout de la classe Message'
```

Ces modifications commitées, nous revenons au sein du dépôt du projet initial à partir duquel nous allons tenter de récupérer les modifications distantes réalisées. Celles-ci s'effectuent à l'aide de la commande git fetch *<nom du dépôt distant>* qui va récupérer les modifications distantes au sein de votre dépôt local. Ces modifications ne sont pas fusionnées automatiquement dans votre HEAD mais simplement stockées en vue d'un *merge* explicite qu'il faudra réaliser. Leur stockage s'effectue au sein d'une branche spécifique dont le nom correspond à celui du dépôt dont les modifications sont issues, suffixé par le nom de la branche sur le dépôt distant. Ici, nous avons récupéré la branche courante master du dépôt distant repo1. De ce fait, la branche a pour nom repo1/master et est considérée comme une branche distante pour notre dépôt local. Avant de réaliser l'intégration de ces modifications, on va inspecter les différences entre ce que nous venons de récupérer et le contenu de notre HEAD grâce à l'utilisation de la commande *git diff* comme suit : *git diff repo1/master*. En sortie, Git nous permet de visualiser les différences entre le HEAD de notre dépôt local et les modifications récupérées depuis le dépôt distant [Fig.4].

La visualisation des différences ne montre aucun conflit, c'est-à-dire que les différences entre les fichiers ne concernent pas des lignes identiques. De fait, ce type de merge est dit "fast-forward" et va donc pouvoir être réalisé automatiquement avec la commande git merge de Git de la manière suivante :

```
git merge repo1/master HEAD
```

La commande git pull vient combiner ces 2 commandes et permet de réaliser à la fois la récupération des données et la fusion de celles-ci au sein du HEAD du dépôt local à condition que le merge à réaliser soit de type "fast-forward". Cependant, il reste préférable d'utiliser les 2 commandes séparées comme nous l'avons fait ici afin de bien comprendre le travail réellement réalisé.

Fig.4

```
$ git diff repo1/master HEAD
diff --git a/src/Main.java b/src/Main.java
index 3706f52..0db2bbc 100644
--- a/src/Main.java
+++ b/src/Main.java
@@ -1,10 +1,9 @@

 public class Main {
-
-       private static Message message = new Message();

        public static void main(String[] args) {
-               System.out.println(message.sayHelloTo("Sylvain SAUREL"));
+               System.out.print("Hello World !");
+               System.out.println(" Sylvain SAUREL");
        }

