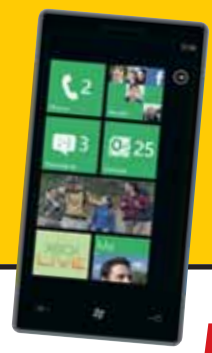
Tout savoir sur Windows Phone 7

Cahier spécial 48 pages



PROGRAMMEZ !

Le magazine du développement

PROgrammez !

www.programmez.com

mensuel n°134 - octobre 2010

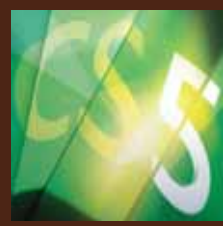
Géolocalisation
La killer app !

Google Maps
Bing Maps

La géolocalisation dans vos applications, vos smartphones et sites web

© Microsoft

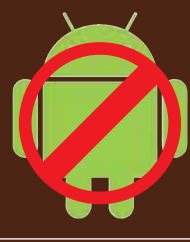
Webmaster



Dreamweaver CS5 à la sauce HTML 5

Oracle *contre* Google

Android utilise-t-il un vrai Java ?



Technique

Intégration continue : l'outil indispensable du développeur

IDE
Développement multiplateforme avec Titanium

CPU - GPU
Maîtrisez la puissance de OpenCL

Web
Transformer xHTML en document ODT

Projet
Mettre en place Sketchflow de Expression Blend

M 04319 - 134 - 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

WINDEV®

DÉVELOPPEZ 10 FOIS PLUS VITE

Un code unique :
Windows, Internet, Mobile
Java, .Net, PHP, J2EE, Mac,
Webservice, XML, Ajax,
Linux, Android, SaaS,...

ENVIRONNEMENT PROFESSIONNEL DE DÉVELOPPEMENT (AGL)

Créez 10 fois plus vite vos applications pour Windows, Internet et Mobile.

Votre code est compatible multi-cibles.

AGL N°1 en France

Demandez le Dossier + DVD gratuit

Tout est inclus
Gestion du cycle complet de développement
Support Technique gratuit

Elu «Langage le plus productif du marché»

VERSION EXPRESS GRATUITE
Téléchargez-la !

PC SOFT 25ème ANNÉE Au service des développeurs

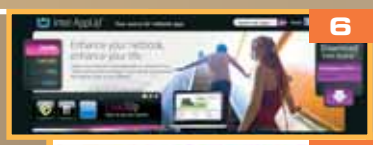
Fournisseur Officiel de la Préparation Olympique

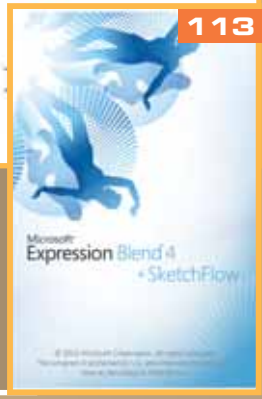
www.pcsoft.fr

Dossier gratuit 200 pages sur simple demande. Tél: **04.67.032.032** info@pcsoft.fr

PROgrammez! Le magazine du développement
numéro 134 - octobre 2010

sommaire\\





LES CONTRÔLES LES PLUS RAPIDES...

Chez Infragistics, nous nous assurons que nos contrôles pour .Net vous permettent de créer les meilleures interfaces utilisateur possible. C'est pourquoi nous avons testé et retesté nos Data Grids afin de nous assurer qu'elles sont les plus rapides sur le marché et que nos Data Charts surpassent tout ce que vous avez expérimenté. Utilisez nos contrôles et vous obtiendrez non seulement le temps de téléchargement le plus rapide mais aussi des applications qui sont toujours attrayantes. Rapide et belle... C' est une « Killer app ». Essayez les vous-même sur **infragistics.com/wow**.

Infragistics
KILLER APPS. NO EXCUSES.

Infragistics Ventes France N° Vert 0800 667 307
Infragistics Europe Ventes +44 (0) 800 298 9055
twitter.com/infragistics

éditO

Comment draguer un développeur en 10 lignes de code ?

C'est à n'y rien comprendre. Un jour, on nous dit, *« le développeur, c'est mort, il sert à rien »,* un autre, *« justement, on cherche des développeurs motivés »,* et encore un autre jour, *« un… quoi ? ».* Le monde informatique serait-t-il devenu schizophrène ?

Pas réellement. Mais à force d'entendre que le développeur est un boulet et qu'il ne rapporte rien, de tout vouloir externaliser, on en oublie le rôle du développeur. Oui effectivement, si c'est pour « pisser du code », son utilité est limitée ainsi que sa valeur ajoutée. Sauf que, le développeur actuel est source d'inspiration, source de proposition et surtout source de valeur ajoutée, qui peut faire la différence face à la concurrence.

Cette redécouverte de notre ami le développeur n'est pas nouvelle, enfin, pas pour tout le monde. Depuis l'avènement du Smartphone- oups de l'iPhone -, le développeur est devenu le centre du monde. Car le succès d'une plateforme mobile et de son AppStore dépend du développeur et des applications que l'on y trouve. Et ce n'est pas un hasard si aujourd'hui, tous les acteurs du Smartphone cherchent à attirer sur leur système les développeurs. Vous développez sur iPhone ? Pas grave, venez sur Windows Phone 7… etc. Le développeur est le nerf de la guerre de l'éditeur mobile. Et sans AppStore débordant d'applications, vous pouvez fermer boutique. Car ce supermarché des applications est à double détente : l'utilisateur veut y trouver son bonheur, et le développeur peut y développer son business, se faire connaître. C'est le cercle vertueux.

Microsoft l'a compris. Durant la conférence de rentrée de Microsoft France le 9 septembre dernier, le mot développeur est revenu des dizaines de fois. Ce martelage vise une seule chose : rassurer le socle actuel des développeurs Windows / .Net et attirer les développeurs mobiles iPhone ou Android. Car il faut bien alimenter la MarketPlace de Windows Phone 7. Et si Nokia ou RIM souhaitent réellement reprendre de la vigueur sur ce marché, il est impératif d'aller draguer le développeur. Car le prix du Smartphone n'est pas un argument suffisant. Et l'utilisateur cherche autre chose que le simple prix. Il veut du service, il veut des logiciels, il veut des jeux, avec une expérience utilisateur digne d'un iPhone. Voilà la véritable bataille du Smartphone aujourd'hui : le développeur. Et Apple a réagi, contraint d'assouplir les règles de développement des applications pour son AppStore. XCode et Objective-C ne sont plus les seuls outils acceptés… Le regard va vers Flash bien entendu mais aussi d'autres plateformes de programmation.

Si on se projette un peu plus loin, le développeur va devenir le nerf de la guerre du cloud computing, au moins sur la partie PaaS. Mais là, la situation est loin d'être claire car les plateformes se structurent à peine et le marché reste encore atone. Mais incontestablement, pour le développeur, le cloud représente le prochain eldorado.

■ François **Tonic**
Rédacteur en chef
ftonic@programmez.com

Le magazine du développement
PROgrammez!

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. Mazue, C.Soutou, O. Thery, S. Saurel, D. Catuhe. **Experts :** D. Mage, P. de Saint Steban, N. Bonaert, S. Warin, C. Villeneuve, G. Boissinot, A. Bompard, A. Petit, G. Rouchon, A. Quinaut, F. Guibert, M. Szablowski, G. N'toumi, Y. Ameur, G. Hutchins, P. Ognibene, D. Deraedt.
Illustration couverture : © Microsoft
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

Abonnement : Programmez 22, rue René Boulanger, 75472 Paris Cedex 10
Tél. : 01 55 56 70 55
abonnements.programmez @groupe-gli.com
Fax : 01 40 03 97 79 - 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

PROCHAIN NUMÉRO

N°135 novembre 2010
parution 30 octobre

✓ **Modélisation**
Simplifiez-vous (enfin) le développement

✓ **PHP**
Optimisation, déploiement, outils.
Le meilleur du monde PHP !

■ Oracle s'engage envers le projet OpenJDK afin de rassurer la communauté Java suite à la plainte déposée contre Google et son système Android. OpenJDK continuera à être une base de travail pour le code et Oracle travaillera avec lui. Le projet gardera une licence GPL.

■ Kraken est la nouvelle batterie de tests destinée à mesurer les performances du moteur javascript des navigateurs. Ce projet a été initié par Mozilla qui reproche à d'autres outils similaires (SunSpider notamment) de ne pas être assez complets et réalistes. Les critiques diront qu'il est trop lié à Mozilla… site : http://blog.mozilla.com/rob-sayre/2010/09/14/release-the-kraken/

■ Surprise, **Intel** ouvre son AppStore : AppUp. Cette boutique se dédie aux netbooks qui souffrent d'un manque de logiciels optimisés.
Ce centre propose des applications tant gratuites que payantes, pour le divertissement, les réseaux sociaux, les jeux et la bureautique, tous optimisés pour la mobilité et le format de l'écran d'un netbook. Pour inciter les particuliers à essayer de nouvelles applications, il propose des solutions d'essai avant l'achat effectif.
Site : http://www.appup.com/

IE 9 : Microsoft relance la guerre des navigateurs

Le 15 septembre 2010 marque une nouvelle étape dans la vie mouvementée d'Internet Explorer, le navigateur web de Microsoft. Alors que la concurrence fait basculer le marché, l'éditeur avait du mal à faire évoluer IE dont la v8 a déjà presque deux ans. Une éternité pour un navigateur. Et la présentation officielle de la bêta de IE 9 montre parfaitement l'enjeu du logiciel web pour l'éditeur : grande conférence aux Etats-Unis, soirée spéciale au Campus, le siège de Microsoft France près de Paris réunissant plus de 100 développeurs, designers. Si IE 8 avait suscité un accueil assez mitigé, cette v9 est bien mieux jugée. Microsoft rattrape son retard et propose quelques fonctions inédites sur un navigateur, ce qui ne manquera pas de forcer Chrome, Firefox et les autres, à réagir. Cette nouvelle version apporte de nombreuses nouveautés : accélération matérielle, CSS3, HTML5 (du moins les quelques fonctions disponibles), SVG, ICC, ECMAScript 5, DOM L2 et L3, format Open Formats… Bref, le meilleur des standards web actuel ! Pour les performances, IE9 mise sur deux cœurs optimisés : l'accélération matérielle et le multiprocess (au niveau onglet), sans oublier un tout nouveau moteur javascript (Chakra). L'accélération matérielle repose sur les API de Windows et les API DirectX (d'où la nécessité d'une mise à jour DirectX éventuellement). Reste à savoir comment le développeur peut en profiter efficacement sans que IE9 ne le fasse autoritairement. Et ensuite, il faudra se méfier des configurations matérielles et des GPU supportés. Par contre, par rapport à HTML 5, cette accélération est active sur l'élément canvas. Et on sait aujourd'hui que cette accélération concerne beaucoup d'éléments d'une page web : texte, image, SVG, fond de page, etc. L'accélération se décompose en trois phases : le rendu du contenu, la composition de la page puis la composition desktop (résultat final sur l'écran). Par défaut, l'accélération se veut globale. Les premiers tests montrent parfaitement la vélocité de IE et la bonne prise en charge de l'accélération, mais mieux vaut posséder un PC récent.
Les outils de développement ont eux aussi été revus et corrigés pour améliorer le debug de vos sites. L'interface a été épurée, tendance actuelle des navigateurs. La gestion des onglets a été considérablement refondue. Le multiprocess fait des merveilles notamment pour gérer, 10, 50 onglets. Et petit plus : on peut sortir un onglet dans une nouvelle fenêtre par simple glisser. Pratique. La gestion des add-ons a elle aussi été améliorée, tout comme, la recherche, largement facilitée.
Maintenant un gros travail attend le développeur web pour vérifier la compatibilité avec IE9, voire, comment optimiser certains éléments à ce navigateur. On attend maintenant son support par les outils de développement… Si quelques outils commencent à prendre en charge, imparfaitement HTML 5, il ne faut rien espérer de concret avant 2011.
D'autre part, Windows Phone 7 utilise une « simple » version 8 du navigateur, cependant, une mise à jour vers la v9 est sans aucun doute prévue en 2011, le navigateur mobile étant indépendant de l'OS. Et dommage que seul Windows soit supporté. A noter qu'une partie des sites IE9 officiels sont stockés sur Windows Azure. Microsoft veut du beau web ! A vous de le créer !

■ **Adobe** dévoile une version 64-bit de son Flash 10.2. Cette version s'avérait indispensable avec les systèmes 64 car les versions 32 posent de sérieux problèmes de stabilité et de performances, qui devraient être améliorées. Site : http://labs.adobe.com/downloads/flashplayer10.html

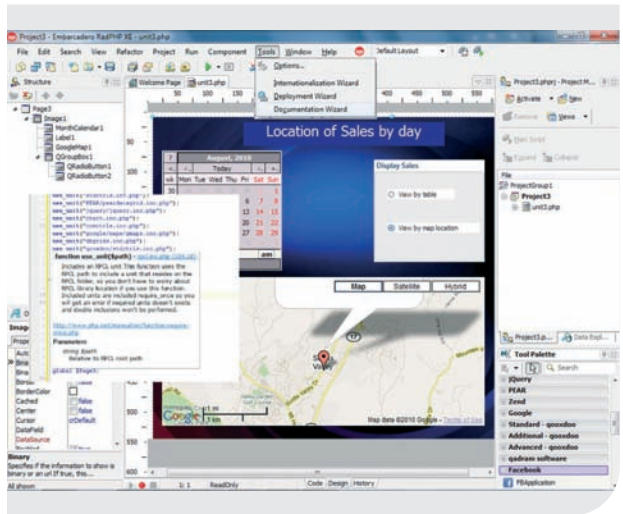
■ **Orange** veut concurrencer frontalement Apple et Google ! Pour l'opérateur, il s'agit de ne pas laisser iPad et autres Andoid seuls sur le marché et surtout de reprendre la main par rapport à ces constructeurs. Orange souhaite l'appui des grands opérateurs européens…

■ Le projet **Bespin** de Mozilla vient de changer de nom pour devenir Skywriter. Il s'agit d'un éditeur de code en ligne. Désormais, les développeurs veulent le transformer en éditeur HTML5. Ce mouvement s'accompagne d'un autre, plus important : disponibilité du code source sur la forge GiHub. Site : http://github.com/mozilla/skywriter

■ **Google** étend le nombre de licences open source disponibles pour sa forge : Code.google.com. Dorénavant, toutes les licences estampillées OSI sont acceptées par Code Google. C'est une bonne nouvelle pour les développeurs, même si Google s'éloigne de son idée de départ : limiter le nombre de licences ouvertes sur sa plateforme.

Embarcadero renouvelle les ex-IDE Borland

L'éditeur annonce la nouvelle version de sa suite de développement : RAD Studio XE incluant Delphi, C++ Builder, Delphi Prism et RadPHP XE, un nouveau venu. Cette version marque l'extension du nouveau label et des nouvelles fonctionnalités « XE » à la ligne d'outils de développement d'application d'Embarcadero, dont RAD Studio et chacun des outils indépendants de la suite. Les produits XE se distinguent par trois composants : le support pour les multiples types de base de données et d'environnement de déploiement, la technologie Embarcadero ToolCloud pour la gestion centralisée des licences et l'accès aux outils à la demande, et un protocole de mise à jour facilité pour All -Access XE.



Parmi les grandes nouveautés, retenons :

- Nouvelles fonctionnalités Cloud permettant aux développeurs d'interagir avec les environnements Cloud Computing. Les utilisateurs peuvent créer des applications ayant accès à la plateforme de services de stockage Microsoft Windows Azure (blob, queue, table). Les développeurs peuvent également déployer facilement des applications de serveur grâce à un simple « bouton » de déploiement sur les environnements Amazon Elastic Compute Cloud (EC2) et y accéder depuis de multiples types d'applications client léger.
- Environnement multi-tiers étendu pour une plus grande flexibilité de développement. L'environnement amélioré DataSnap offre aux développeurs : plus de possibilité pour créer des applications multi-tiers avec la capacité de bâtir des serveurs avec C++Builder et Delphi ; de nouveaux wizards pour créer des applications serveur et des modules client ; et le support pour JavaScript, REST et les protocoles de transport HTTP et HTTPS.
- Une nouvelle gamme d'outils intégrés pour le test, l'analyse des performances, l'automatisation et l'analyse de la qualité du code, qui permet aux développeurs de rationaliser leurs processus de développement et de passer moins de temps sur la gestion et davantage sur le codage. Les nouveaux outils incluent : le moteur de différence Beyond Compare de Scooter Software ; des outils d'analyse des performances et de débogage de mémoire/ressources avec AQtime de SmartBear Software ; l'identification intégrée du code source pour une introspection plus fine dans l'exécution du code avec Raize CodeSite ; et le développement d'application Internet avec les protocoles de communication IP avancés, avec la suite de composants /n Software IP*Works.
- Nouvelles fonctionnalités de contrôle de version avec l'intégration de Subversion, permettant aux développeurs de gérer plus facilement les modifications de code source pour leurs propres codes ou ceux de leur équipe, et fonctionne dans les IDE Delphi, C++Builder et Delphi Prism.

Une mise à jour avec un tarif spécial est possible pour les utilisateurs des éditions 2007-2010.

Site : www.embarcadero.com/products/application-development

agenda \\

OCTOBRE

• jusq'au 3 novembre, **Microsoft Days**. Aix - 6 et 7/10 : Issy, 12/10 : Lyon, 14/10 : Toulouse, 19/10 : Strasbourg, 21/10 : Lille, 3/11 : Nantes.

• Le 1er octobre 2010, Paris dernier jour de la 3e édition de l'**Open World Forum** http://www.openworldforum.org

• Le 8 octobre 2010, Nancy 2010, Locaux de l'ESIAL, **Agile Tour -- Nancy 2010** http://www.agiletour.org/

• Le 12 octobre 2010, Grande Arche La Défense : **Blackberry Innovation Forum.** http://www.blackberryinnovationforum.com

• Le 19 octobre, Bois Colombes, **Innovate 2010 Rational Software Conference** : conférence annuelle de IBM Rational sur le développement logiciel, l'agilité, la gestion de projets avec les solutions Rational. http://www.05.ibm.com/fr/events/innovate_2010/

NOVEMBRE

• Les 9 et 10 novembre 2010, Cité des Sciences de La Villette, **Forum PHP 2010.** Rasmus Lerdorf, créateur de PHP, sera l'invité d'honneur de cette édition anniversaire (15 ans). http://www.afup.org

• 25 novembre, Issy les Moulineaux, **MD Day 2010** , la journée du «Model Driven». www.mdday.fr

ETRANGER

• Du 23 au 27 octobre 2010, U.S.A. Los Angeles Convention Center. **Adobe MAX 2010** https://max.adobe.com/registration

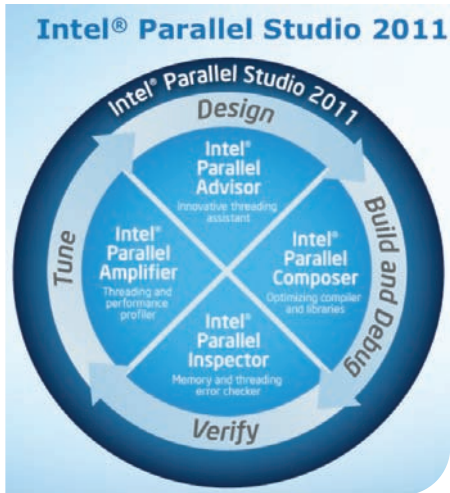
• Du 27 au 28 octobre 2010, Cambridge - Angleterre. **Embedded Linux Conference Europe 2010.** http://www.embeddedlinuxconference.com/elc_europe10/index.html

Intel dévoile Intel Parallel Studio 2011

C'est le 2 septembre dernier qu'Intel a dévoilé la nouvelle édition de sa gamme Parallel Studio 2011, dédiée aux développements parallèles en C et C++ avec Visual Studio. Cette version introduit de nombreuses nouveautés et deux nouveaux outils.

Tout d'abord, Intel Parallel Building Blocks qui permet de traiter en parallèle les données et les tâches, et le très attendu Parallel Advisor pour le design des applications parallèle. Il s'agit d'un outil de conseil et point d'entrée pour détecter et visualiser facilement là ou le code perd du temps et par où commencer pour introduire le parallélisme et répartir la charge de calcul sur les différents cores. D'autre part, le package studio inclut le support « premier » de l'éditeur avec support technique illimité et mise à jour valable un an. L'édition 2011 supporte Visual Studio, Intel TBB 3 et un compilateur amélioré sur les automatismes parallèles.

Intel met en avant désormais trois lignes d'outils : Parallel Studio, Compiler (édition professionnelle) et Cluster Toolkit. Un des credo de la version 2011 est l'intégration des outils entre eux, avec un cycle de vie de développement mieux intégré et un support élargi des versions de Visual Studio. L'autre grosse nouveauté est la disponibilité du langage Cilk Plus, langage pour simplifier le parallélisme des données et des tâches (C, C++ pour Windows et Linux). Intel Array Building Blocks fera aussi son apparition (en bêta). Il s'agit d'un nouveau Template C++ pour la parallélisation des données. Notons parmi les autres nouveautés de la v2011 :



- support de la vectorisation sur le SIMD pragma
- présence du GAP : parallélisation automatique « guidée »
- compatibilité Visual Studio 2010

Parallel Studio se compose désormais de Advisor, Composer, Inspector et Amplifier. Le package complet est vendu 799 $.

■ **Parasoft** annonce la sortie de SOAtest Oracle Fusion Edition. Cette version de SOAtest se dédie à la solution Oracle pour assurer le cycle de vie qualité avec des tests fonctionnels, des tests de montée en charge et de performance. Deuxième produit annoncé, l'environnement Concerto pour la gestion et le développement agile. Il supporte XP et Scrum. Il s'interface avec C, C++, Java, .Net, et permet de vérifier la qualité du code via des bonnes pratiques incluses par défaut.

■ **Plone 4.0** est disponible. Cette version introduit de nombreuses améliorations : performances en hausse, recherche plus efficace, nouvelle gestion des media et des gros fichiers, framework de formulaire dynamique, module de mise à jour simplifié, empreinte mémoire réduite.
Site : http://plone.fr

■ Perl n'est pas mort. Pour preuve, le projet **Dancer**, un framework en Perl. On peut utiliser les principaux standards du web (JSON, XML, REST). Son objectif est de concevoir des applications web en Perl sans forcément le connaître.
Site : http://perldancer.org/

■ **Rails 3.0** est là ! Très grosse nouveauté du monde web de septembre, Rails 3 était attendu avec impatience. Cette version apporte beaucoup de bonnes choses : nouveau moteur de requête dans le Active Record, modification du Action Controller utilisant REST, apparition du Action Mailer, protection contre les XSS, meilleure gestion des dépendances. Les plug-ins ont subi une sérieuse refonte et toute une partie du code interne a été réécrit.
Site : http://weblog.rubyonrails.org/

JavaOne 2010
Oracle veut rassurer la communauté

Après plusieurs mois de vives tensions suite à la plainte d'Oracle contre Google, l'événement du monde Java, JavaOne, se déroule alors que nous terminons ce numéro de Programmez !. Cette année, c'est une JavaOne inédite qui s'est déroulée, avec l'absence remarquée de James Gosling et de... Sun.