 }
diff --git a/src/Message.java b/src/Message.java
deleted file mode 100644
index 6932fd0..0000000
--- a/src/Message.java
+++ /dev/null
@@ -1,11 +0,0 @@
-
-public class Message {
-
-       public Message() {
-       }
-
-       public String sayHelloTo(String who) {
-               return "Hello to " + who;
-       }
-
```

Visualisation des modifications avant le merge

PUBLIER SES MODIFICATIONS

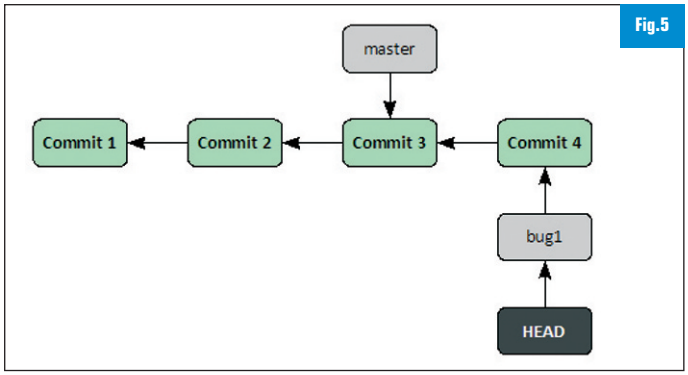
Une fois les modifications distantes récupérées, il peut être nécessaire de publier ses propres modifications sur le dépôt distant concerné. Cette fonctionnalité est dévolue à la commande git push. Sous conditions que le dépôt distant nous soit ouvert en écriture, la publication des modifications ne posera pas de problèmes si personne n'a réalisé de push sur ce dépôt distant dans l'intervalle depuis votre dernier fetch. Ceci est dû au fait qu'un dépôt ne peut réaliser seul que des merges de type "fast-forward". Dans le cas où il y aurait des conflits, il serait nécessaire de refaire un fetch puis une résolution des ces conflits sur votre dépôt local avant d'effectuer la publication des modifications sur le dépôt distant.

BRANCHES

Git tire une part importante de sa puissance de son modèle distribué, comme nous avons pu le voir jusqu'à présent. Néanmoins, il possède une fonctionnalité supplémentaire qui le différencie nettement des autres gestionnaires de version distribués : sa gestion des branches. Pour rappel, une branche permet de diverger de la ligne de développement principale (la branche master avec Git) pour continuer à travailler sans avoir à l'affecter avec des modifications pas forcément stables. Le modèle de branches de Git est incroyablement léger et surtout extrêmement rapide. En effet, là où les autres outils vont nécessiter la copie de l'ensemble du contenu du projet pour la création d'une branche, Git se contente de créer un simple fichier contenant l'identifiant SHA-1 du commit pointé par cette branche. Ainsi, par défaut seule la branche master existe et il s'agit de la branche courante de notre projet. A chaque commit, Git va se charger de faire avancer le pointeur de cette branche courante vers le nouveau commit réalisé. Le chaînage entre ces commits se faisant grâce à un pointeur contenu au sein d'un objet de type commit. Nommé parent, ce pointeur n'aura aucune valeur dans le cas du commit initial, il pointera vers un seul commit dans le cas normal et aura plusieurs parents dans le cas où le commit résulterait du merge entre plusieurs branches.

Pour notre projet, nous créons une branche censée résoudre un bug. Cette création se fait via la commande *git branch bug1*. La création est instantanée et l'utilisation de *git branch* liste ensuite nos 2 branches sur le dépôt local à savoir la branche master, marquée par une étoile car branche courante, et la branche bug1 nouvellement créée. Pour connaître à tout instant quelle est la branche courante, Git utilise la référence symbolique HEAD qui pointe dans le cas présent sur la branche master. Pour switcher sur notre branche bug1, il suffit d'exécuter la commande *git checkout bug1*. Celle-ci se charge de déplacer le pointeur HEAD sur la branche bug1 et va

Fig.5

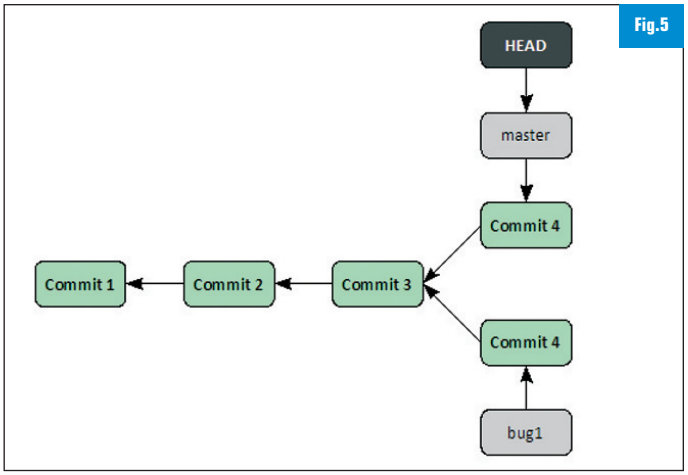


Historique des commits

ensuite remplacer au sein de votre répertoire de travail vos fichiers par les fichiers contenus dans le snapshot pointé par la branche bug1. Après avoir effectué quelques modifications permettant la résolution du bug, nous commitons les fichiers de correction sur la branche bug1, ce qui nous donne l'historique des commits illustré à la [Fig.5]. Supposons maintenant que nous rebasculions sur la branche master via un *git checkout master* et que nous effectuons un commit sur cette branche. Désormais, les branches master et bug1 ont un historique des commits qui diverge [Fig.6]. Le travail sur la branche bug1 terminé, il est temps de merger ses modifications dans la branche master. Avant de réaliser cette opération, nous pouvons utiliser la commande *git diff bug1* afin de vérifier les différences dans le contenu de la branche bug1 et de la branche courante master. Puisqu'il n'y a pas de différences introduisant des conflits, nous pouvons utiliser le merge automatique de Git en lançant la commande *git merge bug1 master* afin de merger la branche bug1 dans la branche master. Le travail sur la branche bug1 fusionné, nous la supprimons grâce à la commande *git branch -d bug1*. Il est bon de noter que dans le cas où le merge n'aurait pas été réalisable automatiquement par Git, les fichiers non mergés auraient listés en tant que tels dans le résultat de la commande git status et les zones non mergées dans les fichiers