Tout d'abord, Java Standard Edition continue à supporter les nouveautés du monde Java et améliore, version après version ses performances sur les nouvelles architectures matérielles. Cette édition supporte les langages de script, améliore la productivité du développeur, notamment avec Netbeans. OpenJDK continuera à être le socle de Java open source et Oracle travaille toujours avec, sur les versions 7 et 8 de la JDK. Ces versions devraient arriver en 2011 et 2012. Décidément, Java 7 n'en finit plus d'être retardé. Avant JavaOne, deux plans étaient évoqués pour Java 7 : sortie en 2012 (plan A) ou JDK 7 en 2011 et JDK 8 fin 2012. Désormais, c'est le plan B qui semble avoir la corde. Ce qui oblige à redéfinir l'implémentation de certains JSP et projets Java. Ainsi, le projet Coin doit déterminer les changements (mineurs) du langage, initialement prévu dans JDK 7, mais qui pourrait finalement n'intervenir totalement que dans la v8.

Dans la v7, on devrait donc garder une partie des lambdas dans le langage, Jigsaw et partiellement Coin. La v8 reprendrait la base de la JDK 7 en l'étendant et en rajoutant de nouveaux éléments... Jigsaw était le projet de modularisation de JDK 7. Bref, malgré les éclaircissements, l'avenir de Java reste nuageux...

M. Kurian, nouvelle tête pensante Java, a dévoilé les travaux d'Oracle sur les différentes JVM maison (JRockit, HotSpot). Elles travailleront avec Oracle JRockit Mission Control pour diagnostiquer les causes des problèmes de temps de réponse, de performances d'une application Java. Oracle a aussi dévoilé une partie RIA Java avec de notables améliorations de JavaFX avec HTML 5, CSS, Javascript qui seront supportés nativement. Par contre sur la partie mobile, un certain flottement apparaît. Alors que JavaFX était la stratégie offensive de Sun sur mobile, il semble y avoir un changement technique, en remettant Java ME sur les rails, en lieu et place de JavaFX Mobile.

Netbeans a eu droit à sa minute de gloire avec une roadmap sur 2 ans ! La prochaine version sera la v6.10, attendue pour janvier 2011, puis la v7 (juin 2011) et enfin la v7.1 (septembre 2011). Cette dernière fera la jonction avec 2012 avec une mise à jour sans doute courant janvier 2012. La 6.10 renforcera le développement Java EE, PHP, supporta la base de données Oracle et annonce des performances en hausse... Glassfish a lui aussi été présent à la JavaOne. La version 3.1 est attendue pour fin 2010. Et le serveur sera certifié JRockit. Sur GlassFish 4.0, il a été confirmé qu'il sera bien open source. Plus de détails le mois prochain.

Java en quelques chiffres :

1,1 milliard de desktops ayant Java d'installé
930 millions de JRE téléchargés chaque année
1,4 milliard de Java Cards fabriquées chaque année

Dreamweaver CS5 parle HTML 5

Lorsque l'on parle du « HTML 5 », on désigne le plus souvent une série de fonctionnalités relatives aux nouveaux standards du web qui vont au-delà de la nouvelle spécification du langage HTML…

…certains parlent de "HTML 5 and friends" pour recouvrir l'ensemble de ces technologies. Il s'agit principalement de :

- la révision du langage HTML qui succèdera à l'HTML 4.01 / XHTML 1.1. Elle remplace le XHTML 2.0 qui ne verra finalement jamais le jour, suite à des divergences d'opinions au sein du W3C qui ont amené à la création du WHATWG.
- les API Javascript associées aux nouvelles API du DOM définies par le HTML 5
- la version 3 du langage Cascading Style Sheet (CSS3)
- d'autres spécifications qui ont été un temps proposées pour le HTML 5 mais qui ont finalement été rejetées, laissant aux navigateurs le choix de leur implémentation.

Toutes ces spécifications HTML, Javascript et CSS sont actuellement en cours de rédaction. Il existe deux rédactions concurrentes de ces futurs standards du web: celle du W3C et celle du WHATWG, toutes deux à l'état de « brouillon » (Draft state). Ian Hickson, le rédacteur principal des spécifications HTML 5, estime que la prochaine étape de rédaction pour le W3C (Candidate Recommandation) sera atteinte en 2012 alors que l'on attend le stade final de recommandation officielle pour 2022.

Ainsi, si de plus en plus de navigateurs implémentent aujourd'hui des fonctionnalités dites "HTML 5", il faudra encore longtemps avant que la plupart de celles-ci arrivent à un taux de pénétration acceptable. En effet, Microsoft Internet Explorer, encore largement majoritaire chez les internautes, ne commencera à implémenter ces nouveautés que dans sa version 9, actuellement en cours de développement.

De plus, ce taux de pénétration doit s'apprécier fonctionnalité par fonctionnalité car chaque éditeur de navigateur fait son marché dans les spécifications HTML 5 et y ajoute les siennes, avec une fâcheuse tendance pour certains de présenter le tout aux utilisateurs sous le terme générique "HTML 5". Et quand elles sont présentes sur plusieurs navigateurs, ces fonctionnalités sont implémentées de manière très disparate. Il est donc légitime d'avoir des doutes quant à l'interopérabilité actuelle et à venir de ces contenus.

Pour toutes ces raisons, Dreamweaver CS5 en tant que tel ne présente pas de fonctionnalité relative au HTML 5, outre la présence du nouveau doctype.

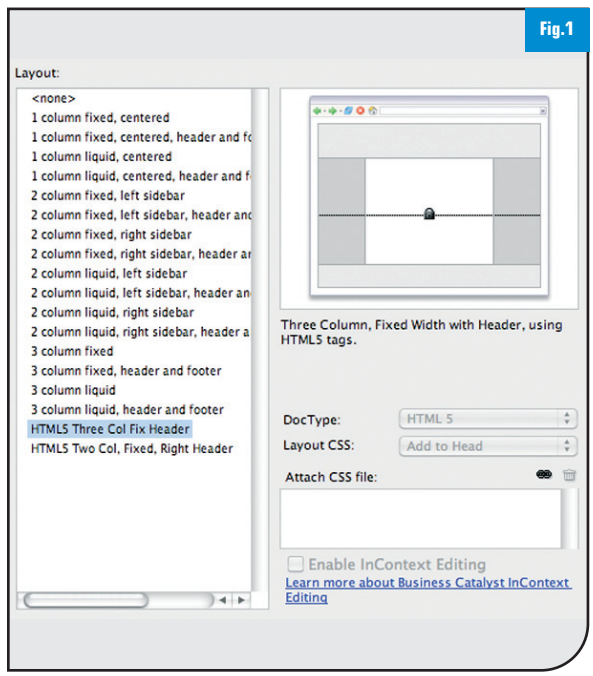
```
<!DOCTYPE HTML>
```

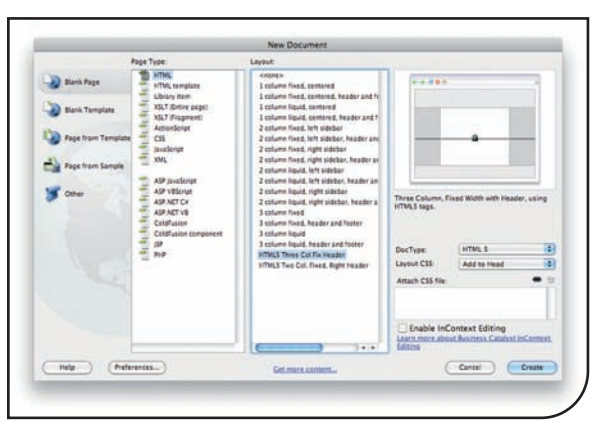
Note : La nouvelle spécification du doctype, qui a l'avantage de grandement simplifier la syntaxe précédente, est à la fois stable et rétro-compatible. Il n'y a donc aucune raison de ne pas l'utiliser dès aujourd'hui.

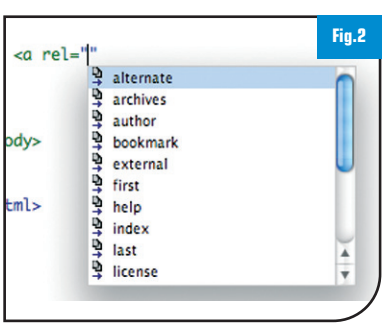
Cependant, malgré ces réserves, il ne fait aucun doute que ces spécifications s'imposeront tôt ou tard et constituent une évolution majeure et nécessaire du web tel que nous le connaissons. Les standards s'adaptent aux usages réels du web, et c'est une excellente chose.

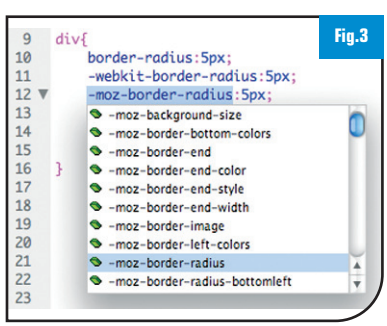
D'autre part, bon nombre de spécifications sont suffisamment stables pour pouvoir être testées dans de bonnes conditions, et dans la plupart des cas le fait que le navigateur ne sache pas correctement les interpréter ne lui posera pas pour autant de problème pour afficher le reste de la page.

Pour anticiper l'évolution de ce standard, Adobe a annoncé la disponibilité immédiate d'une mise à jour pour



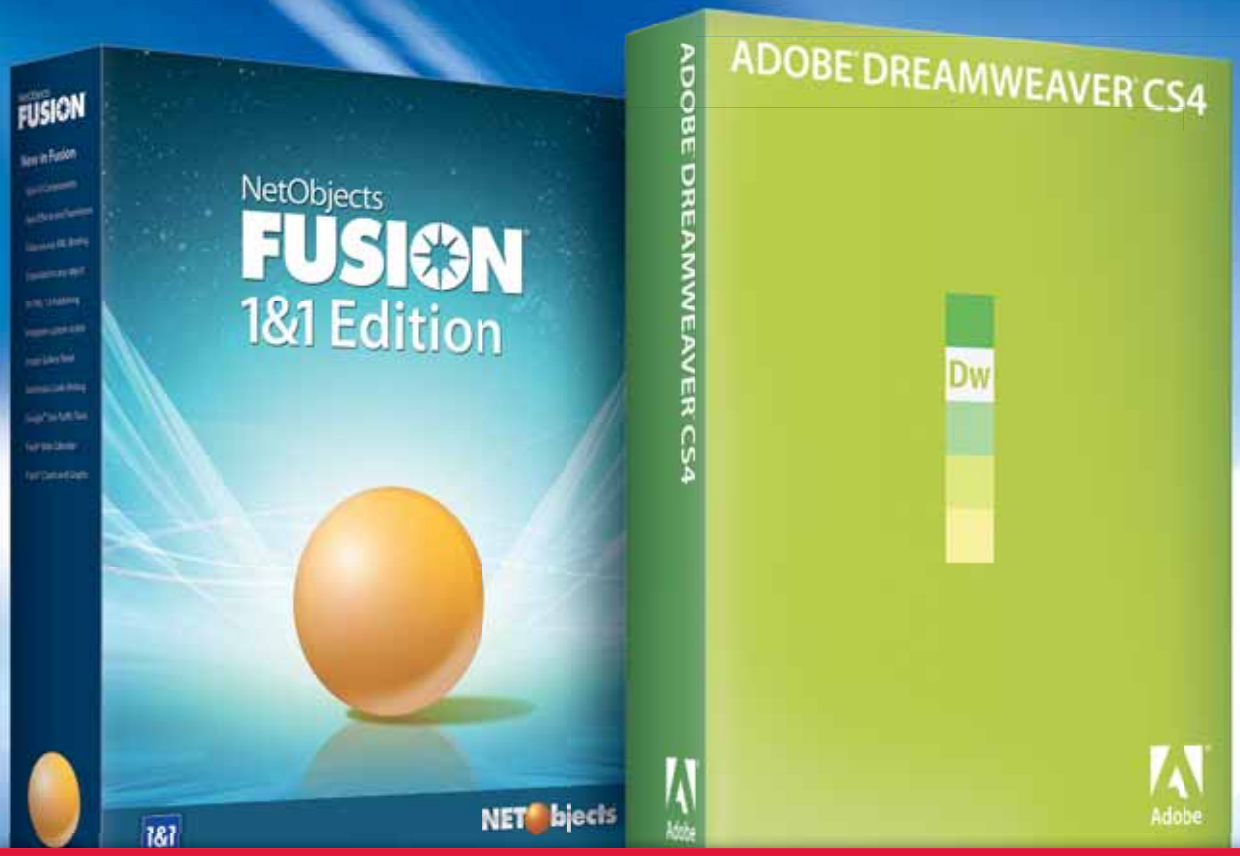






Les nouveaux packs hébergement 1&1 ré

VOTRE SITE DE



LOGICIELS GRATUITS

INCLUS DANS LES NOUVEAUX PACKS HÉBERGEMENT 1&1*

NetObjects Fusion® 1&1 Edition est un logiciel de création qui vous permet de réaliser votre site ou de l'adapter afin qu'il puisse être consulté depuis n'importe quel appareil mobile. Cette version optimisée de NetObjects Fusion® 11 vous offre des modèles de mise en page et des designs exclusifs spécialement adaptés au Web mobile.

Référence absolue en matière de conception de site Internet, **Adobe® Dreamweaver® CS4** vous apporte des fonctionnalités avancées pour prévisualiser et tester vos pages dans un environnement mobile. Grâce à Device Central, vous adaptez le code HTML et les applications flash de votre site à tous types de terminaux mobiles.

* Logiciels offerts pour toute nouvelle commande d'un pack hébergement (voir pages suivantes) et disponibles en téléchargement à partir de l'Espace Client 1&1.

MEMBRE DE united internet

0970 808 911 Appel non surtaxé

volutionnent votre présence Web

VIENT MOBILE

De plus en plus d'utilisateurs consultent le Web à partir de leurs appareils mobiles, mais de nombreux sites ne sont pas encore adaptés à l'affichage sur écrans de petite taille.

Désormais, nous vous fournissons les logiciels pour convertir votre site au Web mobile.

- ✓ **Votre site compatible avec les navigateurs mobiles**
- ✓ **Interface intuitive et simple d'utilisation**
- ✓ **Modèles de mise en page exclusifs**

1&1

www.1and1.fr

Dreamweaver CS5 (11.0.3), gratuitement téléchargeable à partir du site web Adobe (http://www.adobe.com/support/dreamweaver/downloads_updaters.html) et intégrant un support de l'HTML 5.

DREAMWEAVER CS5 ET HTML 5

Courant mai 2010, Adobe avait soumis à la communauté des utilisateurs, le « pack d'extension HTML 5 pour Dreamweaver CS5 » via le site Adobe labs. La mise à jour Dreamweaver CS5 11.0.3 reprend désormais ces fonctionnalités en natif dans l'outil. Notons qu'il existe également un pack d'extension HTML 5 disponible sur Adobe Exchange à destination des versions CS3 et CS4 de Dreamweaver, mais celui-ci se limite à l'ajout de l'auto-complétion des nouvelles balises HTML.

Les pages de démarrage HTML 5

Les pages de démarrage de Dreamweaver sont des pages HTML et des feuilles de style CSS prêtes à l'emploi avec une structure répondant à des besoins classiques, et dont le formatage et la syntaxe respectent les bonnes pratiques. Elles sont par ailleurs abondamment commentées afin de permettre à chacun de se familiariser avec un certain nombre de concepts de base ou avancés du web design [Fig.1].

Le pack d'extension ajoute deux pages de démarrage HTML 5: l'une avec deux colonnes, l'autre avec trois colonnes. On y retrouve notamment l'utilisation de nouveaux éléments sémantiques tels que <nav>, destiné à accueillir des éléments de navigation, <section> qui définit un groupe de contenus liés par un même thème, <header> et <footer> pour contenir le haut et bas de page respectivement, et d'autres comme <aside>, ou <article>.

L'auto-complétion HTML 5 et CSS3

Les balises HTML 5 dont les spécifications semblent suffisamment avancées ont été ajoutées à l'auto-complétion. Au delà du simple gain de productivité, cette aide nous évite d'avoir à connaître par coeur la syntaxe exacte de ces éléments encore peu connus de la plupart des développeurs. En plus des balises sémantiques mentionnées précédemment, on trouvera par exemple la balise <canvas>, la balise <video> et la balise <audio>. Les attributs liés à ces nouvelles balises, les nouveaux attributs des balises existantes et nouvelles valeurs d'attributs ont également été ajoutés. On retrouvera notamment les nombreuses nouvelles relations de liens définies [Fig.2].

De la même manière, le pack HTML ajoute l'auto-complétion pour un certain nombre de nouveaux styles CS33, tels que les transformations et les transitions.

Puisque l'implémentation de ces fonctionnalités n'est pas encore finalisée par les navigateurs actuels, il est nécessaire à ce stade d'utiliser des préfixes spécifiques pour pouvoir en bénéficier. Heureusement le pack HTML 5 ajoute l'auto-complétion à la fois des syntaxes W3C et de ces syntaxes propriétaires [Fig.3].

Le support des éléments Canvas et Video dans l'affichage en direct

Les éléments canvas et video sont correctement pris en charge par l'affichage en direct de Dreamweaver. Rappelons que ce mode utilise le moteur de rendu open source Webkit afin de disposer d'un affichage représentatif de la page dans l'environnement du logiciel.

En mode de division code / design, il est donc possible de voir du code javascript s'exécuter en live dans un canvas à droite de son éditeur de code, ce qui s'avère bien plus productif que d'avoir à systématiquement passer de l'éditeur au navigateur [Fig.4]. En ce qui concerne les vidéos insérées par la balise du même nom, notons que si leur affichage est effectivement possible dans l'affichage en direct, il n'en va pas de même des contrôles vidéos associés (boutons pause, lecture, volume, etc.). En effet, ces contrôles sont entièrement implémentés par les navigateurs, et il faudra donc exécuter notre page dans chacun d'entre eux pour constater leur présence et leur bon fonctionnement.

La gestion des medias queries via le panneau multiscreen

Les medias queries permettent de spécifier au navigateur quelle feuille de style utiliser dans tel ou tel contexte. Elles sont notamment utilisées afin d'appliquer des feuilles de styles particulières selon les résolutions d'écran, et donc selon que notre page est vue sur un smartphone, sur un ordina-

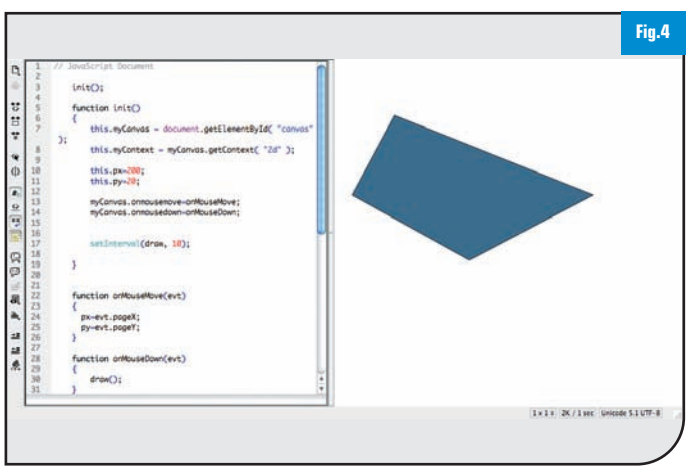


Fig.4

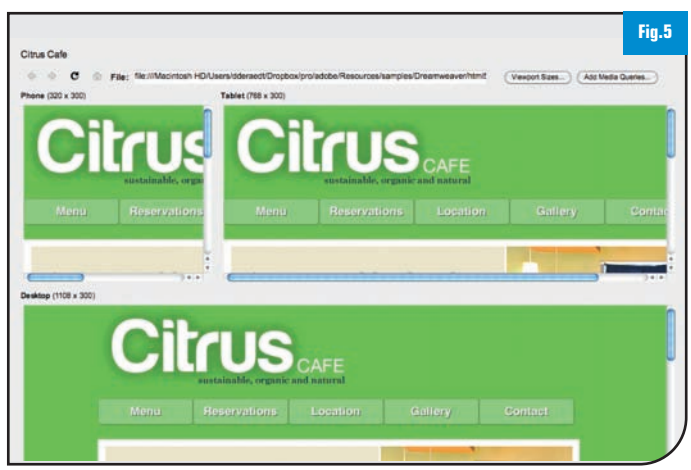


Fig.5

CHOISISSEZ VOTRE PACK HÉBERGEMENT

À MOITIÉ PRIX

Non seulement 1&1 vous fournit gratuitement les meilleurs logiciels pour rendre votre site mobile, mais en plus vous bénéficiez de -50 % pendant 6 mois sur nos nouveaux packs d'hébergement !

6 MOIS À -50 %
LOGICIEL OFFERT !*

1&1 PACK CONFORT

- 2 domaines au choix INCLUS
- NOUVEAU : 100 Go d'espace disque
- Trafic ILLIMITÉ
- 100 comptes email
- 5 bases de données MySQL (100 Mo)
- Outils de création de site : éditeurs Web, blog, album photo, e-Boutique Start
- PHP5, PHP6 (bêta), Perl, Python, Ruby, C, tâches cron
- Outils de communication : formulaire, chat
- 1&1 Référencement
- 1&1 WebStat
- NOUVEAU : Logiciel offert* NetObjects Fusion® 1&1 Edition

~~4,99 €~~ HT/mois (5,97 € TTC/mois)

2,49 € HT/mois (2,98 € TTC/mois)*

1&1 PACK PRO

- 3 domaines au choix INCLUS
- NOUVEAU : 250 Go d'espace disque
- Trafic ILLIMITÉ
- 500 comptes email
- 20 bases de données MySQL (100 Mo)
- Outils de création de site : éditeurs Web, blog, album photo, e-Boutique Start
- PHP5, PHP6 (bêta), Perl, Python, Ruby, C, tâches cron
- Outils de communication : formulaire, chat, newsletter, RSS, listes de discussion
- 1&1 Référencement
- 1&1 WebStat
- Google Adwords® : 50 € offerts
- NOUVEAU : Logiciel offert* NetObjects Fusion® 1&1 Edition ou Adobe® Dreamweaver® CS4 (au choix)

~~9,99 €~~ HT/mois (11,95 € TTC/mois)

4,99 € HT/mois (5,97 € TTC/mois)*

1&1 PACK PREMIUM

- 4 domaines au choix INCLUS
- NOUVEAU : 500 Go d'espace disque
- Trafic ILLIMITÉ
- 1000 comptes email
- 50 bases de données MySQL (100 Mo)
- Outils de création de site : éditeurs Web, blog, album photo, e-Boutique Start
- PHP5, PHP6 (bêta), Perl, Python, Ruby, C, tâches cron
- Outils de communication : formulaire, chat, newsletter, RSS, listes de discussion
- 1&1 Référencement
- 1&1 WebStat
- Google Adwords® : 75 € offerts
- Certificat SSL dédié INCLUS
- NOUVEAU : Logiciel offert* NetObjects Fusion® 1&1 Edition ou Adobe® Dreamweaver® CS4 (au choix)

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

9,99 € HT/mois (11,95 € TTC/mois)*

Votre domaine à prix sensationnel :
le .fr à 4,99 € HT/an (5,97 € TTC/an),
le .eu à 0,99 € HT/an (1,18 € TTC/an)* !

* Offre « 6 mois à -50 % » soumise à un engagement de 12 mois. Frais de mise en service : 5,97 € TTC (Pack Confort) ou 11,95 € TTC (Pack Pro, Pack Premium). A l'issue des 6 premiers mois, les produits concernés sont aux prix habituels (Pack Confort à partir de 5,97 € TTC/mois, Pack Pro à partir de 11,95 € TTC/mois, Pack Premium à partir de 23,91 € TTC/mois). Logiciel offert pour toute nouvelle commande d'un pack hébergement et disponible en téléchargement à partir de l'Espace Client 1&1. Offre domaine applicable la première année uniquement au lieu du prix habituel de 6,99 € HT/an (8,36 € TTC). Conditions détaillées sur www.1and1.fr. Offres sans engagement également disponibles.

MEMBRE DE united internet

Appel non surtaxé
0970 808 911

www.1and1.fr

1&1

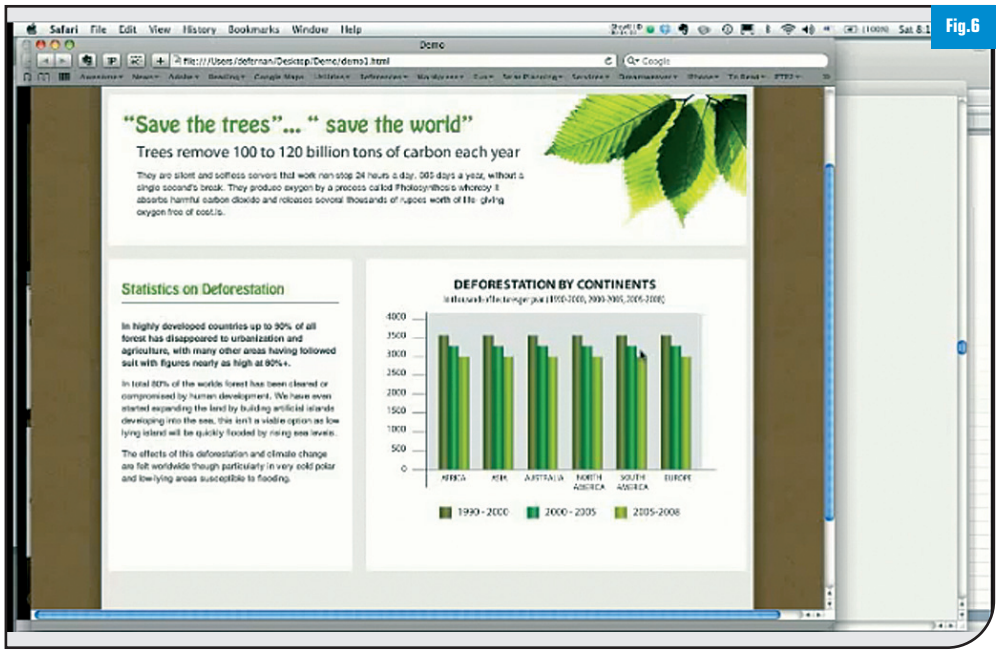


Fig.6

teur de bureau, ou encore sur une tablette. Le problème que va typiquement rencontrer le webdesigner est qu'il va lui falloir tester ces différentes résolutions depuis l'écran de son poste de travail. Le *multi-screen panel* répond à ce besoin en permettant d'afficher côte à côte trois rendus Webkit de tailles différentes (paramétrables) afin de pouvoir tester directement les feuilles de style associées à chaque résolution [Fig.5].

LES FUTURS OUTILS

Le pack d'extension HTML 5 n'est qu'une première étape dans le processus d'adaptation des outils de webdesign à l'évolution des standards du web. En l'état, des limites importantes existent: certaines balises HTML 5 et propriétés CSS3 ne sont pas reconnues par l'aide de code, l'auto-complétion Javascript ne connaît pas encore les nouvelles API, l'affichage des vidéos pourrait être amélioré, et j'en passe…

Mais, au delà de ces limites relativement mineures, il va falloir penser et concevoir une nouvelle génération d'outils capables de tirer parti des nouvelles possibilités offertes par ces nouveaux standards.

Les pages web de demain seront très différentes de celles que l'on connaît aujourd'hui, et les méthodes de travail doivent s'y adapter. Adobe s'est récemment engagé à mettre à la disposition des utilisateurs des outils de premier ordre pour créer du contenu HTML 5. Si ces futurs outils n'ont pas encore été dévoilés publiquement, certaines expérimentations montrées lors de ces derniers mois peuvent donner une idée de ce qui se prépare en coulisse. Lors d'Adobe MAX 2009 à Los Angeles, un ingénieur a montré une version expérimentale de Dreamweaver dans laquelle il est possible, via une fonction « smart paste », d'importer des animations créées dans Flash Professional et des images illustrator et de les afficher dans un canvas HTML, via une simple bibliothèque Javascript [Fig.6].

Au Google I/O de mai 2010, Kevin Lynch (le CTO d'Adobe) a fait la démonstration d'un nouvel outil WYSIWYG entièrement dédié à la création d'animation avec les CSS3. L'objectif de ce proof of concept était de montrer à quoi pourrait ressembler le processus de création de bannières et de publicités entièrement réalisées en HTML.

Nous nous trouvons indéniablement dans une période de transition. Même si le chemin est encore long avant que les technologies dont nous avons parlé fassent partie du quotidien des utilisateurs, il est important pour les développeurs et les webdesigners d'anticiper ce virage suffisamment tôt.

Voici quelques liens (encadré ci-contre) qui vous permettront d'aller plus loin dans la découverte des futurs standards du web.

■ David **Deraedt**
Consultant web chez Adobe

Télécharger le pack d'extension HTML 5 pour Dreamweaver CS5

http://www.adobe.com/support/dreamweaver/downloads_updaters.html

Tableaux de correspondance des fonctionnalités HTML 5 pour chaque navigateur
http://caniuse.com/

Une excellente introduction au HTML 5
http://diveintohtml5.org/

Une présentation des technologies HTML 5 en action
http://slides.html5rocks.com

Les spécifications HTML 5 du W3C
http://dev.w3.org/html5/spec/Overview.html

Spread for Windows Forms | à partir de € 767

GrapeCity

Feuille de calcul complète pour applications Windows Forms.

- 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
- Nouveau - outil intégré de création de diagrammes avec 85 styles
- Nouveau - préserve les .XLS et restaure les fonctions non supportées
- Inclut des apparences prédéfinies ainsi que la possibilité de créer des apparences personnalisées

DXperience Enterprise | à partir de € 998

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

FusionCharts | à partir de € 153

InfoSoft Global

Graphiques Flash & JavaScript (HTML5) interactifs pour les applications Web.

- Animez vos applications Web avec des graphiques interactifs et pilotés par les données
- Créez des graphiques AJAX avec des possibilités d'exploration en quelques minutes
- Exportez les graphiques au PDF et les données en CSV directement depuis les graphiques
- Créez des jauges, des tableaux de bord, des graphique financiers et plus de 550 types de carte
- Adopté par plus de 16 000 clients et 330 000 utilisateurs dans 110 pays

Resco MobileForms Toolkit | à partir de € 576

resco.net

Plus de 20 contrôles et bibliothèques Windows Mobile pour NET Compact Framework.

- Inclut image, list, tree, chart, detailView, grid, zip, notes, calendrier Outlook, CustomKeyboard pour les écrans tactiles et bien plus encore
- Environnement de développement unique et totalement intégré avec Visual Studio
- Inclut des thèmes de composants accessibles directement depuis le concepteur Visual Studio
- Nouveaux contrôles incluant Resco ScrollBar, Resco ProgressBar et Resco MaskedTextBox

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

Récursivité avec Oracle 11gR2

« Chi va piano, va sano, e va lontano », il aura fallu attendre 11 ans avant qu'Oracle n'implémente le concept de récursivité, prévu par la norme SQL en 1999. Microsoft et IBM l'avaient quelque peu devancé sur ce point (2005 pour SQL Server).

Depuis la release 2 de la version 11g, l'opérateur WITH permet de programmer la récursivité. Une sous-requête peut désormais utiliser la requête principale. Ce type d'écriture est plus concis et plus efficace que la bonne vielle clause CONNECT BY que vous pouvez désormais ranger dans le tiroir.
La structure d'une telle requête est la suivante. Chaque résultat d'une sous-requête est appelé CTE *(Common Table Expression)*. La norme SQL avait introduit la directive RECURSIVE afin de différencier une CTE récursive d'une autre non récursive. Aucun des éditeurs n'a adopté ce mot-clé et c'est au travers de la requête qu'on peut statuer à propos de la récursivité.

```
WITH
nomTableTemporaire (col1, col2...) AS
   (SELECT d'initialisation
      UNION ALL
    SELECT de récursion)
SELECT résultat
```

La première des deux sous-requêtes est dite anchor member. La seconde est appelée recursive member. La première ne doit pas référencer la requête principale tandis que la seconde doit impérativement la référencer mais une seule fois. La première sous- requête peut être elle-même composée de requêtes reliées par des opérateurs ensemblistes (UNION ALL, UNION, INTERSECT ou MINUS). Par ailleurs, vous devrez utiliser UNION ALL entre la requête anchor member et la requête recursive member.

SYNTAXE

La syntaxe Oracle de l'opérateur WITH permettant la récursivité est la suivante :

```
WITH
  nomRequete1 ([alias_col1 [,alias_col2]...])
AS
  (sousRequete1)
  [SEARCH { DEPTH FIRST BY alias_c1 [,alias_c2]...
            [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ]
          | BREADTH FIRST BY alias_c1 [,alias_c2]...
            [ ASC | DESC ] [ NULLS FIRST | NULLS LAST ] }
        SET col_ordre ]
  [CYCLE alias_c1 [,alias_c2]...
    SET alias_col_cycle TO valeur_cycle
        DEFAULT valeur_non_cycle ]
  [,nomRequete2 ([alias_col1 [,alias_col2]...])
AS
 (sousRequete2) …
```

Premier exemple

Considérons l'exemple suivant décrivant la hiérarchie de quelques aéroports. Dans l'exemple la récursivité va parcourir les liaisons entre enregistrements fils et parents (Castelnaudary dépend de Toulouse qui dépend d'Orly, lui-même sous Charles de Gaulle).

Données à parcourir

```
SQL> SELECT OACI, nomAero, OACI_resp FROM Aeroport;

OACI  NOMAERO                        OACI_RESP
----- ------------------------------ ----------
LFPG  Paris Charles de Gaulle
LFPO  Paris Orly                     LFPG
LFBO  Toulouse Blagnac               LFBD
LFBD  Bordeaux Merignac              LFPO
LFCI  ALbi                           LFBO
LFCK  Castres                        LFBO
LFMW  Castelnaudary                  LFBO
LFMT  Montpellier Fregorgues         LFMM
LFMM  Marseille Marignane            LFPO
```

Le nombre d'alias de colonnes d'une requête principale doit être identique au nombre d'alias de colonnes des requêtes anchor member et recursive member. La requête recursive member ne doit pas contenir DISTINCT, GROUP BY, MODEL, fonctions d'agrégat, sous-requêtes ou jointures externe avec la requête principale.

La figure 2 décrit le parcours de l'arbre des aéroports récursivement (la condition de liaison est basée sur l'égalité des colonnes clés. Dans l'exemple, le point de départ est l'aéroport d'Orly à partir duquel Oracle recherche tous ses subordonnés, pour chaque subordonné, Oracle recherche tous ses subordonnés, etc.

Parcours de l'arbre [Fig.2]

Requête

```
COL arbo FORMAT A15
WITH
  sous_Paris_Orly (OACI, OACI_resp, niveau)
AS ( SELECT OACI, OACI_resp, 0 niveau
     FROM Aeroport
     WHERE OACI = 'LFPO'
   UNION ALL
   SELECT a.OACI, a.OACI_resp, niveau+1
     FROM sous_Paris_Orly sp, Aeroport a
     WHERE sp.OACI = a.OACI_resp )
SELECT OACI_resp,
   LPAD(' ',2*niveau)||OACI arbo, niveau
   FROM sous_Paris_Orly
   WHERE niveau>0
   ORDER BY niveau, OACI;
```

Résultat

```
OACI_RESP  ARBO            NIVEAU
---------- --------------- ----------
LFPO         LFBD                1
LFPO         LFMM                1
LFBD           LFBO              2
LFMM           LFMT              2
LFBO             LFCI            3
LFBO             LFCK            3
LFBO             LFMW            3
```

CONSTITUER UNE LISTE DES DESCENDANTS

La figure 3 illustre le parcours de l'arbre en partant de l'aéroport d'Orly. A chaque subordonné trouvé, la liste des descendants est complétée.

Liste des descendants [Fig.3]

Requête

```
COL OACI FORMAT A5
WITH
  sous_Paris_Orly
  (OACI, OACI_resp, niveau, aro_liste)
AS ( SELECT OACI, OACI_resp, 0 niveau,
       CAST(OACI_resp AS VARCHAR2(25))
     FROM Aeroport
     WHERE OACI = 'LFPO'
   UNION ALL
   SELECT a.OACI, a.OACI_resp, niveau+1,
       CAST(aro_liste || ',' ||
         a.OACI_resp AS VARCHAR2(25))
     FROM sous_Paris_Orly sp, Aeroport a
     WHERE sp.OACI = a.OACI_resp )
SELECT OACI, niveau, aro_liste
    FROM sous_Paris_Orly
    ORDER BY niveau, OACI;
```

Résultat

```
OACI      NIVEAU ARO_LISTE
----- ---------- --------------------
LFPO           0 LFPG
LFBD           1 LFPG,LFPO
LFMM           1 LFPG,LFPO
LFBO           2 LFPG,LFPO,LFBD
LFMT           2 LFPG,LFPO,LFMM
LFCI           3 LFPG,LFPO,LFBD,LFBO
LFCK           3 LFPG,LFPO,LFBD,LFBO
LFMW           3 LFPG,LFPO,LFBD,LFBO
```

ORDONNER LES DESCENDANTS

La clause SEARCH permet d'ordonnancer les lignes extraites lors du parcours. L'option BREADTH FIRST BY retourne les lignes d'un même niveau (sibling rows) avant de descendre dans l'arbre (child rows). L'option DEPTH FIRST BY réalise l'inverse. L'ordre est indiqué par la colonne citée dans l'opérateur BY et remonte au niveau de la requête principale à l'aide d'une colonne fictive présente dans l'opérateur SET. La figure 4 illustre le parcours de l'arbre en profondeur d'abord puis en largeur en partant de l'aéroport d'Orly.

Parcours en profondeur puis en largeur [Fig.4]

Requête

```
COL OACI FORMAT A5
WITH
 sous_Paris_Orly (OACI, OACI_resp, niveau)
AS
   ( SELECT OACI, OACI_resp, 0 niveau
      FROM Aeroport
      WHERE OACI = 'LFPO'
    UNION ALL
    SELECT a.OACI, a.OACI_resp, niveau+1
      FROM sous_Paris_Orly sp, Aeroport a
      WHERE sp.OACI = a.OACI_resp )
SEARCH DEPTH FIRST BY OACI_resp SET order1
SELECT OACI_resp,
    LPAD(' ',2*niveau)||OACI arbo, niveau
FROM sous_Paris_Orly
WHERE niveau>0
ORDER BY order1;
```

Résultat

```
OACI_RESP  ARBO                NIVEAU
---------- ---------- ----------
LFPO       LFBD                 1
LFBD         LFBO               2
LFBO           LFCI             3
LFBO           LFCK             3
LFBO           LFMW             3
LFPO       LFMM                 1
LFMM         LFMT               2
```

La figure 5 décrit le parcours de l'arbre en largeur d'abord puis en profondeur en partant de l'aéroport d'Orly. Cela ne représente pas, dans notre exemple, l'arbre modélisé mais cela peut résoudre certaines problématiques.

Parcours en largeur puis en profondeur [Fig.5]

Requête

```
COL OACI FORMAT A5
WITH
 sous_Paris_Orly (OACI, OACI_resp, niveau)
AS
   (SELECT OACI, OACI_resp, 0 niveau
      FROM Aeroport
      WHERE OACI = 'LFPO'
    UNION ALL
    SELECT a.OACI, a.OACI_resp, niveau+1
      FROM sous_Paris_Orly sp, Aeroport a
      WHERE sp.OACI = a.OACI_resp )
SEARCH BREADTH FIRST BY OACI_resp SET order1
SELECT OACI_resp,
    LPAD(' ',2*niveau)||OACI arbo, niveau
FROM sous_Paris_Orly
WHERE niveau>0
ORDER BY order1;
```

Résultat

```
OACI_RESP  ARBO                NIVEAU
---------- ---------- ----------
LFPO       LFMM                 1
LFPO       LFBD                 1
LFBD         LFBO               2
LFMM         LFMT               2
LFBO           LFMW             3
LFBO           LFCK             3
LFBO           LFCI             3
```

CYCLES DE COLONNES

La clause CYCLE permet de détecter les cycles de colonnes, une ligne compose *in cycle* lorsque l'une de ses lignes ancêtre (ascendant) a la même valeur pour une colonne donnée (celle du cycle cherché). Attention, il ne s'agit pas de cycle de l'arbre, qui s'il existe devra être considéré comme un graphe (voir la section qui suit).

```
CYCLE alias_c1 [,alias_c2]...
 SET alias_col_cycle TO valeur_cycle DEFAULT valeur_non_cycle
```

- Les alias suivant la clause CYCLE doivent faire référence aux colonnes de la requête principale.
- Les paramètres valeur_cycle et valeur_non_cycle doivent être des caractères de taille 1.
- Dès qu'un cycle est détecté, alors la colonne alias_col_cycle est initialisée à la valeur valeur_cycle. La récursivité s'arrête sur cette ligne (aucune ligne descendante n'est examinée) mais le traitement se poursuit sur les lignes de même niveau (et leurs descendantes).
- Si aucun cycle n'est trouvé, la colonne alias_col_cycle contiendra la valeur valeur_non_cycle pour les lignes extraites. Cette colonne est d'ailleurs automatiquement ajoutée à la requête finale.

Ajoutons à l'exemple précédent la colonne année de création (figure 6) et cherchons les aéroports qui ont un ancêtre construit la même année.

Nouvelles données [Fig.6]

Contenu de la table

```
SQL> SELECT OACI, nomAero, OACI_resp, anneeCreation FROM Aeroport;

OACI       NOMAERO                        OACI_RESP  ANNEECREATION
---------- ------------------------------ ---------- -------------
LFPG       Paris Charles de Gaulle                            1978
LFPO       Paris Orly                     LFPG                1967
LFBO       Toulouse Blagnac               LFBD                1968
LFBD       Bordeaux Merignac              LFPO                1972
LFCI       ALbi                           LFBO                1967
LFCK       Castres                        LFBO                1980
LFMW       Castelnaudary                  LFBO                1981
LFMT       Montpellier Fregorgues         LFMM                1973
LFMM       Marseille Marignane            LFPO                1975
```

La figure 7 présente le parcours récursif de l'arbre et la détection d'un cycle sur l'année de création entre Albi et Orly.

Détection d'un cycle [Fig.7]

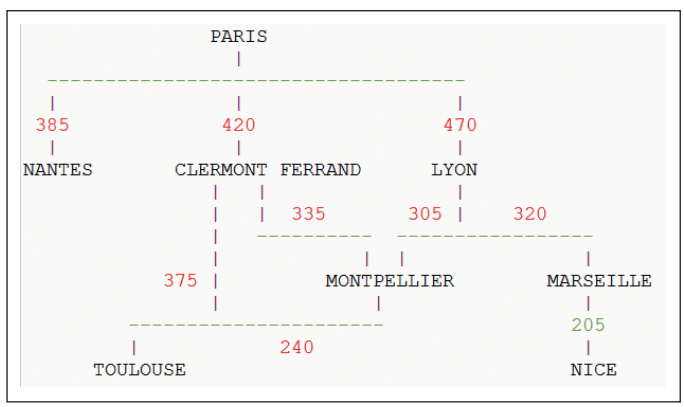
Requête et résultats

```
COL arbo FORMAT A15
COL est_cycle FORMAT A9
WITH
 sous_Paris_Orly (OACI, OACI_resp, niveau, creation)
AS
  (SELECT OACI, OACI_resp, 0 niveau, anneeCreation
     FROM Aeroport WHERE OACI = 'LFPO'
   UNION ALL
   SELECT a.OACI, a.OACI_resp, niveau+1, anneeCreation
     FROM sous_Paris_Orly sp, Aeroport a
     WHERE sp.OACI = a.OACI_resp )
SEARCH DEPTH FIRST BY OACI_resp SET order1
CYCLE creation SET est_cycle TO 'Y'  DEFAULT 'N'
SELECT OACI_resp, LPAD(' ',2*niveau)||OACI arbo,
     niveau, creation, est_cycle
FROM sous_Paris_Orly ORDER BY order1;

OACI_RESP  ARBO                NIVEAU   CREATION EST_CYCLE
---------- --------------- ---------- ---------- ---------
LFPG       LFPO                     0       1967 N
LFPO         LFBD                   1       1972 N
LFBD           LFBO                 2       1968 N
LFBO             LFCI               3       1967 Y
LFBO             LFCK               3       1980 N
LFBO             LFMW               3       1981 N
LFPO         LFMM                   1       1975 N
LFMM           LFMT                 2       1973 N
```

PARCOURS D'UN GRAPHE NON ORIENTÉ

L'exemple du graphe non orienté suivant est extrait du blog de Frédéric Brouard, expert SQL et spécialiste de SQL Server (http://blog.developpez.com/sqlpro).

Graphe non orienté [Fig.8]



La table Autoroutes (figure 9) permet d'implémenter cet état de fait. Nous allons progressivement rechercher le détail des trajets possible entre la capitale et le Capitole.

Table Autoroutes [Fig.9]

Contenu de la table

```
SQL> SELECT ville_de, ville_vers, km FROM Autoroutes;

VILLE_DE        VILLE_VERS          KM
--------------- --------------- ----------
PARIS           NANTES             385
PARIS           CLERMONT-FERRAND   420
PARIS           LYON               470
CLERMONT-FERRAND MONTPELLIER       335
CLERMONT-FERRAND TOULOUSE          375
LYON            MONTPELLIER        305
LYON            MARSEILLE          320
MONTPELLIER     TOULOUSE           240
MARSEILLE       NICE               205
```

La figure 10 décrit le parcours récursif permettant d'extraire le nombre d'étapes des différents trajets entre Paris et la ville rose. On y retrouve la jointure entre la table de référence et celle construite récursivement.

Nombre d'étapes entre Paris et Toulouse [Fig.10]

Requête

```
WITH trajets (ville_vers, etape)
AS
   (SELECT DISTINCT ville_de, 0
      FROM  Autoroutes
      WHERE  ville_de = 'PARIS'
    UNION  ALL
    SELECT  a.ville_vers, t.etape + 1
      FROM  Autoroutes a, trajets t
      WHERE t.ville_vers  = a.ville_de)
SELECT ville_vers, etape
FROM   trajets
WHERE  ville_vers = 'TOULOUSE';
```

Résultat

```
VILLE_VERS          ETAPE
--------------- ----------
TOULOUSE                 2
TOULOUSE                 3
TOULOUSE                 3
```

La requête de la figure 11 ajoute à la précédente la somme des kilométrages pour chaque ligne extraite du graphe.