concernés annotées par des marqueurs spécifiques. Dans un workflow de travail classique, il arrivera fréquemment que l'on soit en train de travailler sur une branche spécifique et que d'un seul coup on doive corriger un bug par exemple. Le travail sur la branche courante n'étant pas terminé, il n'est pas possible de le commiter dans l'état et il ne serait pas convenable non plus d'intégrer la correction du bug au sein des modifications en cours. Pour ce genre de cas, Git propose le concept de branche temporaire ou de cache. Ainsi, avant de quitter la branche de travail courante il suffit de lancer un git stash pour que les modifications en cours soient stockées de manière temporaire dans un espace dédié par Git. Vous pouvez ensuite créer une nouvelle branche et basculer dessus, effectuer le travail nécessaire sur cette branche. Au retour sur la branche où vous étiez précédemment, il suffira d'utiliser *git stash apply* pour récupérer l'ensemble des modifications en cours que vous n'aviez pas commitées. Plutôt pratique !

TAGS

Comme la majorité des gestionnaires de version, Git met à disposition la possibilité de mettre des tags sur des états précis de l'historique des commits. Cela peut s'avérer utile pour marquer des versions précises d'un projet. Cela se réalise via la commande git tag *-a <nom du tag>*. Ceci aura pour effet de marquer le HEAD du dépôt local. Pour marquer un commit spécifique, il suffit d'utiliser la même commande en y ajoutant à la fin l'identifiant SHA-1 du commit concerné. Quant au partage d'un tag vers un dépôt distant, il est réalisé via la commande *git push <nom du dépôt distant> <nom du tag>*



Historique des commits avec divergence entre master et bug1

WORKFLOWS DE TRAVAIL FLEXIBLES

La puissance et la flexibilité apportées par Git changent le mode de travail du développeur au quotidien. Son modèle de branches léger et performant permet d'utiliser ces dernières pour mettre en place un workflow de travail local inimaginable avec les autres gestionnaires de version. Ce workflow est basé sur une utilisation intensive des branches puisque Git peut en gérer des centaines voire des milliers sans souci de performance. En local, 2 grandes possibilités sont couramment utilisées :

- La première consiste à posséder un certain nombre de branches récurrentes contenant du code regroupé par niveau de stabilité. Ainsi, on pourrait imaginer une branche *master* contenant uniquement du code stable permettant d'effectuer des livraisons, une branche *develop* utilisée pour tester le code avant de merger cela dans la branche master et enfin on pourra créer une branche *topic* qui contiendra les fichiers liés au développement de fonctionnalités ou à la correction de bugs. Cette dernière est par nature plutôt instable et elle sera mergée au sein de develop où la stabilité globale sera testée.
- La seconde consiste à posséder une branche courante master et à créer des branches à durée de vie limitée qui seront utilisées pour le développement de fonctionnalités ou de bugs. Le travail lié à une branche terminé, celle-ci se voit intégrée dans la branche master et peut ensuite être supprimée.

On le voit, Git permet de mieux isoler le travail via l'utilisation de branches, ce qui facilitera par la suite les retours en arrière éventuels ou les modifications introduites par le développement d'une fonctionnalité ou la correction d'un bug puisque celles-ci seront bien isolées.

ORGANISATION ENTRE DÉPÔTS DISTANTS

La flexibilité apportée par Git permet d'imaginer toutes les organisations possibles pour la collaboration entre dépôts distants. Parmi les innombrables topologies d'organisation entre dépôts distants imaginables, il se détache 3 grands modèles :

- Le premier modèle consiste à reproduire le modèle classique centralisé, tout en gardant la puissance du modèle distribué, en mettant un dépôt local Git partagé sur un serveur distant. Ce dépôt est légèrement différent des dépôts présents sur les postes clients et nécessite à sa création l'option –bare rajoutée à la commande *git init*. Ce modèle peut être adéquat pour des projets de petite ou de moyenne envergure.
- Le second consiste à avoir une personne en charge de l'intégration des modifications sur un dépôt serveur partagé. Dans ce modèle, chaque développeur va récupérer les modifications du dépôt partagé accessible seulement en lecture pour eux avant ensuite de publier les éventuelles modifications au sein de leur dépôt personnel public. Le responsable de l'intégration se chargeant ensuite de récupérer les modifications depuis les dépôts personnels publics des collaborateurs du projet sur son dépôt local avant de les intégrer ou non au dépôt partagé. Ce modèle est