Nombre d'étapes et distances entre Paris et Toulouse [Fig.11]

Requête

```
WITH trajets
   (ville_vers, etape, distance)
AS
   (SELECT DISTINCT ville_de, 0, 0
      FROM  Autoroutes
      WHERE ville_de = 'PARIS'
    UNION ALL
    SELECT  a.ville_vers,
       t.etape + 1, t.distance+a.km
      FROM  Autoroutes a, trajets t
      WHERE t.ville_vers  = a.ville_de)
SELECT ville_vers, etape, distance
FROM   trajets
WHERE  ville_vers = 'TOULOUSE';
```

Résultat

```
VILLE_VERS          ETAPE   DISTANCE
--------------- ---------- ----------
TOULOUSE                 2        795
TOULOUSE                 3       1015
TOULOUSE                 3        995
```

La requête de la figure 12 ajoute à la précédente la construction progressive des chemins parcourus (que j'estime à 50 caractères) et l'ordonnancement en fonction de la distance totale.

Chemins entre Paris et Toulouse [Fig.12]

Requête et résultats

```
WITH trajets (ville_vers, etape, distance, trajet)
AS
   (SELECT DISTINCT ville_de, 0, 0, CAST('PARIS' AS VARCHAR(50))
      FROM Autoroutes WHERE  ville_de = 'PARIS'
    UNION  ALL
    SELECT  a.ville_vers, t.etape + 1, t.distance+a.km,
         CAST(t.trajet|| ',' || a.ville_vers AS VARCHAR2(50))
      FROM  Autoroutes a, trajets t
      WHERE t.ville_vers  = a.ville_de)
SELECT trajet, etape, distance
      FROM trajets WHERE  ville_vers = 'TOULOUSE' ORDER BY distance;

TRAJET                                                  ETAPE   DISTANCE
------------------------------------------------------- ---------- ----------
PARIS,CLERMONT-FERRAND,TOULOUSE                              2        795
PARIS,CLERMONT-FERRAND,MONTPELLIER,TOULOUSE                  3        995
PARIS,LYON,MONTPELLIER,TOULOUSE                              3       1015
```

Qu'adviendrait-il si toutes les étapes inverses étaient également stockées dans la table Autoroutes ? Toutes les requêtes perdraient les pédales dans les cycles.

Oracle renvoyant l'erreur « ORA-32044: cycle détecté lors de l'exécution de l'interrogation WITH récursive ». DB2 retournant « SQL0347W The recursive common table expression may contain an infinite loop. SQLSTATE=01605 ». SQL-Server « The statement terminated. The maximum recursion 100 has been exhausted before statement completion».

Nous allons voir comment éviter ces désagréments.

PARCOURS D'UN GRAPHE ORIENTÉ

La requête de la figure 13 ajoute les étapes inverses à la table implémentant les trajets possibles.

Ajout des chemins inverses [Fig.13]

Requête et résultats

```
SQL> INSERT INTO Autoroutes
           SELECT ville_vers,ville_de, km FROM Autoroutes;

SQL> SELECT ville_de, ville_vers, km
           FROM Autoroutes ORDER BY ville_de, ville_vers;

VILLE_DE        VILLE_VERS          KM
--------------- --------------- ----------
CLERMONT-FERRAND MONTPELLIER       335
CLERMONT-FERRAND PARIS             420
CLERMONT-FERRAND TOULOUSE          375
LYON            MARSEILLE          320
LYON            MONTPELLIER        305
LYON            PARIS              470
MARSEILLE       LYON               320
MARSEILLE       NICE               205
MONTPELLIER     CLERMONT-FERRAND   335
...
```

Les requêtes précédentes ne conviennent désormais plus à cette table car Oracle détecte des cycles sans fin.

Il est possible de se débarrasser des cycles en comparant tout chemin courant avec la colonne évaluée en question (avec NOT LIKE '%...' par exemple). Ce n'est toutefois pas suffisant, car Oracle ne vous fait pas confiance et teste la requête avant d'exécuter cette condition. En conséquence, il faut ajouter une condition concernant un nombre maximal d'itérations récursives.

La requête de la figure 14 teste la ville d'arrivée avec tout chemin construit récursivement et élimine ainsi les cycles. Par ailleurs, la recherche dans le graphe se limite à 10 niveaux. Une nouvelle route apparaît (la dernière), puisque Montpellier est située sur le chemin de Clermont et de Lyon. Comme dirait F. Brouard, *« c'est la plus longue, mais peut-être la plus belle »*.

Parcours d'un graphe en évitant les cycles [Fig.14]

Requête et résultats

```
WITH trajets (ville_vers, etape, distance, trajet)
AS
   (SELECT DISTINCT ville_de, 0, 0, CAST('PARIS' AS VARCHAR(50))
      FROM  Autoroutes WHERE  ville_de = 'PARIS'
    UNION  ALL
    SELECT  a.ville_vers, t.etape + 1, t.distance+a.km,
         CAST(t.trajet|| ',' || a.ville_vers AS VARCHAR2(50))
      FROM  Autoroutes a, trajets t
      WHERE t.ville_vers  = a.ville_de
      AND   t.trajet NOT LIKE '%' || a.ville_vers || '%'
      AND   etape <10)
SELECT trajet, etape, distance
      FROM  trajets
      WHERE ville_vers = 'TOULOUSE' ORDER BY distance;

TRAJET                                                  ETAPE   DISTANCE
------------------------------------------------------- ---------- ----------
PARIS,CLERMONT-FERRAND,TOULOUSE                              2        795
PARIS,CLERMONT-FERRAND,MONTPELLIER,TOULOUSE                  3        995
PARIS,LYON,MONTPELLIER,TOULOUSE                              3       1015
PARIS,LYON,MONTPELLIER,CLERMONT-FERRAND,TOULOUSE             4       1485
```

Christian **Soutou**

Université Toulouse II - IUT Département Réseaux & Télécoms

Créez des rapports sophistiqués grâce à la série complète d'outils Altova®

ALTOVA® missionkit®

Voyez comment le Altova MissionKit®, d'Altova, la suite intégrée d'outils XML, de bases de données et d'intégration de données vous permet d'afficher et d'analyser des données en générant des diagrammes et rapports optimisés.

La génération de rapports enfin simplifiée et abordable grâce aux outils de rapport d'Altova MissionKit :

StyleVision® – outil de création de feuilles de style et de rapports

- Rapport consolidé basé sur le XML, les bases de données et/ou les données XBRL
- Sélection et rendu dynamique de données
- Création de rapports sous HTML, Microsoft Word, PDF et e-Form

XMLSpy® – éditeur XML avancé

- Création instantanée de diagrammes pour analyser des données XML
- Exportation en un clic de diagrammes vers XSLT, Xquery ou fichiers images

MapForce® – outil de mappage inter-données

- Intégration avec StyleVision pour un rendu de rapports attractif
- Génération de rapports pour presque tous les formats de données: XML, bases de données, EDI, fichiers plats, Excel 2007+ et plus encore

Téléchargement gratuit!

Testez avant d'acheter avec une version d'essai de 30 jours gratuite et entièrement fonctionnelle, disponible sur **www.altova.com**.

Agile et productif !

2e partie

© iStockPhoto.com/4x6

Le mois dernier, nous avions mis le focus sur la productivité et l'agilité des développeurs. Nous terminons, provisoirement, notre dossier, ce mois-ci avec une plongée en eau profonde de Scrum. Nous continuons à nous interroger sur l'intérêt des développeurs à s'y mettre. Bonne agilité !

Scrum ? C'est quoi ?

Le mot vient du Rugby; le "Scrum", traduit de l'anglais, représente la mêlée. Cela suggère l'idée d'une équipe soudée qui avance d'une façon très dynamique vers un but partagé. L'équipe est auto-organisée, et doit réagir rapidement aux événements imprévus pour réussir.

Scrum est le processus agile le plus utilisé dans le monde. Il est souvent combiné avec la méthode XP (Extreme Programming). Formalisé par Ken Schwaber et Jeff Sutherland au début des années 90, Scrum est assez simple à définir mais plus difficile à pratiquer.

Pour la plupart des organisations, l'implémentation de Scrum représente des changements de rôles, de responsabilité et d'interaction, c'est-à-dire une profonde transformation. Le comportement des gens et la culture d'entreprise ne sont pas faciles à changer, spécialement dans les organisations hiérarchiques, traditionnelles et monolithiques.

Les principes de SCRUM

Pour comprendre les bases de Scrum, il faut appréhender les aspects suivants:

- Cycle de vie – le cycle de vie du Processus Scrum
- Rôles – quels sont les rôles principaux et leur responsabilités
- Artefacts – simplification et réduction du nombre d'artefacts, à l'opposé de processus plus lourds
- Cérémonies – quelles sont les événements à mettre en place pour appliquer SCRUM ?

Cycle de vie

Le cycle de vie d'un produit développé avec Scrum se compose de cinq niveaux de planning :

1• Vision produit – le but ultime de cet investissement, de cette innovation.

2• Feuille de route du produit, qui donne d'un point de vue haut niveau les livraisons prévues, avec les thèmes principaux.

3• Livraison (Release) – normalement une période de 3-12 mois pour le développement d'une version significative pour livraison aux clients, avec une liste des fonctionnalités souhaitées.

4• Itérations ou "Sprints": périodes courtes de développement de 2 à 4 semaines, qui incluent la définition de besoin, l'analyse, le design, le développement, les tests et validations ainsi que les différentes cérémonies. Un projet est constitué de Sprints successifs.

5• Processus journalier – l'équipe commence et peut finir des tâches chaque jour, avec un haut niveau de transparence. Le Daily Standup Meeting et la mise à jour de l'Itération Backlog et du task board sont critiques.

Rôles

Les trois rôles principaux en Scrum sont le Product Owner, le Scrum Master, et l'équipe.

Product Owner
(ou Responsable Produit)

- Expert (~MOA) du domaine métier, et responsable pour les décisions de contenu fonctionnel (périmètre du produit)
- Pour les produits, gère le contact avec les clients finaux et comprend leur besoin

- Gardien de la vision et de la feuille de route
- Responsable de la gestion des priorités des exigences (ou Stories). Il doit également fournir des clarifications aux équipes projet si le besoin s'en fait sentir.
- Responsable de la planification des livraisons
- Validation fonctionnelle de premier niveau des livraisons

Scrum Master

- Garant de l'application du processus
- Protège et soutient l'équipe projet
- Gestion des aspects humains et relationnels
- Facilitateur, il veille à l'élimination des impondérables
- Ce n'est pas un chef ou un directeur de projet – ce rôle n'existe plus en SCRUM !

L'équipe

- Analyse et chiffrage des Stories (exigences) et des tâches, engagement collectif pour chaque Release et Sprint
- Réalisation du système
- Auto-organisée, polyvalente
- Est capable de gérer seule, sans intermédiaire, les interactions avec d'autres équipes et entités

Artefacts

Les artefacts sont l'ensemble des « livrables » résultant de l'application de SCRUM.

Product Backlog

La liste de l'ensemble des besoins priorisés par le Product Owner en collaboration avec l'équipe, qui représente tout ce que le produit doit offrir pour réaliser la vision. Les besoins prennent la forme de « Product Backlog Items » (ou PBI), dont la priorité relative est déterminée par le Product Owner.
Ces PBI sont ensuite raffinés en User Stories, estimées en « story points », qui peuvent être implémentées par une équipe une fois redécoupées en tâches.
Les story points n'ont pas d'unité : il s'agit d'une complexité relative d'une User Story par rapport à une autre. Il ne s'agit en aucun cas de jours hommes ou de points de fonctions !

Release Backlog

Extrait du Product Backlog prévue pour une Release. Cet artefact n'est présent que dans des projets de taille conséquente. Le Product Owner choisit au début de chaque période de livraison ce qui sort du Product Backlog pour entrer dans le release backlog.

Sprint (ou "Itération") Backlog

C'est l'ensemble des besoins fonctionnels et techniques à implémenter par l'équipe pour une itération, avec des estimations plus fines pour l'ensemble des tâches. Les tâches sont estimées en heures, et résultent d'un découpage des User Stories.

User Story

C'est l'unité de définition des exigences la plus fine en Scrum. Les User Stories peuvent être associées avec des tests d'acceptation, des maquettes, de la documentation.

Burndown chart

C'est un diagramme montrant le rapport entre l'avancement et le temps, soit pour une itération, soit pour une release. L'avancement est estimé en story points ; le temps en jours. Le Burn Down chart doit être visible de tous dans le lieu de travail

Task board

Le Task Board est simplement une surface plane disponible dans l'espace de travail. On y dispose les tâches, en général dans 3 catégories (A faire, en cours, fait). On peut également y afficher les User Stories afin d'établir la relation de manière visuelle entre une tâche et sa user Story. Chaque jour, les membres de l'équipe vont déplacer des tâches (matérialisées par des post-it) de la colonne « A faire » vers la colonne « en cours », ou de « en cours » vers « fait ».
Le task board est visible de tous, y compris le Product Owner.
Il permet d'éliminer la plus grande partie du reporting d'un projet classique.

Cérémonies

Les « cérémonies » en Scrum sont en fait des réunions imposées par le processus :

Vision planning

C'est le premier événement dans le cycle de vie d'un nouveau produit. L'équipe marketing peut être impliquée, ainsi que le Product Owner. Il s'agit de déterminer quelles sont les grandes fonctionnalités du produit, et quelle est sa valeur ajoutée pour le client final.

Création de la Release Roadmap

Pendant cet événement, le Product Owner, l'équipe et le Scrum Master élaborent un plan pour une série de releases avec les grands thèmes associés.

Sprint planning meeting

Le sprint planning meeting est un événement au début de chaque Sprint. Le Product Owner présente les User Stories qu'il a priorisées pour la prochaine itération, à concurrence de la capacité en story point de l'équipe. En effet, il est inutile de charger à 50 story points une équipe dont la vélocité est en moyenne de 20 story points !
L'équipe va ensuite découper en tâches chaque User Story. Chaque tâche est estimée en heure, par exemple avec la pratique du planning poker. Si le découpage en heures des tâches conduit à dépasser la capacité de l'équipe, alors elle peut refuser les User Stories de priorité plus faible.

Révision du Release backlog

Une révision périodique du release backlog est nécessaire, pour modifier les priorités, les complexités, ou ajouter de nouvelles Stories dans la backlog. Les modifications n'ont d'impact sur l'équipe qu'à la release suivante.

Daily Scrum (ou « Standup ») meeting :

Le daily meeting permet la synchronisation journalière de l'équipe et du Scrum Master, avec trois questions :
- Sur quoi ai-je travaillé hier ?
- Sur quoi vais-je travailler aujourd'hui ?
- Ai-je rencontré des problèmes ?

Cette réunion ne doit pas prendre plus de 10-15 minutes. Pas d'interruptions ni d'interaction ; s'il y a des discussions de coordination ou de résolution de problèmes, cela doit être APRES le Daily Standup meeting.

Sprint close
A la fin du Sprint, il y a une clôture finale, avec une revue de toutes les User Stories produites entre le Product Owner, le Scrum Master, et l'équipe. Le Product Owner pourra accepter ou rejeter les User Stories. La vélocité réelle est calculée selon la somme de toutes les stories complétées et acceptées.

Démo
Dans la démo à la fin de chaque itération, un membre de l'équipe présente le produit afin de valider les tests d'acceptation de chaque story. Le Product Owner participe à la démonstration. C'est une opportunité pour échanger avec le Product Owner et les utilisateurs, de mettre en exergue rapidement ce qui a de la valeur, et ce qui est mal compris ou mal expliqué.

Rétrospective du sprint
A la suite du sprint review, après une pause, l'équipe, le Product Owner, le Scrum Master et les parties prenantes du projet (qui veulent y participer) font la rétrospective de l'itération. Cela permet de faire le point sur les bonnes et mauvaises pratiques appliquées durant l'itération et de faire émerger des actions d'améliorations pour les prochaines itérations. A la suite du Sprint review et de la rétrospective, le Scrum Master doit réaliser un Bilan d'itération en y rapportant tous les éléments des deux réunions mais aussi pour communiquer la vélocité et les métriques à toutes les parties prenantes du projet.

Facile à mettre en œuvre ou pas?

SCRUM comporte finalement peu de principes, si on compare à une méthodologie traditionnelle. Cependant, notre expérience sur des déploiements de grande ampleur nous a permis d'observer certaines dérives :

Commencer par intégrer les règles de base avant de personnaliser l'agilité à la "sauce" Entreprise.
La tendance actuelle est à l'Agilité, et de ce fait les entreprises revendiquent **leur** méthode Agile, façonnée, revue et corrigée par leurs spécificités. Pourquoi pas ? Sauf que nous assistons à des dérives et une distorsion des méthodes Agiles qui n'en n'ont plus que le nom.
Les entreprises ne prennent que les éléments qui les arrangent et ne modifient si possible en rien leur organisation. Aujourd'hui, les entreprises retiennent du concept : livraison dans les temps, pas de débordement dans les projets au niveau budget et deadline. Changement du périmètre comme on veut, une certaine flexibilité (avec une tendance à l'ajout plus qu'à la contrepartie). Prenons certains principes de l'agilité et voyons ce que les entreprises en font:
Faire faire des estimations par une équipe projet : Certaines entreprises l'ont intégré, d'autres non. C'est coûteux de faire intervenir plus de 2 individus pour des estimations, un chef de projet et un leader technique suffisent. Une équipe doit réaliser dans les temps impartis par un chef de projet, un point c'est tout, prendre en compte le rythme de travail des membres d'une équipe est quelque chose d'inconcevable.
Refaire des ajustements d'estimations au fur et à mesure des sprints : les entreprises prennent pour comptant les estimations faites au début du projet et ont du mal à comprendre que des réajustements soient faits au fur et à mesure que l'on gagne en visibilité. La culture d'amélioration de la finesse des estimations est théorique et doit si possible rester sur papier
Fixer des équipes Scrum : ce point est pris en compte pour des projets avec des équipes dédiées, et cela fait partie de l'organisation de l'entreprise. Mais cela ne fonctionne plus dès que les projets se partagent des ressources communes.
Les tests : Les tests sont coûteux et tout le monde est d'accord pour les intégrer le plus tôt possible. Mais même en agilité, les tests n'ont pas de valeur business et restent un centre de coût pour le client et pour le réalisateur. Les entreprises ayant des équipes de tests estiment qu'il faut rentabiliser cette activité et intégrer des testeurs relativement tôt dans le processus de développement. Mais la réalité est autre, les testeurs sont intégrés tardivement dans le processus de développement. Nous assistons donc parfois à du waterfall déguisé intégré dans l'Agilité.
Remettre de l'humain dans les projets : ce concept est invisible et n'est jamais retenu par les managers… une équipe travaillant dans une tranche d'heures acceptable, c'est tout simplement inconcevable, vu l'état du marché et la crise. C'est un des concepts de Scrum qui n'est pas populaire et jamais cité…
Il y a de quoi écrire des pages et des pages sur la manière dont les entreprises se sont appropriées les différents principes de l'Agilité. Cette tendance a amené bon nombre d'entreprises à adapter la méthode Scrum avant même d'avoir essayé d'appliquer les principes standard de base au moins une fois - avec des résultats non concluants.
Le retour d'expérience de Valtech, est que les entreprises intègrent d'abord les principes de base, les mettent en application sur un pilote en respectant les règles et les adaptent ensuite au vu des résultats et de leurs contraintes organisationnelles et culturelles.

Lectures

Scrum Primer, 2010. http://scrumtraininginstitute.com/home/stream_download/scrumprimer

Scrum and XP from the Trenches, *Henrik Kniberg*

Agile Project Management with Scrum, *Ken Schwaber*, 2004

Agile & Iterative Development, A Manager's Guide, *Craig Larman*, 2004

Scaling Lean and Agile Development, *Craig Larman and Bas Vodde*, 2008

Practices for Scaling Lean and Agile Development, *Larman and Vodde*, 2010.

■ Gladys **N'toumi**
■ Yannick **Ameur**
■ Greg **Hutchins**
■ Pascal **Ognibene**
Consultants Valtech
sur les transformations Agiles.

TOUTES LES TECHNOLOGIES MICROSOFT®

La référence officielle

9782100547418 ▪ 640 pages ▪ 49 €

9782100547425 ▪ 624 pages ▪ 39 €

9782100547449 ▪ 576 pages ▪ 39 €

9782100543045 ▪ 416 pages ▪ 39 €

9782100551170 ▪ 368 pages ▪ 19,90 €

Tous nos ouvrages sont disponibles en librairie

Réalisation : MATEO

www.dunod.com

Microsoft® Press

Un sprint dans la vie de l'équipe d'Urban Turtle à Pyxis Technologies

Urban Turtle est un produit s'intégrant à Team Foundation Server et plus particulièrement à Team System Web Access. Il regroupe toute l'expérience de Pyxis Technologies dans la conduite et la réalisation de projets Agiles. Demandons à son équipe ses recettes pour construire « agilement » un produit Agile.