celui adopté par la plateforme GitHub qui permet de stocker des projets open source.

- Le dernier apporte un niveau hiérarchique et est celui adopté par les projets de grande envergure ou un grand nombre de personnes collaborent. Au sein de ce modèle, on dénote un responsable général du projet en charge de publier les modifications sur le dépôt partagé dont il est le seul à avoir accès en écriture. En dessous de ce responsable se trouvent des sous-responsables en charge de sous-systèmes du projet. Ces derniers vont être en charge d'un certain nombre de développeurs qui viendront publier leurs modifications sur le dépôt local de ces sous-responsables lorsque leurs modifications seront terminées. A charge ensuite à ces derniers d'intégrer ou non ces modifications.

Les deux derniers modèles sont illustrés à la [Fig.7]. Bien entendu, de nombreux autres modèles restent possibles ce qui permet à chaque organisation de mettre en place un mode de collaboration distant adapté à ses besoins.

CONCEPTS AVANCÉS

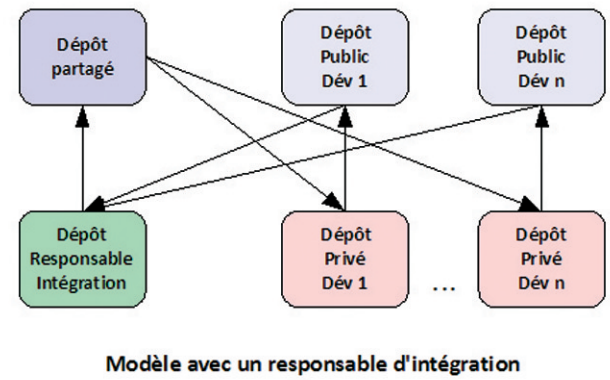
Cet article aura fait le tour des fonctionnalités essentielles dans le travail avec Git au quotidien. Pour des raisons de place, nous n'avons pu aborder un certain nombre de commandes moins connues mais pourtant fort utiles. Parmi celles-ci, on citera notamment la commande *git rebase* qui constitue une seconde voie pour réaliser le merge entre branches. Le rebase consiste à chercher le commit ancêtre commun entre les deux branches à fusionner afin de réappliquer les modifications introduites à partir de cet ancêtre commun sur l'une des deux branches. Autre commande notable, la commande *git cherry-pick* permettant de sélectionner un commit quelconque et de l'appliquer sur la branche courante. Ceci s'avère très utile pour appliquer le commit corrigeant un bug sur la branche de production sans y appliquer l'ensemble des autres modifications réalisées depuis sur la branche master. Enfin, lorsqu'un projet contient une régression et que l'on cherche à savoir de quel commit elle provient, il est approprié d'utiliser la puissante commande *git bisect*. Une fois localisé le dernier commit sain, la commande va réaliser une recherche dichotomique afin de retrouver le commit incriminé entre le dernier commit sain et le HEAD du projet. L'intérêt étant de fournir en entrée de commande un script shell effectuant les tests de non-régression et dont le retour permettra de savoir si oui ou non le contenu du commit testé est sain ou non.

CONCLUSION

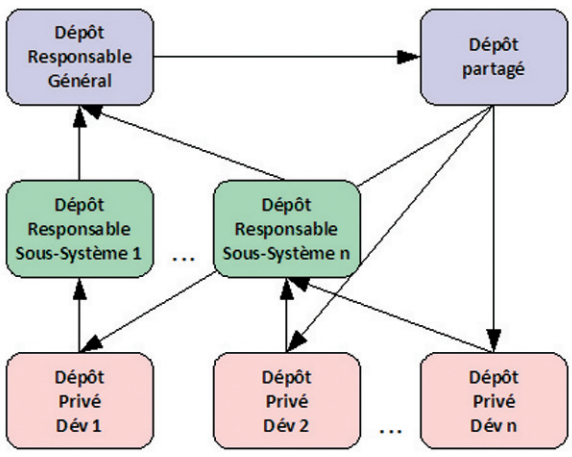
Dans le sillage des grands du monde de l'open source, Git connaît un bel essor et tend à s'imposer comme une solution d'avenir pour la gestion de version au sein des projets de développement informatique. Cet article aura mis en avant sa puissance et sa flexibilité en présentant ses concepts essentiels et en introduisant son utilisation sur des exemples concrets en faisant le tour de ses fonctionnalités principales. Bien que le nombre de commandes associées à Git soit finalement assez restreint, ce qui devrait permettre de faciliter son apprentissage, l'utilisation de Git n'est pas sans contraintes. En effet, si son utilisation de manière basique est très accessible il en coûtera un certain investissement avant de pouvoir tirer parti de toute sa puissance. Ce coût d'entrée pour utiliser pleinement Git risque peut être de ralentir son arrivée en entreprise où les dirigeants ne trouveront pas forcément un intérêt à basculer depuis SVN alors que les équipes sont déjà formées à l'outil. Cependant, pour les développeurs il paraît très important de s'investir dès aujourd'hui dans Git sur des projets personnels afin de ne pas rater le bon wagon car c'est sûr, l'avenir de la gestion de version appartient à Git !

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

Fig.7



Modèle avec un responsable d'intégration



Modèle avec niveaux hiérarchiques

Exemples de modèles d'organisation

L'animation web en HTML 5 et en 2D

La version 5 de HTML propose de nombreuses nouvelles fonctionnalités, dont le format devrait être validé à la norme W3C en 2014 [Fig.1].

La balise CANVAS

Elle est un des éléments HTML de la version 5 qui supporte les objets graphiques en script (comme Javascript), avec la possibilité de manipuler les objets images externes. La balise CANVAS est supportée par une large gamme des navigateurs tel que :

- Firefox 1.5 et +
- Opera 9 et +
- Safari
- Chrome
- Internet Explorer 9 et +

Les versions précédentes sont considérées comme dépassées et il est conseillé d'effectuer la migration vers une version plus récente pour bénéficier des avantages du langage.

Exemple basic

Au niveau de la balise CANVAS, celle-ci s'utilise de la façon suivante :

```
<html>
<body>
<canvas id="exemple" width="200" height="100"></canvas>

<script type="text/javascript">
var view=document.getElementById("exemple");
var obj=view.getContext("2d");
obj.fillStyle="#FF0000";
obj.fillRect(0,0,190,75);
obj.font="22pt Arial";
obj.fillStyle="white";
obj.fillText("Programmez",10,50);
</script>
</body></html>
```

L'ensemble du script montre la déclaration de la balise « canvas » avec un id, appelé 'exemple', qui fait appel à un script en javascript. L'utilisation du javascript va afficher dans une zone un rectangle entièrement rouge avec un texte écrit en blanc [Fig.2].

Comme le montre l'exemple, de nombreuses fonctionnalités peuvent être insérées dans la balise CANVAS car elle accepte différents langages, comme les lignes, points, cercles, carrés/rectangles, dont une partie sera détaillée dans les lignes qui suivent.

ANIMATION DÉTAILLÉE

L'animation en général et l'univers du jeu en particulier, ont toujours été un secteur qui a contribué à l'évolution des ordinateurs, en provenance aussi bien des amateurs, que des professionnels avec la 2D (2 dimensions). Bien sûr, la 3D a vu sa puissance s'accroître encore, grâce aux cartes graphiques. Cependant, la représentation technologique a toujours existé bien avant l'apparition du matériel, par le biais des calculs mathématiques, pour obtenir un rendu jouant principalement sur les couleurs et les ombres. Maintenant elle s'invite dans l'univers du web au côté de la 2D classique, avec de nombreuses fonctionnalités et des ouvertures vers l'avenir. Nous allons montrer une petite animation pour le web en 2D avec une apparence 3D, qui était réalisée dans les introductions et animations des années 80/90, ces effets étaient programmés en assembleur 68000, en langage C, sur des ordinateurs moins puissants (4/8 bits). Bien sûr, cette technologie est toujours utilisée dans des environnements portables.

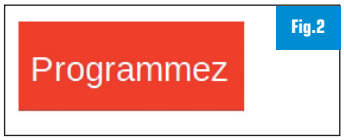
Fig.1

Share of Compatible Browser Versions

WebGL	21.33%	68.32%	77.58%	99.20%
Video	21.33%	68.32%	77.58%	99.20%
Audio	21.33%	93.07%	77.58%	99.20%
Geolocation	21.33%	93.07%	77.58%	99.20%
Canvas	21.33%	93.07%	77.58%	99.20%

Based on a sample of web traffic, October 2011

Fig.2



DÉCLARATION

L'élément de la balise « canvas » se décompose de la façon suivante :

```
<canvas width="320" height="200" id="animation" style="border
:solid 1px #000000;">
 Votre navigateur ne supporte pas la balise canvas. Merci d'en
choisir un autre
</canvas>
```

Cette balise se résume par une zone de dessin avec une largeur et une hauteur. Vous pouvez ajouter une feuille de style CSS sous la forme d'un cadre et un identifiant pour la repérer dans votre page HTML. Comme ici, vous obtenez une zone rectangulaire de 320 x 200 pixels (largeur x hauteur) avec un cadre noir de 1 pixel de large, illustré par l'image [Fig.2]. Pour insérer cet élément dans une page HTML, vous procédez de la façon suivante :

```
<html>
<head>
<title>Exemple animation</title>
</head>
<body>
<canvas width="320" height="200" id="animation" style="border:
solid 1px #000000;">
 Votre navigateur ne supporte pas la balise canvas. Merci d'en
 choisir un autre
</canvas>
</body>
</html>
```

Bien entendu, si la version de votre navigateur ne supporte pas le format HTML 5 et par conséquent la balise 'canvas', vous aurez un message de non compatibilité dans la zone pré-définie.

ANIMER UN OU PLUSIEURS OBJETS

Un objet graphique peut s'animer de différentes manières et principalement par le biais de calculs mathématiques. Comme le montrent les quelques captures d'écrans, avec un élément 'canvas', vous pouvez insérer des fonctions graphiques comme des images ou du

texte. L'article ne traitera pas l'ensemble des possibilités car le sujet est très vaste, mais vous verrez dans les lignes qui vont suivre :
• Un champ étoilé (starfield)
• Un défilement de texte (scroll-text)
• Une animation verticale d'un autre texte
• Deux rasters verticaux représentant un effet d'éclairage.
L'ensemble des possibilités présentées aujourd'hui a été inspiré de la démoscène des années 80/90 dont, a l'époque, les réalisations s'effectuaient sur les ordinateurs Atari et Amiga et faisaient la « UNE » des magazines [Fig.3, 4, 5, 6].

LE POINT DE DÉPART

Quel que soit le message que vous souhaitez faire passer dans une balise canvas, celle-ci se définit par l'appel d'un script javascript et se présente de la façon suivante :

```
<script type="text/javascript">
...
</script>
```

Pour commencer à dessiner à l'intérieur de la balise, vous devez initialiser la méthode DOM, appelé getContext. Comme cela, vous bénéficiez des fonctionnalités de dessins de base que propose le langage HTML, c'est-à-dire les cercles, les lignes, les carrés.
Cette méthode permet de bénéficier de la puissance 2D ou 3D suivant le format de déclaration. Cependant, l'article vous montrera que l'on peut réaliser de la 3D sans le définir.

```
var canvas = document.getElementById('animation');
var demo = canvas.getContext('2d');
```

LES EFFETS

Comme nous l'avons signalé quelques lignes plus haut, l'animation présente 4 effets, dont un en effet 3 dimensions.

Les Rasters

Les rasters sont définis comme une image en point par point pour obtenir une représentation graphique. La technique utilisée est une ligne avec une extrémité fixe et l'autre en mouvement. Par ailleurs, pour rendre un aperçu de mouvement et de relief 3D, plusieurs lignes seront représentées avec un petit décalage.