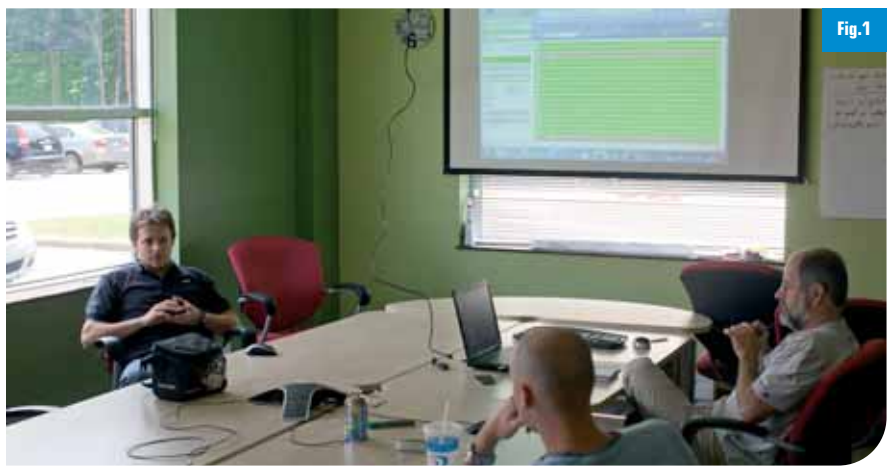


Fig.1

L'équipe durant un « Daily meeting »

Depuis le début du développement du produit en 2008, l'équipe d'Urban Turtle est composée de 5 à 7 équipiers. Elle applique la méthode Scrum pour la conduite de projet et adapte, au rythme des différents sprints, ses pratiques pour mettre à la disposition des utilisateurs des versions de plus en plus complètes et s'intégrant aux nouvelles versions de Team Foundation Server.
De toutes les questions que les groupes d'utilisateurs et les forums traitant de l'Agilité et de Scrum regroupent, il en est une qui revient régulièrement sur le devant de la scène : *"Comment mettre en place Scrum dans un contexte d'éditeur logiciel ?"*
Les éditeurs ressentent en effet dès les premiers points de présentation de Scrum les difficultés qu'ils peuvent rencontrer dans leur contexte. Comment nommer un propriétaire de produit *(Product Owner)* représentant la majorité de mes utilisateurs? Comment livrer plus souvent et plus régulièrement sans faire décoller le coût de mes livraisons ? À ces questions, Dominic Danis, précédemment ScrumMaster et qui a désormais le rôle de propriétaire de produit, présente sa mission : *« Ma mission au sein de l'équipe d'Urban Turtle ne diverge pas de la mission classique d'un propriétaire de produit. Je dois bien sûr fournir à l'équipe une description des fonctionnalités et leurs conditions de succès respectives. À chaque revue de sprint, je m'assure également de leur application. Bien sûr, je suis plus éloigné des utilisateurs finaux que dans le cadre d'un projet d'entreprise mais je m'assure que le produit réponde aux besoins du marché en contactant régulièrement par divers médias nos clients et clients potentiels. »*
Chacun des utilisateurs ayant demandé une licence d'évaluation sera ainsi contacté pour donner son avis sur le produit actuel et éventuellement intégrer ses besoins particuliers. Dominic précise : *« Il s'agit bien ici d'entendre et d'apprécier toutes les contraintes d'un client pour déterminer si sa demande peut être intégrée dans notre produit. Je reste toutefois le garant de la vision. Par exemple, notre objectif avec Urban Turtle est avant tout de faciliter la mise en place et l'application de la méthode Scrum avec Team Foundation Server. Certaines demandes qui vont à l'encontre de notre vision ne pourront être intégrées dans le carnet de produit (product backlog) ».*
Afin de se rapprocher de ses utilisateurs, l'équipe a par ailleurs décidé d'inviter le public à ses différentes réunions (planification, revue…). Ces réunions sont diffusées sur le Net à l'aide du site UStream (http://www.ustream.tv/channel/urbanturtle). Chaque « poule » pouvait alors réagir à ce qui y était décidé dans le forum du produit (http://urbanturtle.com/forums/) ou en communiquant directement avec l'équipe [Fig.1].
Cette mise en avant de l'équipe a également un autre objectif. Dans l'impossibilité de présenter le fruit de son travail à un panel d'utilisateurs, c'est au propriétaire de produit que l'équipe doit montrer que les différentes tâches et les scénarios sur lesquels elle s'était engagée en début de sprint ont été bien réalisés. La revue a pourtant un impact

Fig.2

Les coûts du projet, facteur de responsabilisation

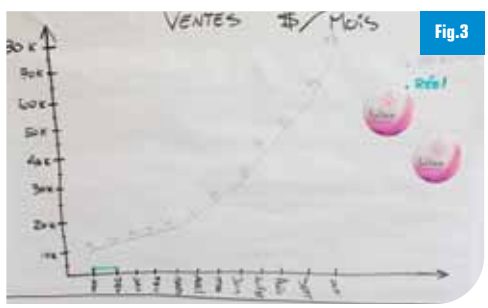


Fig.3

Les ventes, vecteur de motivation

significatif sur la responsabilisation des développeurs. Dans le cadre du développement du produit, c'est en intégrant d'autres notions ou d'autres rapports que l'équipe trouve les vecteurs de motivation et de responsabilisation. Ainsi, les murs de la « War Room » de l'équipe présentent l'information relative au suivi des ventes et du coût global du développement. Ces données sont reprises dans le wiki d'entreprise. Elles permettent à chaque partie prenante du projet de connaître les résultats de l'équipe[Fig.2 et 3].

Car si un des objectifs de l'Agilité (par la mise en place de pratiques Agiles) est bien de revenir à un rythme soutenable (voire agréable), il est certain qu'un éditeur, au-delà de l'équipe de développement de son produit, doit faire face à un certain nombre d'impondérables. Par exemple, la sortie d'une nouvelle version du logiciel d'un concurrent est un évènement qui rend primordial de présenter de nouvelles fonctionnalités. Tous ces évènements pèseront au final sur les jalons de l'équipe et donc sur les équipiers. Pour Urban Turtle, la solution réside dans la faculté de l'équipe à livrer régulièrement : *« Notre rythme de livraison nous permet d'offrir de nouvelles fonctionnalités, des correctifs et des améliorations ergonomiques chaque mois. Ces versions sont le produit du travail de l'équipe pendant 2 sprints. Nous poursuivons nos efforts afin de raccourcir encore la mise sur le marché, et nous avons l'espoir qu'un jour, nous pourrons mettre notre produit à la disposition des utilisateurs, et ce, à la demande ou quotidiennement. »*

Techniquement, les développeurs répondent à ce besoin de réactivité à l'aide de pratiques fondamentales.

« Toutes les pratiques et les techniques que nous mettons en place ont pour objectif de raccourcir le temps nécessaire pour livrer les nouvelles fonctionnalités, tout en garantissant l'absence de régressions au fil des différentes versions. » En tout premier lieu, la conception et l'architecture tendent à minimiser les dépendances entre les différentes fonctionnalités et l'application de base (Team System Web Access). Récemment, la mise en place d'un modèle de commande lié au moteur d'inversion de contrôle Windsor a permis de découpler chaque fonctionnalité à la page Web qui l'héberge. Plus généralement, l'application des pratiques et des recommandations du « Domain-Driven Design » guide l'équipe dans la conception et l'architecture de leur produit.

Un effort important est également consenti sur les tests et le binômage, qui permettent d'assurer le fonctionnement et la lisibilité du code. Dans la définition de « Done », la liste de vérification avant la fermeture d'une tâche, d'un scénario utilisateur ou d'un sprint, on retrouve également des points concernant la livraison du produit (documentation de livraison ou création du programme d'installation dans les différentes langues supportées) [Fig.4].

L'automatisation des procédures est également un point sur lequel l'équipe travaille en permanence. Les tests s'appuient sur le framework de tests unitaires de Visual Studio. De plus, les tests Web historiquement basés sur Selenium ont été remplacés par des tests unitaires : *« La durée d'exécution de ces tests ne nous permettait pas de les lancer aussi souvent que nous le souhaitions. De plus, maintenir ces tests de par leur très forte dépendance envers l'interface, qui évolue elle aussi à chaque version, devenait bien trop coûteux. ».*

Tout ce qui ne peut pas être automatisé est documenté de façon succincte mais exhaustive dans le wiki de l'équipe. Tout nouveau membre pourra donc prendre connaissance des standards et des procédures de livraison.

L'équipe pratique également le « dog fooding », qui consiste à gérer le développement d'Urban Turtle. Les développeurs seront donc parfois les premiers à détecter une anomalie. Cette pratique permet également de partager les connaissances de chaque aspect du produit et de communiquer plus facilement lors des différentes réunions. La communication et la collaboration au sein de l'équipe sont aussi des aspects qu'il convient de citer. Dominic Danis présente cette collaboration : *« C'est être un mauvais gestionnaire que de croire que l'on connaît et que l'on maîtrise tout. Je suis présent aux réunions quotidiennes (daily meetings) pour présenter mon travail et mes idées aux membres de l'équipe. Il est vrai que cette démarche nécessite une certaine maturité de l'équipe, mais une fois un climat de confiance installé, on gagne à partager ses choix et ses difficultés. En retour, le ScrumMaster et le propriétaire de produit sont à l'écoute des développeurs. Un Niko-niko Calendar est disponible pour permettre de faire état de son état émotionnel ou de motivation »* [Fig.5].

Au final, un observateur qui prendrait part à une des réunions de l'équipe de développement d'Urban Turtle ne pourrait que ressortir convaincu de l'efficacité des conseils du framework Scrum pour le développement d'un produit logiciel. Il reste que la mise en place de cette équipe a pris du temps, qu'elle a subi des remaniements au fil des livraisons pour devenir ce qu'elle est aujourd'hui : un ensemble de personnes responsables, autogérées et tournées vers le même objectif.

Fig.4

La définition de 'Done'

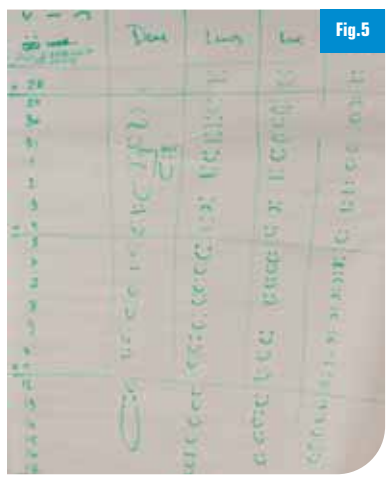


Fig.5

Un Niko-niko Calendar

■ Mathieu **Szablowski** - *Pyxis*

Géolocali

sation

La *killer app* !

Elle fait partie de notre quotidien. On l'utilise sans y faire attention dans les Smartphones, sur son PC. Ses données sont stockées dans les bases de données. De quoi s'agit-il ? De la géolocalisation.

Cette killer app est désormais incontournable pour les applications ! Et offre un petit plus aux utilisateurs : situer un lieu, voir ce qu'il y a autour, calculer un itinéraire, etc. Sur son Smartphone, c'est une des fonctions les plus utiles. N'avez-vous jamais utilisé Plans sur votre iPhone ? Il est devenu normal de voir la position d'un monument, d'une rue, d'un hôtel en quelques secondes avec une carte précalculée…

Finalement est-ce compliqué d'intégrer un service de géolocalisation à son site, à son application mobile ? En quelques mois, les API et les modèles de développement ont évolué et facilitent grandement leur intégration. Aujourd'hui, nous avons principalement trois grands services : Map d'Apple, Google Maps de Google et Bing Maps de Microsoft.

Le service d'Apple repose sur les cartes de Google qui possède une énorme richesse de données. Et MapKit d'Apple évolue constamment pour rajouter des fonctionnalités. Ainsi depuis la sortie de iOS 4, il est possible de créer une carte personnalisée rapidement ou encore de créer un itinéraire (via Google Directions). Et s'il faut passer par le langage Objective-C (ou d'autres langages, car depuis peu Apple autorise d'autres outils de développement que XCode), MapKit n'est guère différent d'un autre service de géolocalisation…

Google Maps est sans conteste le leader du marché avec sa version gratuite et les services professionnels. Et surtout, pour faciliter l'usage de Maps par d'autres technologies, Google propose des API Maps pour Javascript, Flash, sous forme de web services (XML et JSON), sans oublier des API pour créer des cartes statiques, utiliser Earth ou encore rajouter, stocker et manipuler des données via Maps Data !

Mais dès aujourd'hui, une nouvelle dimension de la « géoloc » existe : dans le cloud computing. Les principaux fournisseurs de Paas et Iaas proposent des API pour géocaliser le Datacenter que l'application peut utiliser. Et cette possibilité est d'une puissance imparable pour rapprocher l'application, la donnée de l'utilisateur. Preuve que la géoloc couvre des domaines très différents et si vitaux pour l'informatique moderne.

■ François Tonic

Google Maps : bien débuter en javascript

Google Maps est principalement connu en tant que service d'affichage de carte cartographique à destination des sites internet. Mais c'est bien plus que cela ! Parmi les fonctionnalités offertes par l'API, quelques-unes sont particulièrement sympathiques et simples à mettre en oeuvre.

La dernière version de l'API Google Maps est la version 3, nommée V3. Elle a été complètement réécrite afin d'offrir de bien meilleures performances que son prédécesseur la V2, notamment pour les téléphones de type Android et iPhone. Mais attention tout de même, cette V3 est récente et n'implémente pas encore toutes les fonctionnalités présentes dans la V2.
Bien sur, la V2 sera encore maintenue par les équipes de Google pour garantir le bon fonctionnement des sites internet l'utilisant. Cependant son état est passé dans un mode « deprecated ». Les sites devront dans les mois qui viennent commencer à entamer une migration, pour bénéficier de performances accrues, mais surtout pour profiter des futures fonctionnalités.
La série d'exemples suivants illustre les syntaxes permettant de mettre en œuvre et personnaliser l'affichage de cartes.

GOOGLE MAPS V3

L'API javascript est très proche de la V2, cependant beaucoup de choses ont changé dans son implémentation interne. Comme je le disais précédemment, cette version est axée sur la rapidité de chargement, de la librairie et des cartes. Une autre particularité de la V3 est qu'elle n'a plus besoin de clef pour être utilisée par un site internet.

La création d'une carte simple

Pour commencer simplement, nous allons afficher une carte centrée sur les coordonnées géographiques: latitude = 48.856667 et longitude = 2.350987 correspondant à la ville de «Paris». [Fig.1]

```
<!DOCTYPE html>
<html>
<head>
<meta name=»viewport» content=»initial-scale=1.0, user-scalable=no» />
<style type=»text/css»>
  html { height: 100% }
  body { height: 100%; margin: 0px; padding: 0px }
</style>
<script type=»text/javascript»
    src=»http://maps.google.com/maps/api/js?sensor=true»>
</script>
<script type=»text/javascript»>
  function initialize() {
   var latlng = new google.maps.LatLng(48.856667, 2.350987);
    var myOptions = {
      zoom: 8,
      center: latlng,
      mapTypeId: google.maps.MapTypeId.ROADMAP
    };
    var map = new google.maps.Map(document.getElementById
(«map_canvas»), myOptions);
  }
</script>
</head>
<body onload=»initialize()»>
 <div id=»map_canvas» style=»width:100%; height:100%»
></div>
</body>
</html>
```

Dans cet exemple, les points suivants sont importants :

- L'application est déclarée en HTML 5 avec la déclaration <DOCTYPE html>. Ce mode va permettre à la grande majorité des navigateurs web d'effectuer le rendu de la page de façon identique. Il a aussi l'avantage d'être ignoré par les anciens navigateurs, pour être remplacé par un autre mode de rendu appelé «quirks mode».
- L'API javascript Google Maps est chargée avec le tag <script> avec comme source «http://maps.google.com/maps/api/js?sensor=true». Toutes les définitions et symboles de l'API V3 seront ainsi chargés. Le paramètre « sensor » (true ou false) indique si le navigateur client embarque un système de positionnement GPS (cas de certains smartphones)
- Un élément nommé «map_canvas» accueille l'affichage de la carte. Il s'agit d'un simple élément de l'arbre DOM du navigateur.
- Un objet javascript «myOptions» contient les propriétés de la carte, nécessaire à son initialisation. Cet objet possède les informations pour centrer la carte par rapport à un point de référen-

Fig.1

ce, le niveau de zoom et le type de rendu de carte.

- Un autre objet javascript contient l'instance de la carte de type «google.maps.Map». Il a comme paramètre de constructeur l'élément référençant le et l'objet contenant les propriétés par défaut. Si l'on désire avoir plusieurs cartes sur une même page, il suffira alors de créer autant d'objets que nécessaire.
- Le chargement s'effectue dans le tag <body> sur l'événement «onload». Cet événement est appelé lorsque la page est complètement construite, c'est-à-dire lorsque le chargement des images et des scripts est terminé.
- Le point de référence est déclaré comme un objet javascript de type «google.maps.LatLng» avec comme paramètre de constructeur la latitude et la longitude.

Ajout d'un marqueur

Nous souhaitons maintenant afficher un marqueur sur la ville de «Paris». Ce marqueur affichera un message, lorsque l'utilisateur positionnera le pointeur de la souris dessus.

Nous ajoutons la création d'un objet de type «google.maps.Marker» à notre exemple précédent dans la méthode initialize(), après la création de la carte. Pour s'afficher au bon endroit et sur la bonne carte, cet objet à besoin de la position géographique de la ville de «Paris» et d'une référence à notre objet «map».

```
var marker = new google.maps.Marker({
      position: latlng,
      map: map,
      title:»Hello Paris!»
  });
```

Fig.3

Fig.4

Nous obtenons l'affichage suivant dans le navigateur: [Fig.2]

Changer l'image du marqueur est bien évidement possible. Il suffit d'indiquer l'emplacement d'une nouvelle image [Fig.3] et l'API s'occupera de la charger et de la re-dimensionner [Fig.4].

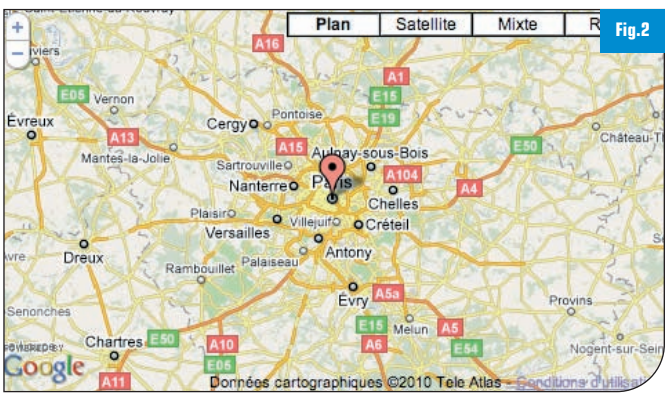
```
var image = 'flower.png';
var marker = new google.maps.Marker({
  position: latlng, map: map, title:»Hello Paris!», icon: image
});
```

Service de Geocoding

Le terme Geocoding signifie convertir une adresse comme «Paris, France» en coordonnées géographiques (latitude, longitude). L'API propose une classe offrant ce service. Cependant, son utilisation est limitée à un certain nombre de requêtes, ce qui permet de prévenir tout abus. L'accès au service s'effectue par un appel asynchrone, une «callback» est passée en paramètre lors de l'appel. La «callback» sera ensuite exécutée, avec les résultats provenant des serveurs de Google. Si l'adresse passée n'est pas suffisamment précise; exemple : la ville «Toledo», le service pourra renvoyer plusieurs coordonnées géographiques dans le monde, et comme premier résultat la ville «Toledo au US». En revanche si l'adresse passée est plus précise, exemple la ville «Toledo en Espagne» : «Toledo, es», alors le résultat renvoyé sera unique et correspondra bien à notre recherche [Fig.5].

En pratique, le service dans la V3 a tendance à ne renvoyer qu'un seul résultat, probablement pour des raisons de performance.

Fig.2




```
var map;
var geocoder;

function initialize() {
 ....
 map = new google.maps.Map(document.getElementById
                          («map_canvas»), myOptions);
 geocoder = new google.maps.Geocoder();
 }

function codeAddress() {
 var address = document.getElementById(«address»).value;
 geocoder.geocode( { 'address': address}, function(results,
status) {
   if (status == google.maps.GeocoderStatus.OK) {
     map.setCenter(results[0].geometry.location);
     var marker = new google.maps.Marker({
       map: map,
       position: results[0].geometry.location
     });
   } else {
     alert(«Geocode was not successful for the following
reason: « + status);
   }
 });
}
</script>
</head>
<body onload=»initialize()»>
 <div id=»map_canvas» style=»width: 320px; height: 480px;
»></div>
 <div>
   <input id=»address» type=»textbox» value=»toledo, es»>
   <input type=»button» value=»Encode» onclick=»codeAddress()»>
 </div>
</body>
```

Service de Reverse Geocoding

Ce service permet d'effectuer l'inverse du service de Geocoding. A partir d'une position géographique, nous souhaitons obtenir une liste d'adresses lisibles par un utilisateur.

Nous interrogeons le service avec la position : 48.856667, 2.350987, et nous obtenons comme résultat, la liste suivante :

1) 1-5 Place de l'Hôtel de Ville, 75004 Paris, France [Fig.6]
2) 4ème Arrondissement, 75004 Paris, France
3) 75004 Paris, France
4) Paris, 75004 Paris, France
5) Paris, France
6) Ile-de-France, France
7) France

```
function codeLatLng() {
  var geocoder = new google.maps.Geocoder();
  var input = «48.856667, 2.350987»;
  var latlngStr = input.split(«,»,2);
  var lat = parseFloat(latlngStr[0]);
  var lng = parseFloat(latlngStr[1]);
  var latlng = new google.maps.LatLng(lat, lng);
  geocoder.geocode({'latLng': latlng}, function(results, status) {
    if (status == google.maps.GeocoderStatus.OK) {
      if (results[0]) {
        map.setZoom(11);
        marker = new google.maps.Marker({
          position: latlng,
          map: map
        });
        infowindow.setContent(results[0].formatted_address);
        infowindow.open(map, marker);
      }
    } else {
      alert(«Geocoder failed due to: « + status);
    }
  });
}
```

Service pour calculer une altitude

Nous allons utiliser la classe «google.maps.ElevationService» en lui passant comme paramètre la position géographique de «Paris». Le service appellera ensuite la callback avec le résultat attendu [Fig.7].

```
function getElevation() {
  var elevator = new google.maps.ElevationService();
  var latlng = new google.maps.LatLng(48.856667, 2.350987);
  var locations = [];
  locations.push(latlng);
  var positionalRequest = {'locations': locations}
  elevator.getElevationForLocations(positionalRequest, function
            (results, status) {
    if (status == google.maps.ElevationStatus.OK) {
      if (results[0]) {
        infowindow.setContent(«The elevation at this point is « +
              results[0].elevation + « meters.»);
        infowindow.setPosition(latlng);
        infowindow.open(map);
      } else {
        alert(«No results found»);
      }
    } else {
      alert(«Elevation service failed due to: « + status);
    }});}
```

Service calcul de distances et d'itinéraires

L'API v3 permet également de requêter :
- Des itinéraires complets avec affichages texte et plan (comme Mappy, ViaMichelin,…)
- Des distances entre 2 lieux
- Des durées moyennes de trajet (pouvant tenir compte de la circulation)

Les résultats peuvent être renvoyés préformatés (itinéraires HTML + trajet sur la carte) ou aux formats JSON et XML (avec le webservice par URL dédié), les itinéraires peuvent également varier en fonction du mode de transport choisi : voiture, piéton ou vélo.

Exemples d'utilisation :

Sur le site www.onbooje.com (service à la personne de proximité) cette API est utilisée pour indiquer une distance entre un employé et un employeur, tout en conservant la confidentialité des adresses (envoi d'adresses partielles). Les résultats d'annonces peuvent également être triés sur le critère « proximité » [Fig.8]. Commençons un exemple en définissant 2 div dans la partie HTML :
- 1 div pour afficher la carte et de l'itinéraire
- 1 div pour afficher les instructions d'orientation.

```
<div id=»myMap» style=»width:400; height:400»></div><br>
<div id=»myItineraire»></div>
```

Pour alimenter ces div, nous allons utiliser les classes :
- google.maps.DirectionsService : permet le calcul d'itinéraire

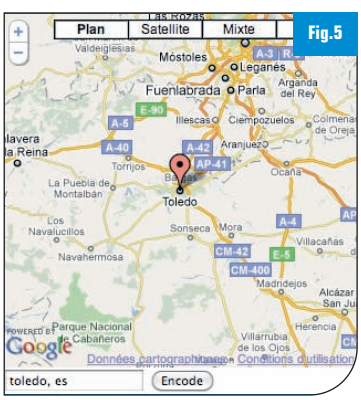


Fig.5

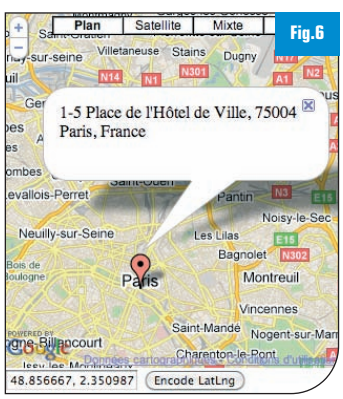


Fig.6

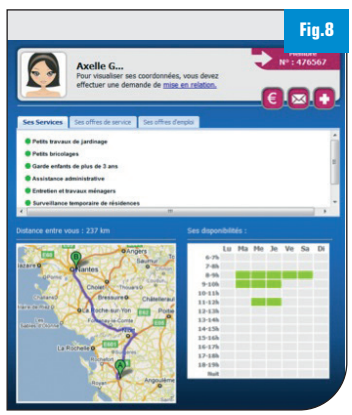


Fig.8

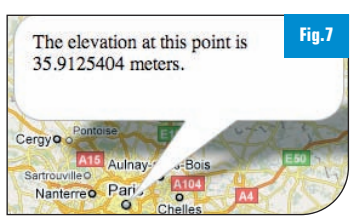


Fig.7

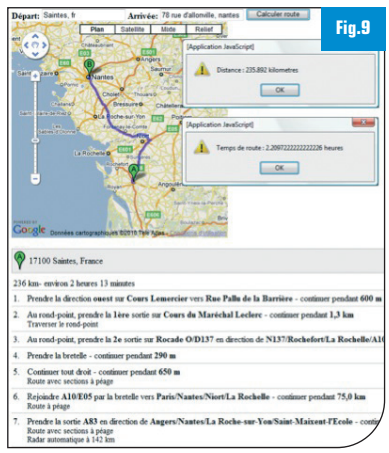


Fig.9

• google.maps.DirectionsRenderer : affichage carte et instructions

Avant d'appeler le calcul de distance, instancier un objet « DirectionsRenderer » et l'associer au deux div précédentes avec les méthodes setPanel(DOM element) et setMap(GoogleMap), en initialisant une carte vierge:

```
directionsDisplay = new google.maps.DirectionsRenderer();
map = new google.maps.Map(document.getElementById(«myMap»));
directionsDisplay.setMap(map);
directionsDisplay.setPanel(document.getElementById(«myItine
raire»));
```

L'API de calcul « DirectionsService » est ensuite interrogeable en AJAX sur n'importe quel évènement, elle attend un objet JS précisant au moins les lieux origine, la destination et le mode de transport choisi :

```
 var request = {
 origin: «Saintes, fr»,
 destination: «78 rue d'allonville, nantes»,
 travelMode: google.maps.DirectionsTravelMode.DRIVING
 };
```

Notes : Il est possible d'envoyer des adresses postales standard (complètes ou non, le cas échéant il prendra le « centre-ville ») ou des coordonnées géocodées.

D'autres options sont disponibles dans la requête : unité de mesure (km ou miles), ajout d'étapes (waypoints), optimisation de trajets, pas d'autoroute / pas de péage, etc.

L'appel est ensuite effectué grâce à la méthode « route() » et la réponse est exécutée sur le callback. En cas de succès, les informations d'itinéraires peuvent finalement être affichées grâce au renderer défini plus haut : [Fig.9].

```
var directionsService = new google.maps.DirectionsService();
directionsService.route(request, function(response, status) {
   if (status == google.maps.DirectionsStatus.OK) {
     directionsDisplay.setDirections(response);
   }});
```

Il convient de tester en premier lieu le résultat du calcul (status) en le comparant aux constantes définies dans DirectionsStatus dont voici les principales :

• OK indique que la réponse contient un itinéraire valide
• NOT_FOUND : une des localisations spécifiées n'a pas été trouvée, ou n'est pas géocodable (cas des adresses incomplètes ou inexistantes)
• INVALID_REQUEST : format de requête invalide
• OVER_QUERY_LIMIT : le nombre de requêtes GoogleMaps est limité pour une période donnée, le maximum a été atteint (voir licence « Premier » pour augmentation)
• UNKNOWN_ERROR : erreur inconnue/serveur.

```
....
<script type=»text/javascript»>
 var directionDisplay;
 var directionsService = new google.maps.DirectionsService();
 var map;

 function initialize() {
   directionsDisplay = new google.maps.DirectionsRenderer();
   var myOptions = {
   zoom:7,
   mapTypeId: google.maps.MapTypeId.ROADMAP,
   center: new google.maps.LatLng(46.850033, -0.6500523)
 }
 map = new google.maps.Map(document.getElementById(«myMap»),
myOptions);
 directionsDisplay.setMap(map);
 directionsDisplay.setPanel(document.getElementById(«myItineraire»));
}

function calcRoute() {
 var start = document.getElementById(«start»).value;
 var end = document.getElementById(«end»).value;
 var request = {
   origin:start,
   destination:end,
   travelMode: google.maps.DirectionsTravelMode.DRIVING
 };
 directionsService.route(request, function(response, status) {
 alert(«Résultat du calcul : «+status);
 if (status == google.maps.DirectionsStatus.OK) {
   directionsDisplay.setDirections(response);
     // Affichage de la distance et de la durée du trajet
   alert(«Distance : «+response.routes[0].legs[0].distance.
value/1000 + « kilometres»);
   alert(«Temps de route : «+response.routes[0].legs[0].
duration.value/60/60 + « heures»);
   }
 });
}
</script>
.....
<body onload=»initialize()»>
<div>
<b>Départ: </b>
<input type=»text» id=»start» value=»Saintes, fr»/>
<b>Arrivée: </b>
<input type=»text» id=»end»  value=»78 rue d'allonville, nantes»/>
<input  type=»button»  value=»Calculer  route»  onclick=»calc
Route()»/>
</div>

<div id=»myMap» style=»width:400; height:400»></div><br>
<div id=»myItineraire»></div>
</body></html>
```

■ Anthony **Quinault**
Consultant / Architecte JEE freelance
aquinault@cadesoft.com - Membre du Poitou-Charentes JUG
http://www.poitoucharentesjug.org

■ Fabien **Guibert**
Consultant / Architecte JEE
NeObject : Expertise - Formation - Production
postmaster@neobject.fr - www.neobject.fr

Google Maps sur Android : vive la géoloc dans sa poche !

Dès la première version, Google a souhaité faire bénéficier sa plateforme Android des fonctionnalités qu'il offrait sur le web : fonctions de recherche, de communication (mail et messagerie instantanée) et fonctions cartographiques. Cet article propose un tour d'horizon d'une des API les plus populaires : celle de la gestion des cartes sur les téléphones mobiles.

Les services Google Maps sur Android, sont accessibles à travers une API conçue pour intégrer dans les applications des fonctionnalités cartographiques. Pour l'habitué de Google Maps en Javascript, cette API mobile est un sous-ensemble simplifié des possibilités disponibles à travers les navigateurs. Cette similarité simplifie la prise en main mais peut décevoir par l'absence de fonctionnalités puissantes comme la gestion des fichiers KML.

C'est pour cela que même si nous présentons uniquement les aspects liés à l'API « Google Maps », il faut concevoir les aspects géographiques d'une application à travers toutes les possibilités de la plateforme Android.

Les ***WebView*** sont un moyen de retrouver les possibilités du Javascript et l'application « Google Maps », en standard, offre un autre éventail de possibilités. L'API mobile fournit aux applications Android une interactivité avec la cartographie qui autorise des fonctionnalités inédites sur un téléphone mobile.

Nous allons détailler les points clés pour développer une telle application avec le SDK Android 2.x et l'IDE Eclipse.

Un démarrage légal et technique

La première étape pour utiliser les services de Google Maps est d'être autorisé, pour cela il faut s'enregistrer auprès de Google et accepter les conditions d'utilisation du service. Ces conditions imposent des limitations importantes sur les applications qui peuvent être développées: la gestion de flotte est, par exemple, proscrite. Compte tenu des enjeux économiques liés à la navigation sur téléphone mobile, l'étude détaillée du contrat d'usage n'est pas à négliger. Une fois les conditions acceptées, le processus de génération d'une clé peut démarrer.

En pratique, deux clés doivent être générées pour autoriser l'usage. Une est liée au certificat de debug et l'autre à celui servant à la signature de l'application. Pour cela, on produit le hash MD5 du certificat avec l'outil du SDK Java : ***keytool*** appliqué sur le fichier contenant le certificat, pour le debug : ***debug.keystore***. Le code, ainsi généré, est fourni au site Google d'enregistrement (code.google.com) et associé avec un compte Google mail pour produire la clé utilisée lors de l'initialisation des API Google Maps.

Les API Google Maps ne sont pas les API standard d'Android, il faut donc créer un projet particulier qui les inclut.

Pour cela, on sélectionne lors de la création du projet un profil de plateforme qui supporte « Google API ». Deux points importants sont à prévoir dans le « manifest » du projet :

- Autoriser les accès à Internet et l'accès à la position GPS
- Ajouter la bibliothèque Google Maps du Mobile.

<exemple de manifest>

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res
/android"
 package="org.ot.map" android:versionCode="1" android:version
Name="1.0">
 <application android:icon="@drawable/icon" android:label=
"@string/app_name"
   android:theme="@android:style/Theme.NoTitleBar">
   <activity android:name=".POIActivity" android:label="@
string/app_name">
     <intent-filter>
       <action android:name="android.intent.action.MAIN" />
       <category android:name="android.intent.category.
LAUNCHER" />
     </intent-filter>
   </activity>
   <uses-library android:name="com.google.android.maps" />
 </application>
 <uses-sdk android:minSdkVersion="3" />

 <uses-permission android:name="android.permission.ACCESS_
FINE_LOCATION"></uses-permission>
 <uses-permission android:name="android.permission.INTERNET">
</uses-permission>

</manifest>
```

Une intégration « sans couture » avec la plateforme

Les objets de base de l'API sont des spécialisations des objets habituels que l'on retrouve dans chaque application Android :

- L'activité « ***MapActivity*** » permet d'intégrer les fonctionnalités de Google maps dans le cycle de vie des applications.
- La vue « ***MapView*** » va gérer la visualisation des cartes avec leur différentes options (rues, trafic, zoom), c'est elle qui va interagir avec les serveurs de Google pour afficher la carte.

La ***MapActivity*** gère les événements normaux du cycle de vie dans lesquels on va pouvoir insérer les actions liées aux autres objets : création, libération, gestion de la pause et de la reprise. Il est à

noter que deux méthodes doivent être implémentées : ***isRouteDisplayed()*** et ***isLocationDisplayed()*** pour indiquer l'usage de fonctionnalités de guidage ou de localisation par GPS.
La ***MapView*** est une vue (***View***), elle peut se gérer dans le ***Layout*** de l'application qui permet de décrire en XML la présentation de l'application Android.

<exemple de layout>

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res
/android"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent">
<com.google.android.maps.MapView   android:id="@+id
/map" android:layout_width="fill_parent"
 android:layout_height="fill_parent" android:enabled="true"
android:clickable="true"
 android:apiKey="4lszm-aa1HpD4ajxlùE2dlzleqKLm" />
<TextView android:layout_width="fill_parent" android:id="@+id
/textInfo"
     android:textColor="#ffffff" android:visibility="invisible"
     android:layout_alignParentLeft="true" android:background
="#aaaaaa"
     android:textSize="14px" android:gravity="center_vertical
|center_horizontal"
     android:layout_height="40px"/>
</FrameLayout>
```

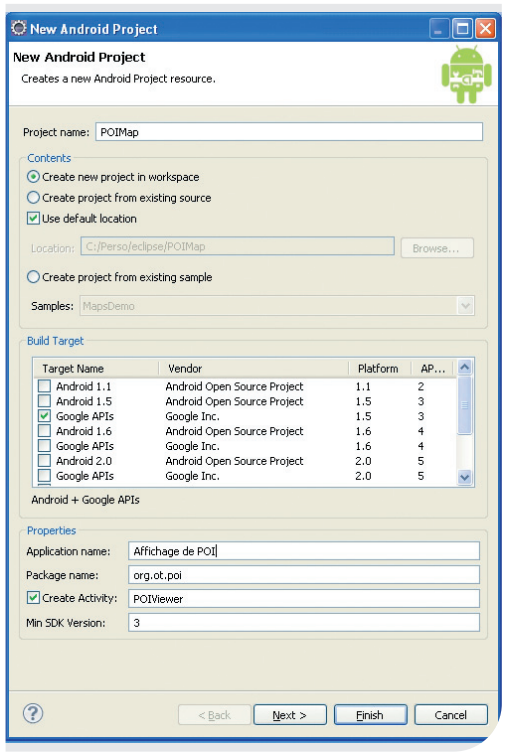
La ***MapView*** gère la visualisation des cartes Google sur l'écran du mobile, elle va offrir une interface de contrôle : le ***MapController*** qui va permettre de contrôler la visualisation :
• La position : ***mapController.setCenter(point)***;
• Le déplacement vers un point : ***mapController.animateTo(point)***;
• Le zoom : ***mapController .setZoom(zoomLevel)***;
Le contrôle du zoom peut s'effectuer via un objet qui va utiliser cette API et qui est placé au-dessus de la vue ou grâce à l'objet en standard positionné par l'appel à la méthode : **setBuiltInZoomControls()** de la Map.

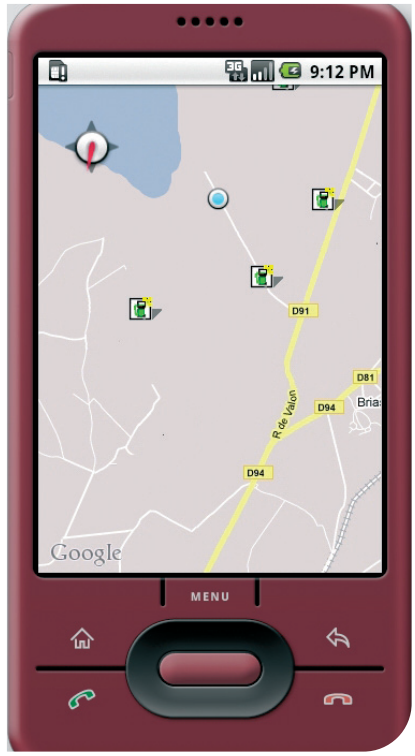
L'affichage de positions grâce aux Overlays

Les ***Overlays*** sont la clé de la mise en œuvre des fonctionnalités de la plateforme. L'***Overlay*** est un calque que l'on applique sur la carte et sur lequel on dessine les marqueurs de position que l'on souhaite faire apparaître.
L'API possède de base deux spécialisations de la classe ***Overlay*** qui couvrent les cas les plus courants d'utilisation :
• ***MyLocationOverlay*** qui gère la location du GPS du mobile
• ***ItemizedOverlay*** qui gère la position de plusieurs points.
MyLocationOverlay permet de s'affranchir de la gestion du GPS du téléphone. Elle prend en charge la gestion de l'affichage de la position, de la précision liée à celle-ci et de la boussole. Elle gère les différents ***Provider*** du téléphone.
La classe offre la possibilité de récupérer les informations liées à la localisation. Cette récupération s'effectue soit de façon procédurale par les méthodes : ***getMyLocation*** et ***getOrientation***. Soit de façon événementielle en surchargeant les méthodes : ***onLocationChanged()***, ***onAccuracyChanged()*** ou ***onSensorChanged()***, ou en utilisant ***RunOnFirstFix()*** pour avoir la première position.
La gestion de la location doit se faire lors des événements pause/resume de la ***MapActivy.***

<exemple de MapActivity>

```
import java.io.IOException;...

public class POIActivity extends MapActivity {
 private MapView map;
```





```
private MapController mapController;
private MyLocationOverlay myLocationOverlay;
private List<Overlay> list;


@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  this.setContentView(R.layout.main);
  map = (MapView) findViewById(R.id.map);
  mapController = map.getController();
  map.setBuiltInZoomControls(true);
  myLocationOverlay.runOnFirstFix(new Runnable() {
    public void run() {
      POIActivity.this.handler.post(new Runnable() {
      public void run() {
        Toast.makeText(POIActivity.this, "Vous êtes localisé",
10).show();
      }
      });
    }
  });
  list = map.getOverlays();
  list.add(myLocationOverlay);
  ...
}

@Override
public void onResume() {
  super.onResume();
  myLocationOverlay.enableCompass();
  myLocationOverlay.enableMyLocation();
}

@Override
public void onPause() {
  super.onPause();
  myLocationOverlay.disableCompass();
  myLocationOverlay.disableMyLocation();
}
)
```

La classe ***ItemizedOverlay*** permet d'afficher des points d'intérêt (POI). Cette classe affiche une liste de points (***OverlayItem***) qu'elle peut associer à une icône. L'exemple permet de créer une ***ItemizedOverlay*** à partir d'un fichier, en voici les étapes importantes :

- Le chargement de la base de points, la méthode ***init*** charge une ressource « raw » contenant les points.
- La définition des méthodes ***size()*** et ***createItem()*** (la classe ***ItemizedOverlay*** est abstraite), qui permettent d'accéder aux ***OverlayItems*** que la classe gère.
- L'appel de la méthode ***populate()*** qui déclenche les appels à size() puis aux createItem(). La méthode populate() peut être longue, c'est pourquoi il peut être intéressant de l'appeler dans un Thread.
- Une étape indispensable est de définir la façon de positionner les icônes par rapport au point sur la carte. Pour cela on utilise **boundCenterBottom() pour positionner le milieu du bas de l'icône sur le point de la carte**.

<exemple d'ItemizedOverlay>

```
import java.io.BufferedReader;...

public class POIOverlayArticle extends ItemizedOverlay<Over
layItem> implements
   Runnable {
 private ArrayList<OverlayItem> points;
 private AlertDialog alertDialog;
 private Handler handler = new Handler();
 private Geocoder geocoder;
 private Context context;
 private OverlayItem selectedPOI;
 private ProgressDialog progressDialog;
 private MapView map;
 private boolean populated = false;

 public POIOverlayArticle(Context context, MapView map, Drawable
 icon) {
   super(icon);
   this.points = new ArrayList<OverlayItem>();
   this.context = context;
   this.map = map;
   geocoder = new Geocoder(context);
   AlertDialog.Builder alertBuilder = new AlertDialog.Builder
(context);
   alertDialog = alertBuilder.create();
   progressDialog = ProgressDialog.show(context, "Chargement
 en cours...",
       "POIs", true);

   boundCenterBottom(points.getIcon());

   new Thread(this).start();

 }

 @Override
 public void draw(Canvas canvas, MapView mapView, boolean shadow) {
   if (!populated)
     return;
   super.draw(canvas, mapView, shadow);
 }

 @Override
 public int size() {
   return (points == null) ? 0 : points.size();
 }

 @Override
 protected OverlayItem createItem(int i) {
   return points.get(i);
 }

 @Override
 protected boolean onTap(int index) {
   selectedPOI = createItem(index);
   handler.post(new Runnable() {
     public void run() {
```

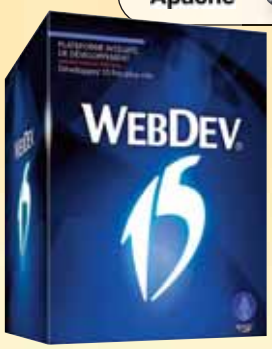
DÉVELOPPEZ
10 FOIS
PLUS VITE

CRÉEZ 10X+ VITE VOS SITES DYNAMIQUES AVEC WEBDEV® 15

EXEMPLES DE SITES EN WEBDEV®

Parmi les **dizaines de milliers** de sites réalisés en WEBDEV, en voici quelques-uns. Retrouvez-en plus de 1000 sur www.pcsoft.fr.

Apache php



Compatible WINDEV et WINDEV Mobile

WEBDEV® 15

ENVIRONNEMENT PROFESSIONNEL DE DÉVELOPPEMENT (AGL)

Créez des sites dynamiques (sites temps réel, reliés à des bases de données) Internet, Intranet et SaaS.

PHP, J2EE, Webservices, XML, Ajax, Linux, Windows...

www.pcsoft.fr

Demandez votre dossier gratuit 200 pages Tél: **04.67.032.032** info@pcsoft.fr

```
        POIOverlayArticle.this.showText(selectedPOI.getPoint(),
            selectedPOI.getTitle());
      }
    });
    return false;
  }

  public synchronized void showText(GeoPoint point, String val) {
    String add = null;
    try {
      List<Address> address = geocoder.getFromLocation(point
          .getLatitudeE6() / 1E6, point.getLongitudeE6() / 1E6, 1);
      Address ad = address.get(0);
      add = "";
      for (int i = 0; i < ad.getMaxAddressLineIndex(); i++)
        add += ad.getAddressLine(i) + " ";
    } catch (Exception ex) {
    }
    if (add == null)
      add = "Pas d'adresse";
    alertDialog.setMessage(add);
    alertDialog.show();
    new Thread(new Runnable() {
      public void run() {
        try {
          Thread.sleep(10000);
        } catch (Exception ex) {

        }
        alertDialog.dismiss();
      }
    }).start();

  }

  @Override
  public void run() {
    init();
    populate();
    populated = true;
    map.postInvalidate();
    progressDialog.dismiss();
  }

  public void init() {
    InputStream is = context.getResources().openRawResource
(R.raw.pois);
    try {
      BufferedReader rd = new BufferedReader(new InputStream
Reader(is));
      String s;
      float latitude;
      float longitude;
      while ((s = rd.readLine()) != null) {
        int i = s.indexOf(',');
        if (i > 2) {
          longitude = Float.valueOf(s.substring(0, i - 1));
          s = s.substring(i + 1);
          i = s.indexOf(',');
          if (i > 2) {
            latitude = Float.valueOf(s.substring(0, i - 1));
            final String name = s.substring(i + 1);
            OverlayItem item = new OverlayItem(
                new GeoPoint((int) (latitude * 1E6),
                    (int) (longitude * 1E6)), name, "");
            points.add(item);
          }

        }
      }
    } catch (Exception ex) {
    }
  }

}
```

Des add-ons qui font la différence

L'affichage des cartes n'est pas la seule fonctionnalité de la plate-forme Android couramment utilisée par les applications devant faire face à des questions de localisation.

Le calcul de distance entre deux points d'une carte est un exercice complexe pour tous les programmeurs. Sur Android, cette fonction est disponible en standard par la classe ***Location***, la classe possède des méthodes pour calculer la distance soit entre deux points arbitraires ***distanceBetween()***, soit à partir d'une ***Location*** connue ***distanceTo(). Location*** offre également la possibilité de déterminer le cap entre deux positions ***bearingTo().***

Une autre classe très utile est le ***Geocoder*** qui permet de faire le lien entre une position et une adresse.

La première opération est offerte par la méthode ***getFromLocation()*** qui retourne une liste d'***Address*** contenant les informations connues sur la position. La seconde opération est fournie par la méthode ***getFromLocationName()*** qui retourne les coordonnées d'un lieu.

Une solution complète pour ajouter la localisation aux applications

Les API Android pour la localisation offrent toutes les fonctionnalités pour une application souhaitant interagir avec une carte et l'application géolocalisée est à coup sûr une « killer app » du Google Market. La seule fonctionnalité manquante est la navigation, pour laquelle la solution est d'invoquer l'application « Google Maps Navigation ». Avec ces API, Android offre des fonctionnalités supérieures aux GPS généralistes et est promis à un bel avenir.

■ Olivier **Théry**

Où suis-je ? Utiliser Google Maps avec moins de 20 lignes de code

Google Maps propose une famille d'API et de services web permettant de réaliser de multiples actions : Intégrer (JavaScript, Flash) une carte "Google Maps" dans son application web, afficher une carte en tant qu'image, rajouter des informations, géolocaliser un utilisateur, etc.