```
function rasters(nLigne, x, gauche, droit, sinusGauche,  sinus
Droit, effet)
{
 i = 0;
 while(i < nLigne)
 {
   demo.moveTo( x + (Math.sin(( (saut + (i * gauche)) / sinus
Gauche) * Math.PI) * effet),0);
   demo.lineTo( x + (Math.sin(( (saut + (i * droit)) / sinus
Droit) * Math.PI) ) , 200);
   i++;
 }
 demo.stroke();
}
```

Cet effet est placé dans une fonction pour précalculer le mouvement grâce aux fonctions moveTo ou lineTo. Le résultat obtenu sera affiché avec la fonction SCROBE.

Le logo

L'utilisation du mot « ATARI » est un clin d'oeil aux ordinateurs ST/STe/TT/Falcon. Ce titre aurait pu être une image bitmap, mais pour faciliter l'article, ce sera un texte. Pour mettre en place ce logo, vous devez définir :
• Le texte
• La police de caractères et sa taille
• Une couleur RGB
La police de caractères est un style de type classique pour le web car l'ensemble des ordinateurs la possèdent. La taille du texte se traduit par pixel vous permettant ainsi de choisir le nombre que vous souhaitez. La définition de la couleur se base sur les couleurs RGB (Rouge, Vert, bleu) et pour chaque couleur, vous pouvez définir une valeur entre 0 et 255. Enfin pour créer un effet de mouvement vertical, la fonction fillText positionne un texte avec les coordonnées X et Y. L'effet de mouvement sera appliqué sur la position Y, avec un petit calcul mathématique pour rendre un déplacement vertical aléatoire. La représentation du logo se décompose de la façon suivante :

```
var logo = "A T A R I";
demo.font = "40px Arial";
demo.fillStyle = "rgb(10,255,190)";
demo.fillText(logo,80, 110+Math.sin((saut / 50) * Math.PI) * 70);
```

Le ScrollText

Le scrollText correspond au défilement d'un texte. Il existe de nombreuses sortes d'effets autour de l'animation d'un texte, car celui-ci est souvent indicateur de messages ou de commentaires. Le mouvement du texte s'effectuera de gauche à droite et contiendra le message suivant :

```
var scrollText = "Réalisation par Christophe Villeneuve (Hello)
du groupe SECTOR ONE pour le magazine PROGRAMMEZ... ";
```

Ce message servira de délimiteur pour permettre de redémarrer au début, pour donner un effet de répétition à l'infini. La boucle d'animation du défilement du texte, sera caractérisée par un effet de vague, c'est-à-dire que chaque caractère du message est converti en un bloc image et sera animé séparément avec un calcul de position verticale moins important que celui du logo.
Bien sûr, quand le message est complètement passé, celui recommence depuis le début.

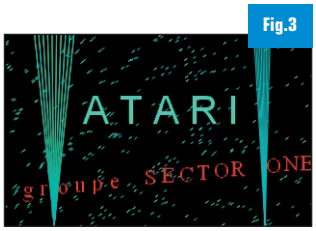


Fig.3

Fig.4

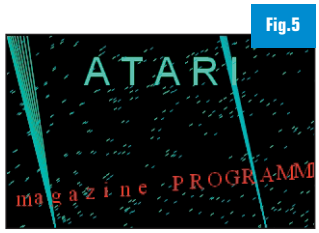


Fig.5

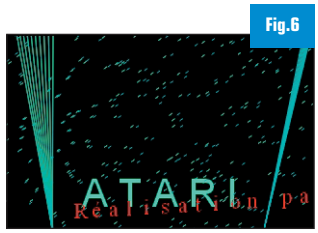


Fig.6

```
i = 0;
demo.font = "22px Courrier";
demo.fillStyle = "rgb(255,10,10)";
while(i < scrollText.length)
{
var left = (320 - (debut * 4)) + (i * 18);
demo.fillText(scrollText.charAt(i), left , 160 + (Math.sin
(((saut + i ) / 50) * Math.PI) * 30) );


if(i == scrollText.length-1 && left < 0)
 debut = 0;

 i++;
}
debut++;
```

Le champ d'étoiles

Le champ d'étoiles est un autre effet, appelé starfield, et représente un effet sous la forme d'une approche de perspective vers l'infini. Ainsi, le rendu permet d'avoir un aperçu de profondeur.

La technique présentée est une technique très répandue, car le principe consiste à mémoriser la position de points, et utilise les fonctions lineTo et MoveTo, comme cela, vous pouvez jouer sur la taille des points et des mouvements.

Il est important de définir une zone de délimitation (dimensions de la zone) pour éviter de provoquer des ralentissements de l'écran.

L'ensemble des points se détermine pour une ligne aléatoirement à partir du point de départ. Ces points seront répétés avec la fonction beginPath qui impose à l'animation de commencer une nouvelle ligne. Ainsi, l'effet de champ d'étoiles est représenté.