L'une de ces API "Google Maps JavaScript" en est à sa version 3 qui propose une amélioration permettant de faciliter l'intégration sur les téléphones portables. Cette nouvelle version vient de sortir dans le courant de l'année et supprime l'obligation d'avoir une clé à générer pour l'application comme demandé pour les versions précédentes.
Nous proposons de nous appuyer sur cette API afin d'implémenter une application. Cette application permettra à l'utilisateur de savoir où il est et d'afficher les informations disponibles sur sa position : les coordonnées GPS, l'adresse postale (avec l'API Geocoder), et l'altitude (avec l'API Elevation) …

Affichage de la carte

Commençons par la création d'une simple page HTML. Il faut ensuite charger le script Javascript contenant tout le code de l'API Google Maps en ajoutant la ligne suivante dans l'en-tête HTML de notre page :

```
<script type="text/javascript"
  src="http://maps.google.com/maps/api/js?sensor=true">
</script>
```

L'option sensor=true indique que l'application utilise un capteur (par exemple, une puce GPS) pour géolocaliser l'utilisateur.
Dans un premier temps, nous nous contentons d'afficher la carte centrée sur Paris :

```
<script type="text/javascript">
  function initialisation() {
    //Coordonnées de la France
    var latlng = new google.maps.LatLng(46.75984, 1.738281);
    var mesOptions = {
      zoom: 5, //Zoom de hauteur 5 : niveau pays (compris entre
1 et 22)
      center: latlng, //On centre la carte sur les coordonnées
de la France
      mapTypeId: google.maps.MapTypeId.ROADMAP //Type de carte
avec les routes
    };
    //Création de la carte définie par mesOptions et qui sera
placée dans le conteneur HTML identifié comme "conteneur_carte"
    var carte = new google.maps.Map(document.getElementById
("conteneur_carte"), mesOptions);
  }
</script>
```

Explicitons un peu ce code d'initialisation. Nous commençons par créer une variable qui contient les coordonnées géographiques de la France (centre de la France). Nous avons trouvé ces coordonnées sur Google Maps en cherchant "France" et en utilisant l'option disponible dans Google Maps Labs qui affiche dans une info-bulle les coordonnées géographiques au survol de la souris.
Ensuite, on construit un tableau d'options que l'on donnera à Google Maps. Dans ce tableau, on renseigne le niveau de zoom. Un zoom de 1 affiche le monde entier, et peut aller jusqu'à un niveau de 20 à 22 en fonction de la précision disponible à un point donné de la terre. L'option "center" permet de donner les coordonnées du centre de la carte, nous lui donnons la variable avec les coordonnées de la France. Enfin, on demande d'afficher la carte de type "ROADMAP", c'est-à-dire les cartes routières. En mettant "SATELLITE", on affichera les cartes satellite ; et en mettant "HYBRID", on superposera les deux.
La méthode initialisation est appelée au chargement dans le onLoad de la balise HTML body. Il est nécessaire aussi de créer une balise div qui va contenir la carte. C'est ce conteneur que l'on a récupéré par le document.getElementById("conteneur_carte")

```
<body onload="initialisation()">
  <div id="conteneur_carte" style="width:100%; height:100%">
</div>
</body>
```

Pour que le marqueur et la bulle d'information s'affichent dans des dimensions lisibles sur les navigateurs mobiles, il est nécessaire d'ajouter la balise suivante :

```
<meta name="viewport"
  content="width=device-width; initial-scale=1.0; maximum-scale
=1.0; user-scalable=0;" />
```

Cette balise permet de dire aux navigateurs mobiles que ce site est adapté à un écran mobile

Géolocalisation

Ajoutons maintenant la géolocalisation :

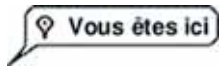
```
<script type="text/javascript">
  var carte;
  var latlng;

  //Cette fonction initialise la latitude et la longitude en
```

```
fonction d'une position
 //puis centre la carte sur cette position
 function afficheOuJeSuis(position) {
    latlng = new google.maps.LatLng(position.coords.latitude,
position.coords.longitude);
   carte.setCenter(latlng); //Centre la carte sur la position
   carte.setZoom(14); // Zoom sur la position
 }

 function erreurAfficheOuJeSuis() {
   alert("Geolocation a échoué.");
 }

 function initialisation() {
   latlng = new google.maps.LatLng(46.75984, 1.738281);
   var mesOptions = {
     zoom: 5,
     center: latlng,
     mapTypeId: google.maps.MapTypeId.ROADMAP
   };
   carte = new google.maps.Map(document.getElementById("conteneur
_carte"), mesOptions);
   navigator.geolocation.getCurrentPosition(afficheOuJeSuis,
erreurAfficheOuJeSuis);
 }
</script>
```

Nous commençons par créer deux fonctions : afficheOuJeSuis() et erreurAfficheOuJeSuis(). La première fonction est appelée lorsque le navigateur réussit à trouver la position de l'utilisateur. La deuxième fonction affiche une alerte JavaScript indiquant que la géolocalisation a échoué.
Dans la fonction afficheOuJeSuis(), nous centrons la carte sur les coordonnées de l'utilisateur que l'on récupère par le paramètre "position". Nous avons la latitude dans position.coords.latitude et la longitude dans position.coords.longitude.
Dans la fonction initialisation(), nous avons seulement ajouté une ligne, qui va appeler la méthode du navigateur getCurrentPosition(). Cette méthode lance la recherche de la localisation sur le navigateur. Si c'est la première visite sur le site, le navigateur va demander à l'utilisateur s'il accepte de partager sa localisation. A cette fonction, nous donnons en paramètre les deux fonctions créées précédemment, pour lui dire de les appeler s'il a trouvé la position ou s'il y a une erreur. La recherche de la position prend du temps. C'est pourquoi nous commençons par afficher la carte sur la France avant de zoomer sur la position dès qu'elle est trouvée.
Notre code est simplifié car nous utilisons l'objet navigator.geolocation. Mais seuls les navigateurs les plus récents, supportant l'HTML 5 (Firefox 3, Chrome 5, iPhone, Android 2), possèdent cet objet. Pour les navigateurs plus anciens (Internet Explorer par exemple), cette fonctionnalité n'est pas disponible.
Puis marquons notre position sur la carte :

```
function afficheOuJeSuis(position) {
 latlng = new google.maps.LatLng(position.coords.latitude,
position.coords.longitude);
 carte.setCenter(latlng);
 var marqueur = new google.maps.Marker({
   position: latlng, //Position du marqueur
   map: carte, //Carte sur laquelle apparaît le marqueur
   "http://chart.apis.google.com/chart?chst=d_bubble_icon_
text_small&chld=location|bb|Vous%20%C3%AAtes%20ici|FFFFFF|0
00000", //Icône du marqueur
   title:"Je suis ici !" //Texte pour le rollover
 });
}
```

Nous créons donc un marqueur en lui indiquant la position géographique, la carte où l'afficher et l'URL de l'image à afficher :



Affichons maintenant, une info bulle lorsque l'on clique sur cette icône :

```
var bulleInfo = new google.maps.InfoWindow();
//Affiche le contenu initial de la popup d'information
bulleInfo.setContent('Latitude et longitude : ' + latlng);
//Ecoute le clic sur le marqueur pour afficher l'info bulle
google.maps.event.addListener(marqueur, 'click', function() {
bulleInfo.open(carte,marqueur);
});
```

Nous créons une infoWindow ayant pour contenu la latitude et la longitude. Puis nous ajoutons un listener sur le marqueur pour ouvrir la bulle lors du clic sur celui-ci [Fig.1].

Service Geocoder et Elevation

Nous souhaitons maintenant afficher l'adresse de notre position. Pour cela nous allons utiliser le service Geocoder qui permet de retrouver une adresse postale à partir de coordonnées GPS.

```
//Service de conversion entre adresse et position

var geocoder = new google.maps.Geocoder();

//Retrouve une adresse à partir de sa position
geocoder.geocode({'latLng': latlng}, function(resultats, statut) {
 if (statut == google.maps.GeocoderStatus.OK) {
   if (resultats[0]) {
   carte.setZoom(14);
   bulleInfo.setContent(bulleInfo.getContent() + '<br/>Adresse :
' + resultats[0].formatted_address);
   } else {
   bulleInfo.setContent(bulleInfo.getContent() + "<br/>Nous
n'avons pas trouvé cette adresse");
   }
 } else {
   alert("Le service Geocoder a rencontré une erreur : " + statut);
 }
});

//Affichage de la bulle d'information reprenant vos informations
de localisation
google.maps.event.addListener(marqueur, 'click', function() {
  bulleInfo.open(carte,marqueur);
});
```

Nous créons une instance du service geocoder dont nous appelons ensuite la fonction geocode avec deux paramètres. Le premier paramètre est un tableau d'options qui contient uniquement l'option "latLng" qui renseigne les coordonnées géographiques. Le second paramètre est la fonction qui sera appelée lorsque le service aura trouvé l'adresse. Dans cette fonction, nous commençons par vérifier l'état du résultat, en vérifiant que le statut de la réponse est google.maps.GeocoderStatus.OK. Nous retrouvons alors dans la variable resultats[0].formatted_address l'adresse complète que nous cherchons. Le résultat fournit beaucoup plus d'informations, comme une décomposition complète de l'adresse, la précision. La réponse du service n'est pas forcement unique. Il peut y avoir plusieurs résultats. Si le tableau des résultats ne possède aucun élément, alors aucune adresse n'a été trouvée.
Pour finir, nous souhaitons connaître l'altitude de notre position. Pour cela, nous utilisons le service ElevationService :

```
//Service permettant de trouver l'altitude d'une position
var altitude = new google.maps.ElevationService();

//Retrouve l'altitude d'une position donnée
function AQuelAltitudeJeSuis(latlng) {
 altitude.getElevationForLocations({'locations': [latlng]},
function(resultats, statut) {
   if (statut == google.maps.ElevationStatus.OK) {
   if (resultats[0]) {
   bulleInfo.setContent(bulleInfo.getContent()  +  "<br/>
L'altitude est de " + resultats[0].elevation + " mètres.");
   } else {
   bulleInfo.setContent(bulleInfo.getContent() + "<br />Pas
d'altitude trouvée");
   }
   } else {
   alert("Le service Elevation a rencontré une erreur : " +
statut);
   }
 });
}
```

Ce service fonctionne de la même façon que le service geocoder, et nous permet de récupérer l'altitude dans la variable resultats[0].elevation [Fig.2].

Conclusion

On a pu voir que l'utilisation des API Google Maps est simple, et très rapidement nous avons pu construire une application fonctionnelle. Vous pouvez consulter la page à partir de votre téléphone mobile à l'adresse suivante :
http://ousuisje.googlecode.com/svn/trunk/index.html
et le code source est disponible ici : http://code.google.com/p/ousuisje/

■ Pierre **Mage** et Patrice **de Saint Steban**
et une équipe d'ingénieurs de chez SFEIR
http://www.insideit.fr

Mélanger Bing Maps et Windows Azure

A travers cet article, nous allons aborder l'utilisation de la plateforme Windows Azure pour héberger des données géographiques qui seront ensuite exposées à travers un Web Service exécuté au sein de Windows Azure. Nous utiliserons ensuite ces données au sein du contrôle Silverlight de Bing Maps for Enterprise. Enfin, nous réaliserons une application de type Bing Maps App qui sera alors chargée sur le portail Bing Maps dédié, celle-ci utilisera pleinement le scénario simple mis en œuvre.

LA PLATEFORME WINDOWS AZURE : SQL AZURE ET SERVICE HÉBERGÉ

Nouveautés et rappels

En fin de mois de juin, Microsoft a annoncé le support des types géométriques (GEOMETRY) et géographiques (GEOGRAPHY) au sein de son offre SQL Azure, en plus du type hiérarchique (Hierarchyld). Avec cette nouvelle fonctionnalité, c'est une grande partie de la puissance des nouveautés de SQL Server 2008 qui est mise à disposition dans une base de données hébergée dans le Cloud. On retrouve notamment la possibilité d'utiliser les fonctions avancées de SQL dédiées à ce genre de requête.

Intérêts

Dans les scénarii classiques d'utilisation et de justification de l'emploi des technologies « On the Cloud », on retrouve principalement le besoin en performance et en adaptabilité des ressources nécessaires. Bien entendu, les principaux avantages d'utilisation de la plateforme SQL Azure résident dans l'architecture sur laquelle s'appuie l'ensemble de la plateforme permettant notamment de profiter d'une haute disponibilité, des possibilités d'évolutivité et de gestion simplifiée pour utiliser une base de données relationnelle et administrer des serveurs SQL de manière complètement transparente. Il faut également préciser que dans le cas de l'application Silverlight personnalisée, on choisit d'utiliser un Web Role pour exposer nos données au travers d'un Web Service et d'héberger notre application Silverlight sur Internet. Ces instances d'exécution pourront être dimensionnées en nombre ou en puissance en fonction du besoin pour répondre à la charge, sans que l'on soit réellement confronté aux problèmes de mise en œuvre en termes de matériels ou de systèmes.

Mise en œuvre de SQL Azure

Pour pouvoir exploiter les fonctionnalités de la plateforme Windows Azure, il est nécessaire d'utiliser un compte adapté. Pour cela, plusieurs solutions sont possibles, il existe une offre pour les abonnés MSDN ou pour les développeurs, ce qui permet de réaliser les développements dans un réel scénario et de tester les applications réalisées.

Création de la base de données

Afin de créer la base de données, il suffit d'utiliser l'interface en ligne dédiée disponible à l'adresse suivante : https://sql.azure.com/
Ensuite, en cliquant sur « Create a database », il suffit de renseigner les informations sur la base de données à créer. On y spécifie alors le nom ainsi que le choix de l'édition dépendant de la facturation et la taille de la base en elle-même (jusqu'à 50 Go en édition Business) : [Fig.1]. La base de données ainsi créée devient alors listée dans l'interface : [Fig.2]. Il est également nécessaire de configurer correctement la connexion à la base de données en définissant les plages d'adresse IP autorisées à se connecter, toujours en utilisant l'interface dédiée : [Fig.3].

Connexion à la base

Avant de se connecter à la base, dans l'interface d'administration, on récupère la chaîne de connexion qu'il nous faudra ensuite compléter pour inclure le mot de passe. Cette chaîne sera utilisée dans le développement de la partie exposition des données qui suit : [Fig.4]. Il est possible de se connecter à la base à l'aide de SQL Management Studio 2008 R2 ou directement depuis Visual Studio 2010, ce qui permet notamment d'exécuter des requêtes avec des outils adaptés.

...suite de cet article p.91 →

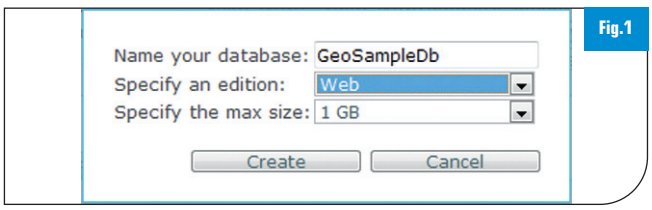


Fig.1

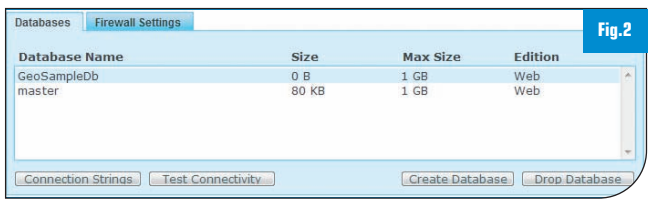


Fig.2

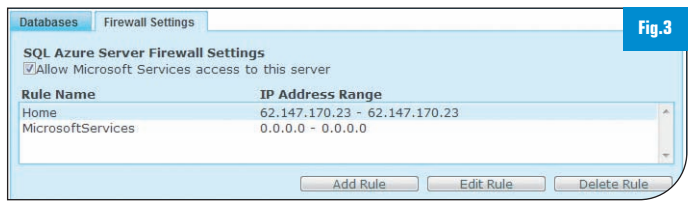


Fig.3

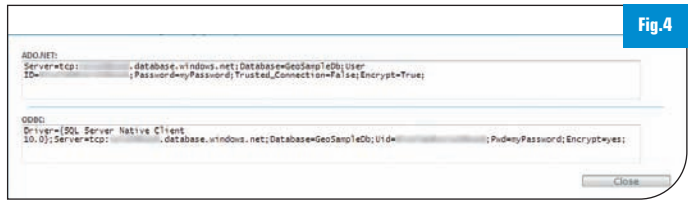


Fig.4

Supplément Spécial détachable

Le magazine du développement

PROgrammez !

www.programmez.com

Tout savoir sur Windows Phone 7

Windows® Phone

Découvrez

La nouvelle plateforme mobile de Microsoft !

Bien démarrer avec Windows Phone 7 : les outils, les bonnes pratiques...

Interface

Metro : la révolution de l'interface mobile !

Créer son contrôle personnalisé

Jeux

Créez vos jeux 2D et 3D avec XNA !

Code

Migrez vos codes Android, Windows Phone 6.5, iPhone

MarketPlace

Distribuez vos applications

© Microsoft

EDITO

Tout pour plaire !

Windows Phone 7 arrive bientôt. Convivialité, interface naturelle, modèle de développement unifié grâce à .net, Visual Studio, Silverlight et XNA, intégration de Zune, la nouvelle plate-forme mobile de Microsoft est LA révolution que l'on attendait pour Windows Phone.
En 3 ans, le paysage des Smartphones a considérablement changé avec iPhone et Android.

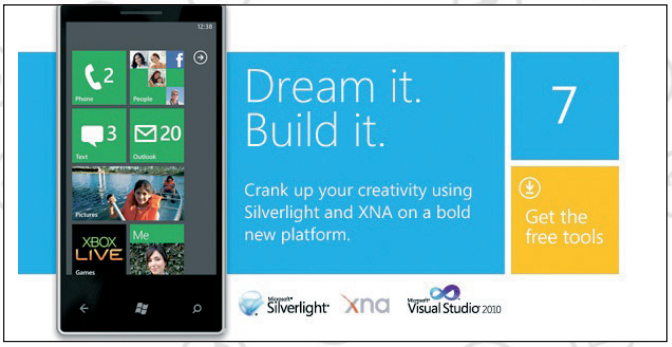
Aujourd'hui, le succès grandissant du Smartphone est intimement lié aux applications. Les développeurs, donc vous, jouent un rôle crucial. Ce cahier spécial Windows Phone 7 (WP7) est fait pour vous. Il vous permettra de découvrir, comprendre, apprivoiser les nouvelles fonctions, Metro, Silverlight, les bonnes pratiques. C'est un nouveau marché qui s'ouvre à vous avec le Marketplace de WP7 ! Surtout, le cahier des charges imposé aux constructeurs est très strict et identique pour tous, limitant les risques de fragmentation du marché.

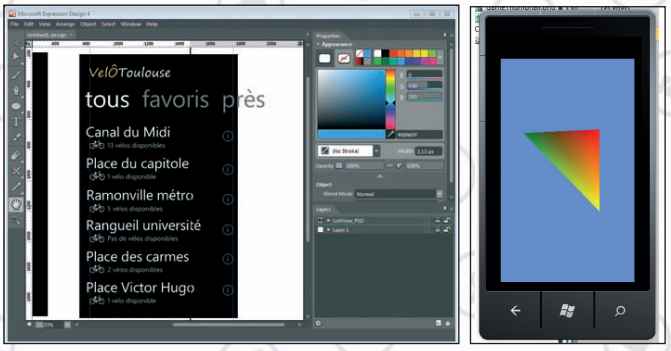
Dans les prochaines semaines, les téléphones Windows Phone 7 vont se multiplier, profitez de cette éclosion dès aujourd'hui.

A vous de jouer !

François Tonic
Rédacteur en chef

SOMMAIRE





Le magazine du développement PROgrammez! www.programmez.com

Ce cahier détachable de PROGRAMMEZ N°134 ne peut être vendu séparément

Experts : J. Corioland, T. Lebrun, C. Maneu, P. Cauchois, J. Dollon, S. Ferquel, G. Mazarico, V. Billotte, S. Sibué, S. Yriarte, F. Decret, P-Y. Gardette, A. Markarian

La nouvelle plate-forme mobile Windows : personnelle, pertinente, connectée

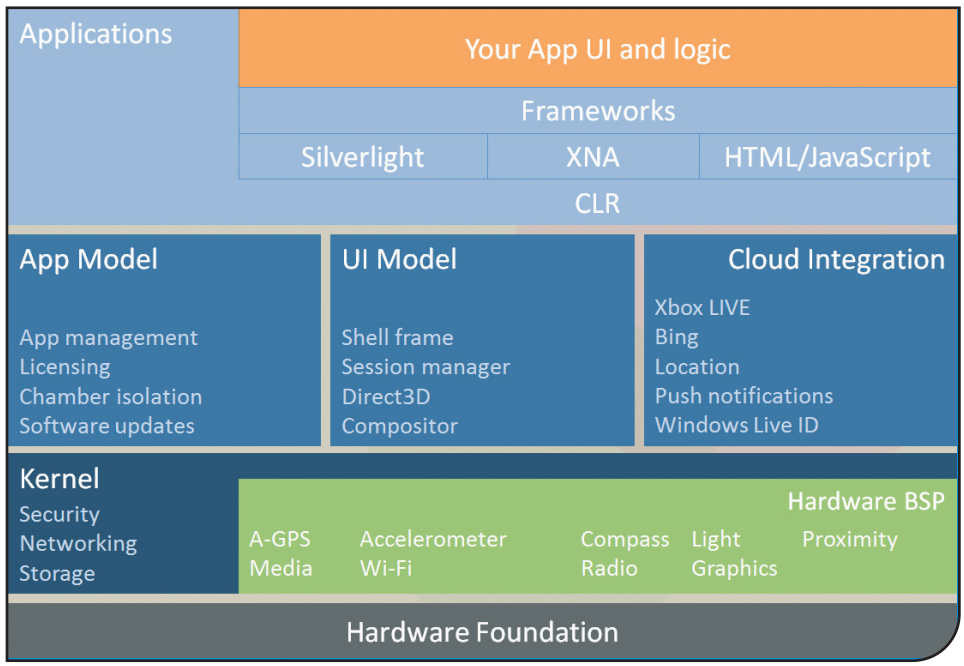
Nous y sommes. Après plusieurs mois de développement, de tests, de retour terrain, Microsoft lance Windows Phone 7 (WP7). Plus qu'un simple numéro, 7 rappelant Windows 7, WP7 renouvelle de fond en comble la mobilité Windows. Pour le bonheur du développeur mais aussi des utilisateurs.

Une architecture renouvelée, resserrée, plus performante !

Cohérence est l'un des maîtres mots de WP7. Jusqu'à présent, créer une interface mobile, et une application, se révélait fastidieux, long et pour un résultat souvent jugé médiocre. Pour WP7, l'expérience utilisateur a été mise en avant. Cela signifie une refonte totale de l'interface utilisateur de Windows Phone : plus sobre, plus fonctionnelle. Fini le portage tel quel de l'interface Windows qui n'a pas de sens sur un Smartphone ou sur une tablette tactile.