```
var stars=[];
function newStar()
{
 return {x: (Math.random() * 400) - 200, y: (Math.random() *
 200) - 100 };
}

i = 0;
while(i<200)
{
 stars.push(newStar());
 i++;
}

function renderStars()
{
 var newx
 var newy;
 i=0;
 demo['beginPath']();
 while(i<200)
 {
   stars[i].x += stars[i].x / 16;
   stars[i].y += stars[i].y / 30;
   newx = stars[i].x + 40;
   newy = stars[i].y + 90;
   demo.moveTo(newx, newy+2);
   demo.lineTo(newx + 3, newy);
  if (newx < 0 || newx > 320 || newy < 0 || newy > 200)
   stars[i] = newStar();
  i++;
  }
  demo.stroke();
}
```

LA BOUCLE

Lorsque l'ensemble des effets souhaités, ont été préparés, il est important de tout regrouper dans une boucle pour animer la totalité des effets. Pour cela, vous devez prévoir un rafraîchissement de l'écran, c'est-à-dire, le nettoyer de la zone de votre navigateur, pour éviter de laisser le texte ou les coordonnées précédentes de vos objets. L'utilisation des rasters sera gérée séparément, avec l'attribution d'une couleur différente et positionnée à chaque extrémité de la zone. Par ailleurs, vous ajoutez à la partie animation le logo et le scrollText. La gestion des étoiles est particulière, car il faut la faire séparément pour éviter les conflits au niveau des calculs et du rafraîchissement, c'est pourquoi, il suffit juste d'appeler l'animation sous la forme d'une fonction pour exécuter automatiquement ce champ d'étoiles. Enfin, il est important de répéter la boucle à l'infini en attribuant un intervalle moyen.

```
function boucle()
{
demo.fillRect(0,0,320,200);

demo.strokeStyle = '#bbbb00';
rasters(10,50,0.9,0.5,30,60,50);
demo.strokeStyle = '#00aaaa';
rasters(10,270,0.6,0.8,50,90,50);


// Ici animation du logo

// Ici animation du scrollText

demo.fillStyle = "rgb(0,0,0)";
renderStars();
saut++;
}

setInterval(boucle, 40);
```

CONCLUSION

Nous vous avons montré à travers cet article une animation 2D. Dans la réalité, celle-ci peut-être réutilisée pour afficher des messages, illustrer une partie d'un écran, et même prendre la place de zones réalisées en Flash ou Java.

L'animation présentée et le source sont disponibles directement sur mon site web: http://www.hello-design.fr

Si vous souhaitez découvrir un peu plus le milieu de l'animation, vous pouvez aller sur des sites comme : http://www.démoscene.fr ou http://www.pouet.net

■ Christophe **Villeneuve**
Consultant pour Alter Way solutions, auteur du livre « PHP & MySQL-MySQLi-PDO, Construisez votre application », aux Éditions ENI. Rédacteur pour nexen.net, membre des Teams DrupalFR, AFUP, LeMug.fr, PHPTV.

LIVRE DU MOIS

HTML 5

*Difficulté : ****
Editeur : Eyrolles
Auteur : Rodolphe Rimelé
Prix : 39 €

HTML5 est partout, même si la spécification définissant le langage est à peine figée et que les batteries de tests ne sont pas encore là et qu'il faudra plusieurs années avant que les outils, fournisseurs, sites s'y mettent. Malgré tout, c'est maintenant qu'il faut regarder HTML5 et ce qu'il apporte aux développeurs. Car il bouleverse beaucoup de choses, au-delà de la vidéo et des animations. Cet ouvrage est une véritable Bible du langage et aborde l'ensemble des éléments : les outils, le support des navigateurs, les attributs, formulaires, balises, canvas, la géolocalisation, File API, les données, DOM, les applications hors ligne, JavaScript. L'auteur propose de nombreux exemples et conseils pour mieux comprendre le fonctionnement de HTML 5. Notre coup de cœur du mois !

SAVOIR

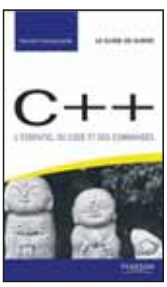
Initiation à la programmation avec Scheme

*Difficulté : ***
Editeur : éditions Technip
Auteur : Laurent Bloch
Prix : 35 €

Comment apprendre la programmation ? C'est une question que l'on se pose fréquemment. Java, C++, C# reviennent souvent mais finalement, sont-ils les meilleurs pour débuter dans la programmation. La facilité d'apprentissage est tout sauf un argument de ces langages. L'auteur va vous démontrer que Scheme est à la fois d'un accès facile et d'une puissance de programmation incroyable. Vous allez tout découvrir : les fondamentaux, les expressions, les tris, les types, la notion de récursif, la manipulation des données, les procédures, etc. Vous trouverez sans doute des éléments intéressants à appliquer à d'autres langages.

LANGAGE

C++ l'essentiel du code et des commandes 2e édition



*Difficulté : ****
Editeur : Pearson
Auteur : Vincent Gouvernelle
Prix : 23 €

Sans aide-mémoire que serait la programmation ? Cet ouvrage, pratique et de petit format, permet de rapidement retrouver une commande ou d'apprendre comment faire une variable ou comprendre l'héritage. Ce sont plus de 150 codes qui sont expliqués par l'auteur. De quoi éviter d'être bloqué trop longtemps…

TESTS

Gestion des tests logiciels

*Difficulté : ***
Editeur : Eni
Auteur : Emmanuel Itié
Prix : 45 €

Le test demeure la bête noire de beaucoup de développeurs, même si aujourd'hui, ils savent qu'ils sont incontournables. Mais au-delà du test, c'est toute sa gouvernance, sa gestion qu'il faut mettre en place. La gestion des tests est capitale en équipe, dans une SSII ou sur de gros projets pour définir, planifier, mettre en place les tests, les suivre. Le cycle de tests a besoin d'organisation et d'une automatisation toujours plus poussée. L'auteur décrit les bonnes pratiques, les outils, les méthodologies possibles. Indispensable !

LANGAGE

Apprenez à programmer en Python

*Difficulté : ****
Editeur : siteduzero.com
Auteur : Vincent Le Goff
Prix : 25 €

Python est sans doute un des meilleurs langages actuels : puissant, léger, d'un apprentissage rapide, il a tout pour plaire et pourtant, il demeure un langage sous-utilisé. Cet ouvrage est là pour vous faire découvrir un langage surdoué. Si la théorie est présente, le but est avant tout de passer aux travaux pratiques : exemples de code, exercices. Une bonne manière de bien débuter en Python.

SERVEUR

Glassfish 3

*Difficulté : ****
Editeur : Eni
Auteur : Jérôme Lafosse
Prix : 11 €

Le serveur Glassfish est l'implémentation officielle de JEE d'Oracle. Mais derrière ce nom de poisson se cache une architecture complexe qu'il faut comprendre et configurer. Ce livre a pour objectif de vous aider à comprendre l'organisation de la v3, les nouveautés, comment installer, configurer le domaine, déployer la sécurité, utiliser à la fois Java et PHP ou encore utiliser au quotidien la supervision et l'équilibre de charge. Bref, un livre à garder avec soi.

agenda \\

DECEMBRE

• Le 01 décembre 2011, Pavillon Cambon Capucines Paris, **Oracle Applications Day 2011**. Oracle vous invite à faire le point les dernières tendances du marché, la valeur ajoutée de la stratégie produit d'Oracle avec Oracle Fusion Applications, les applications d'Oracle nouvelle génération. http://www.oracle.com/webapps/events/ns/EventsDetail.jsp?p_eventId=133743

• Le 01 décembre 2011, Amphithéâtre de l'ESMA - Montpellier Aéroport, **Retour de MAX - Flex sur mobiles**. Michael Chaize, Flash Platform Evangelist chez Adobe, revient pour présenter la roadmap Flex ainsi que les toutes dernières nouveautés annoncées lors de l'événement Adobe MAX. http://flexmontpellier.groups.adobe.com/index.cfm?event=post.display&postid=38696

• Jusqu'au 6 décembre 2011, **Tour De France** des versions 17 de WINDEV, WEBDEV et WINDEV Mobile. http://www.pcsoft.fr/pcsoft/tdfcom/2011/index.html

• Le 03 décembre 2011, Cité Internationale Universitaire, PARIS, **Salon Studyrama des Formations Informatiques, Internet et Multimédia**. Afin de faire découvrir aux jeunes la richesse des cursus liés à ces formations ainsi que toutes les nouveautés en la matière. http://www.studyrama.com

• Le 06 décembre 2011, locaux de Microsoft France, 41 Quai Prés. Roosevelt, 92130 Issy-les-Moulineaux, **Agile Tour Paris 2011**. Venez découvrir les derniers usages autour de l'Agilité, des Tests et de la Qualité Logicielle, et profitez des présentations et des ateliers dédiés. https://msevents.microsoft.com/cui/EventDetail.aspx?culture=fr-FR&EventID=1032495354

• Du 12 au 13 décembre 2011, Centre de conférence Microsoft France, **Les Journées SQL Server, 1re édition.** Première conférence communautaire en Français autour du décisionnel et du moteur relationnel de Microsoft. https://msevents.microsoft.com/CUI/EventDetail.aspx?EventID=1032496260&Culture=fr-FR

KNOW formation

Allier le savoir-faire technique
au savoir-être relationnel,
pour le management et les équipes IT

Formation

pour dirigeants et équipes IT

✔ *Gérer le* **TEMPS**

✔ **MANAGER** *les projets*

✔ **COMMUNIQUER**

www.know-formation.com

CONTACT :

Stéphanie Khalif-Vennat : stephanie@know-formation.com - Tél. 01 74 70 48 91 - Fax 01 41 39 00 22

***Know-Formation** - Tour Albert 1er - 65, avenue de Colmar - 92500 Rueil Malmaison*

Soyez parmi les millions
d'applis téléchargées
chaque jour sur Nokia Store.

Rendez-vous sur **www.nokia.fr/developpeurs**

* Là où il faut être.

NOKIA
Connecting People