D'autre part, cette refonte s'accompagne d'une purge drastique du modèle de développement : Visual Studio, Silverlight, XNA, Expression Blend. Voilà pour l'essentiel des outils nécessaires. Et surtout, les API ont été réduites. Le but : cohérence de la plate-forme. Avec à la clé, une intégration avec les dernières versions de Silverlight et de Visual Studio. Ainsi, il est possible de démarrer un développement WP7 sans débourser un centime. Il suffit d'installer Visual Studio Express incluant le kit de développement WP7 ! Un grand changement par rapport aux versions précédentes qui nécessitaient un Visual Studio professionnel…

Ainsi WP7 avec des spécifications plus précises, plus strictes doit assurer une meilleure cohérence matérielle. Cela signifie que les constructeurs de téléphone suivent un cahier des charges strict. Cette rigueur permettra d'éviter une fragmentation trop grande du marché et une disparité entre les modèles. Plus une plateforme est homogène, mieux les applications mobiles fonctionneront. C'est l'un des objectifs cruciaux de WP7.



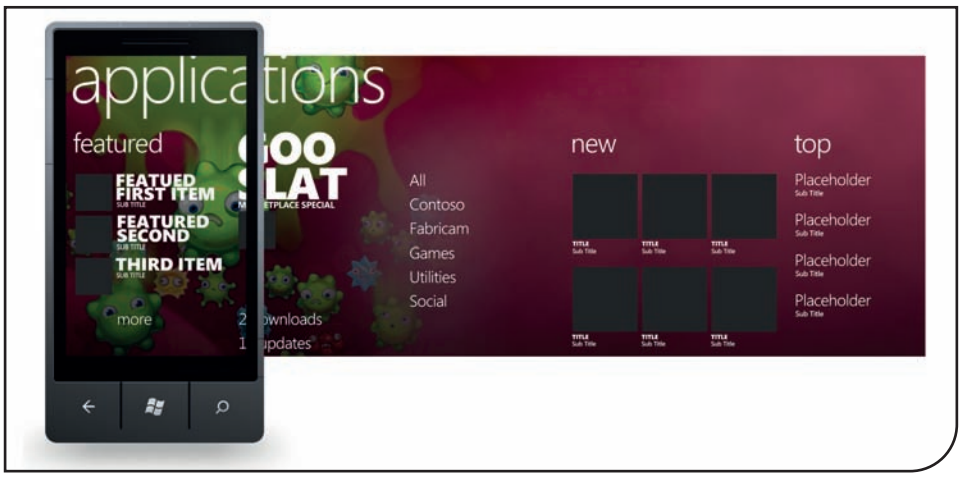
Une nouvelle interface, un nouveau langage : Metro

L'interface utilisateur de WP7, et donc des applications mobiles, repose sur Metro, qui se décrit comme un langage de design. Cela signifie qu'il définit l'ensemble de l'interface, l'ergonomie, les interactions, les bonnes pratiques. Metro n'est pas obligatoire, cependant il est fortement recommandé de l'utiliser. On dispose ainsi des codes couleurs, comment placer les boutons, quel type d'animation il faut à un endroit précis, comment se déroule une transition entre deux fenêtres, etc.

Metro est une contrainte dans le sens, ou le développeur doit apprendre, assimiler de nouveaux concepts, une nouvelle interface mais c'est tout bénéfice pour lui car son application sera parfaitement intégrée à WP7 ! On ne fait pas autrement sur iPhone ou Android.

Au cœur du système

Le système d'exploitation est une base Windows Embedded modifiée en profondeur dans laquelle Microsoft a la responsabilité de la très grande majorité du code, y compris des drivers. L'intégration qui est laissée au fabricant du terminal est la couche basse des drivers, celle directement en contact avec le silicium, qui peut varier d'un téléphone à l'autre (par exemple, tous les fabricants n'utilisent pas la même référence de capteur de mouvement : pourtant, vu du système il doit s'adresser de la même manière). Cela permet d'avoir une API unifiée pour les capteurs, mais aussi pour les couches radios, graphiques, multimédia, etc. Au-dessus de ce noyau (un véritable OS à part entière sans base de code commune avec Windows, avec gestion du paging de la mémoire, de la sécurité), on trouve le modèle d'application, avec la gestion des sandbox, des mises à jour, des installations/désinstallations et des licences des applications, etc.



On trouve également le shell et le modèle graphique, avec notamment un modèle de composition spécifique qui permet d'avoir **une navigation intuitive, cohérente, et surtout, accélérée graphiquement !** Enfin, dernier composant de la couche applicative de l'OS, l'intégration avec les services dans le cloud, notamment les notifications en push, la géolocalisation, l'intégration avec les Windows Live ID, Bing, et le Xbox Live. Enfin, au-dessus de cela, on retrouve la couche applicative avec les deux frameworks de développement, XNA et Silverlight, et bien entendu les applications que l'utilisateur aura téléchargées sur Marketplace.

Sans rentrer dans les détails du système d'exploitation, on peut dire que l'OS est un système temps-réel dur, entièrement multi-tâche, avec un modèle de sécurité, un modèle de drivers et une gestion de la mémoire qui lui sont propres et qui ont complètement été modifiés par rapport à Windows Embedded CE 5.0, qui servait de base pour Windows Mobile 6.x. La limitation à 32 processus n'existe plus, pas plus que la limitation de 32Mo de mémoire par processus. Les espaces kernel et user sont séparés et les drivers peuvent être isolés les uns des autres, procurant ainsi au système une modularité et une indépendance maximale entre les composants, ce qui réduit l'impact en cas de bug ou de faille de sécurité éventuelle.

Windows Phone 7 : pour qui ?

La nouvelle plate-forme cible à son lancement le grand public et une cible en particulier que l'on appelle en anglais : *life maximizer*. Typiquement : une personne d'environ 35 ans, installée, jonglant entre vie personnelle et professionnelle, ayant un besoin de rester connectée et se servant de son Smartphone comme d'un outil de travail. Tout en l'utilisant avec plaisir… La personne « life maximizer » attend toujours plus de son Smartphone : mail, internet, media, connectivité permanente, échange…

Les adolescents, gros consommateurs de téléphones, jeux et gadgets, sont également ciblés par Windows Phone 7. Et des offres (matériels et forfaits) seront proposées par les opérateurs et constructeurs. Sans oublier, les applications du marketplace !

Côté entreprise, la plate-forme convient parfaitement avec le mail, Exchange et Sharepoint, ce qui représente 80 à 90 % des usages. Et les opérateurs proposeront rapidement des services et forfaits pro. Pour ce qui est des applications métiers, il faudra attendre les prochains mois pour les mises à jour et une plus grande souplesse de déploiement (en dehors du marketplace).

Une riche documentation pour le développeur

L'un des défis de WP7 est de proposer rapidement des milliers d'applications sur le marketplace (boutique en ligne des applications WP7). Avec WP7, que vous soyez développeur web ou C# ou VB, devenir un développeur WP7 n'est pas plus compliqué que de créer une application desktop ! L'intégration des outils et des librairies mobiles permet cela.

Pour faciliter le travail du développeur, une importante documentation technique sera disponible en Français ! Il s'agit là d'un travail immense engagé depuis plus de 6 mois. Et c'est une nouveauté réelle car ni Android, ni iOS ne proposent des ressources techniques en Français… Ainsi que vous soyez développeur amateur ou professionnel, vous aurez la même information. D'autre part, des webcast, des coachs seront disponibles en ligne. Et des sessions seront proposées durant les Microsoft Days et le prochain TechDays (2011).

Microsoft mobilise aussi la communauté Windows Phone, Silverlight, Visual Studio pour stimuler la création de contenu, d'articles techniques autour de WP7 ! Un laboratoire de compatibilité et de tests existe aussi au Microsoft Technical Center de Paris.

■ François **Tonic**

Installation des outils de développement

Le développement Windows Phone 7 se base sur des technologies éprouvées : Silverlight et XNA ainsi que sur un langage ergonomique et graphique nommé Metro. Créons dans cet article un nouveau premier projet. Cela nous permettra de faire le tour des outils qui permettent le développement d'application Windows Phone 7.

Comme pour toutes les technologies de développement proposées par Microsoft, il est possible de développer pour Windows Phone 7 à l'aide de Visual Studio 2010. Afin de pouvoir utiliser ces outils avec Windows Phone 7, vous devez au préalable installer les outils de développement pour cette plateforme. Ceux-ci sont disponibles sur la page suivante : http://developer.windowsphone.com/windows-phone-7/ [Fig.1].

Une fois les outils installés, vous avez deux possibilités : la première est d'utiliser la version de **Visual Studio 2010 Express pour Windows Phone 7** qui vient d'être installée (si vous n'aviez pas d'autre version existante au préalable), la seconde est d'utiliser votre édition de Visual Studio 2010 (Premium, Professional ou Ultimate). En plus d'un environnement de développement, les outils développeurs pour Windows Phone 7 installent les éléments suivants sur votre machine :

- **Le SDK Silverlight pour Windows Phone 7 :** ce kit de développement logiciel vous permettra de créer des applications Silverlight pour la plateforme Windows Phone 7
- **Le XNA Game Studio 4.0 :** il s'agit là du SDK vous permettant de développer des applications XNA (jeux vidéo, par exemple) destinées à être exécutées sur un terminal Windows Phone 7
- Un ensemble de templates de projets pour Visual Studio 2010 vous permettant de commencer à développer plus rapidement
- Un émulateur de terminal Windows Phone 7 vous permettant d'exécuter et déboguer vos applications sans avoir à les déployer continuellement au cours de vos développements. L'émulateur sera détaillé dans la partie suivante.

Votre premier projet…

Lancez votre version de Visual Studio 2010. Dans le menu **File,** choisissez **New** puis **Project…** [Fig.2]. Dans la fenêtre qui s'ouvre, sélectionnez **Silverlight for Windows Phone 7** dans la zone de gauche. Comme vous pouvez le constater, plusieurs modèles de projets sont à votre disposition pour démarrer : [Fig.3].

- **Windows Phone Application :** ce modèle vous permet de créer une application Silverlight classique pour Windows Phone 7
- **Windows Phone Databound Application :** ce modèle vous permet de créer une application Silverlight de type liste pour Windows Phone 7. Il s'agit d'une application dont la première page est composée d'une liste d'éléments permettant à l'utilisateur de naviguer dans l'application en les sélectionnant.
- **Windows Phone Class Library :** ce modèle vous permet de créer une librairie de classe pour Windows Phone 7.

Dans la liste de gauche toujours, une autre section, XNA Game Studio 4.0, vous permet d'accéder à d'autres modèles de projets pour Windows Phone 7 : [Fig.4].

- **Windows Phone Game (4.0) :** ce modèle de projet vous permet de créer un jeu XNA pour Windows Phone 7.
- **Windows Phone Game Library :** ce modèle de projet vous permet de créer une librairie de classe pour vos jeux XNA pour Windows Phone 7.
- **Windows Phone Panorama Application :** ce modèle permet de construire une application se servant d'un panorama
- **Windows Phone Pivot Application :** ce modèle permet de construire une application se servant d'un pivot.

Pour découvrir les outils de développement présents dans Visual Studio, retournez dans la catégorie **Silverlight for Windows Phone 7** et créez un projet de type **Windows Phone Application.**

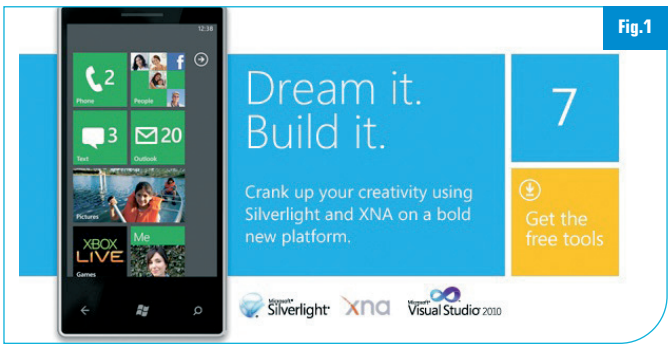


Fig.1

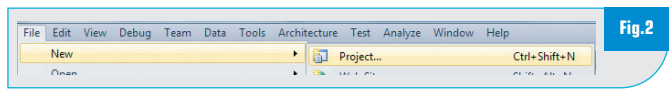


Fig.2

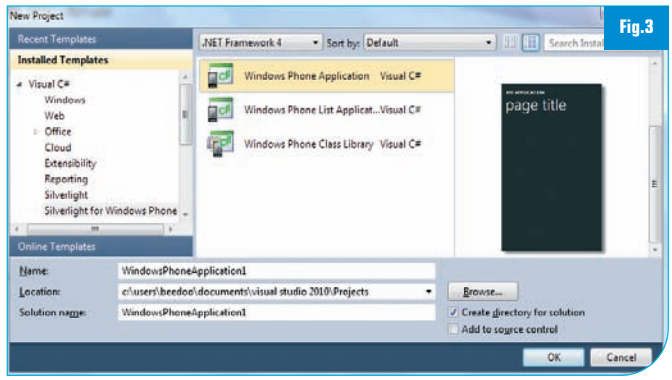


Fig.3

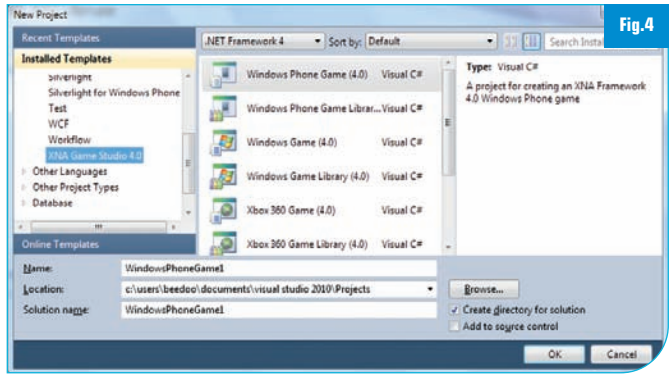


Fig.4

Visual Studio met alors en place toute la structure du projet : [Fig.5]
La liste ci-dessous résume les principaux éléments qui ont été ajoutés à la solution :

- Les références **Microsoft.Phone** et **Microsoft.Phone.Interop** sont les références spécifiques du Framework Silverlight pour Windows Phone 7.
- Les images **ApplicationIcon.png, Background.png** et **SplashScreenImage.jpg** représentent respectivement l'icône de l'application, son fond et son écran d'accueil.
- Les fichiers App.xaml et App.xaml.cs représentent le point d'entrée de l'application. C'est également dans celle-ci que vous pourrez gérer les différents éléments du cycle de vie de celle-ci.
- La page MainPage.xaml constitue la page d'accueil de l'application.

Comme vous avez pu le constater, le fichier MainPage.xaml est directement ouvert dans Visual Studio 2010 et laisse apparaître un nouveau designer : [Fig.6]. Un ensemble de contrôles est également disponible depuis la boîte à outils de Visual Studio 2010 : [Fig.7]. Comme toujours, il vous suffit de glisser / déposer les contrôles depuis celle-ci vers la surface de design afin que ceux-ci soient ajoutés au code XAML de la page.

Découverte de l'émulateur

Afin de pouvoir développer plus simplement et sans avoir nécessairement un terminal Windows Phone 7 sous la main, Microsoft vous propose un émulateur permettant l'exécution et le debug de vos applications, directement depuis Visual Studio 2010. Pour le lancer il vous suffit d'exécuter l'application que vous avez créée dans la partie précédente en vous rendant dans le menu **Debug** puis **Start Debugging** ou encore de presser la touche **F5**. Patientez pendant le chargement de l'émulateur. Cela peut prendre quelques minutes, selon votre machine. En effet, ce qu'il faut savoir c'est qu'il s'agit ici d'une vraie version de Windows Phone 7 s'exécutant dans une machine virtuelle, permettant ainsi d'exécuter les applications sur l'OS destiné à les accueillir ! [Fig.8]
Comme vous pouvez le voir, l'émulateur possède trois boutons :

- Le bouton « back » permettant à l'utilisateur de revenir en arrière lorsqu'il navigue dans ses applications.
- Le bouton « start » permettant à l'utilisateur de revenir à l'écran de démarrage de son téléphone.
- Le bouton « search » permettant à l'utilisateur de lancer une recherche à tout moment.

NB : *vous pouvez et devez prendre en compte ces différents boutons durant vos développements. Tout terminal exécutant Windows Phone 7 les possèdera !*

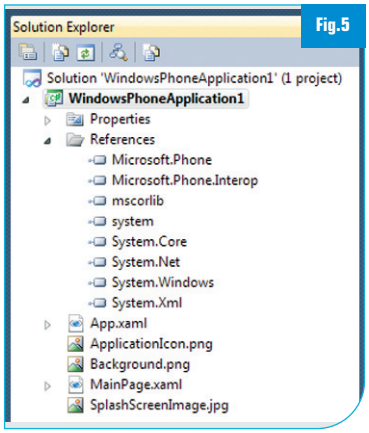
Si vous passez votre souris sur l'émulateur, vous devriez voir apparaître une barre d'outils en haut à droite de celui-ci : [Fig.9]
Voilà les actions des différents icônes, de bas en haut :

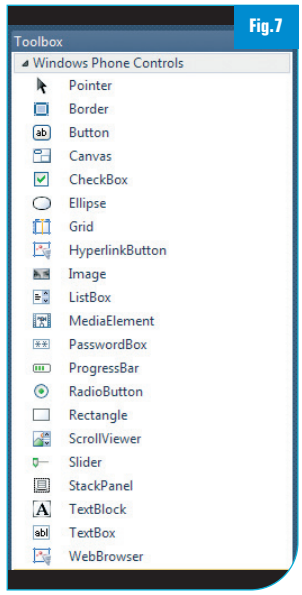
- Fermer l'émulateur
- Réduire l'émulateur
- Faire pivoter l'émulateur vers la gauche : cela permet de gérer cette rotation dans vos applications et ainsi de proposer à vos utilisateurs des pages orientées en portrait, en paysage, ou les deux !
- Faire pivoter l'émulateur vers la droite
- Adapter la taille de l'émulateur à l'écran
- Accéder à une fenêtre permettant de choisir le niveau de zoom de l'émulateur.

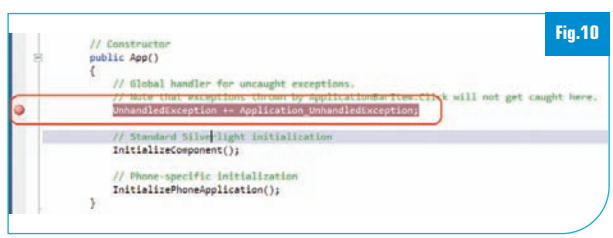
Si vous souhaitez déboguer l'application en pas à pas, il vous suffit de placer un point d'arrêt dans le code source de l'application. Pour cela, stoppez l'exécution de l'application via le menu **Debug** puis **Stop Debugging** de Visual Studio 2010 ou le raccourci clavier **Shift+F5** (il n'est pas utile de fermer l'émulateur).
Ouvrez ensuite le fichier App.xaml.cs depuis l'explorateur de solution et placez un point d'arrêt sur la première ligne du constructeur de la classe : [Fig.10]

NB : *pour placer un point d'arrêt, pressez la touche **F9** après avoir mis le focus sur la ligne souhaitée ou cliquez dans la marge (au niveau du point rouge sur l'image). Relancez l'application. Comme vous pouvez le constater, l'exécution est bloquée au niveau de cette ligne :* [Fig.11]

La fenêtre **Watch 1** (menu **Debug, Windows** puis **Watch**) vous permet de placer des espions sur vos variables afin de vérifier leurs valeurs et la fenêtre **Immediate Window** (menu **Debug, Immediate Window**) vous permet d'exécuter du code lors du debug. Vous pouvez naviguer dans le code via la barre d'outils de debug de Visual Studio 2010 : [Fig.12]







Les 3 boutons ayant une flèche bleue vous permettent de naviguer dans le flux d'exécution : passer dans l'appel, passer à l'appel suivant, remonter à l'appel précédent, respectivement.

Les autres logiciels du développeur

Lorsque vous posséderez un terminal Windows Phone 7, vous pourrez également déployer et débugger vos applications directement sur celui-ci. Pour cela, vous devrez avoir installé le **Zune Software** qui permet de se connecter à un terminal. Ensuite, il vous suffira de choisir l'entrée **Windows Phone 7 Device** dans la barre d'outils d'exécution de Visual Studio 2010 et de lancer l'application : [Fig.13]

Si vous souhaitez partager et vendre vos applications sur le Market Place Windows Phone 7, un outil livré avec les outils développeurs Windows Phone 7 vous permet de vous enregistrer comme développeur Windows Phone 7 auprès de Microsoft. Pour le lancer, ouvrez le menu démarrer et recherchez le logiciel **Windows Phone Developer Registration** puis lancez-le : [Fig.14]

Si un terminal était connecté à votre machine, vous auriez la possibilité de rentrer votre compte Windows Live et de vous enregistrer comme développeur Windows Phone 7 !

Expression Blend pour Windows Phone

Afin de pouvoir réaliser les interfaces graphiques de ses applications Silverlight, il est possible d'utiliser une version spéciale d'Expression Blend : *« Expression Blend for Windows Phone »*. [Fig.15]

Il s'agit d'une version « allégée » d'Expression Blend qui ne peut être utilisée qu'avec les projets Windows Phone. Ainsi, il n'est possible de créer que des projets de type Windows Phone, comme le montre la capture : [Fig.16]. Une fois un projet créé, Expression Blend affiche l'interface graphique que l'on retrouve dans Visual Studio, à savoir une copie d'un téléphone avec une première page déjà développée (de façon simpliste, bien évidemment) : [Fig.17]. Dès lors, toutes les techniques de développement et de design que vous avez pu être en mesure d'acquérir lors de vos développements (glisser/déposer, utilisation des assistants, etc.) pourront être réutilisées dans vos développements Silverlight pour Windows Phone. En effet, les outils sont les mêmes donc les techniques utilisées sont également les mêmes ! Parmi les grandes nouveautés communes à Silverlight « classique » et Silverlight pour Windows Phone, on retrouve les Behaviors, ces fameuses classes que vous pouvez utiliser pour ajouter, sans lignes de code (et donc en un rien de temps), des fonctionnalités à votre application : [Fig.18].

Et l'avantage d'utiliser Expression Blend pour concevoir ses interfaces graphiques, c'est que ces fonctionnalités sont gérées directement ! Ainsi, un simple glisser/déposer d'un Behavior sur un contrôle de celui-ci se retrouve instantanément avec la fonctionnalité correspondante, en un clic de souris !

Si vous voulez concevoir votre application, il vous suffit de faire un ensemble d'opérations bien connues, les fameux « glisser/déposer ». Ainsi, si vous prenez un contrôle qui se situe dans la boîte à outils (par défaut située sur la partie gauche de l'application), il vous suffit de le faire glisser/déposer pour que celui-ci se retrouve dans l'interface graphique de votre application : [Fig.19]

Outre le côté « déjà vu » de ce type de technique (c'est en effet ce qu'il est aussi possible de faire avec Visual Studio), l'un des points forts d'Expression Blend est la réutilisation des styles ! Ainsi, on constate que les contrôles que l'on fait glisser/déposer sur l'interface graphique héritent automatiquement du style sur fond noir avec une écriture blanche (style par défaut des applications Windows Phone). Bien sûr, étant donné qu'il s'agit là de Silverlight, cette notion de style peut parfaitement être redéfinie/surchargée, si vous souhaitez la personnaliser (aux couleurs de votre application, etc.).

Fig.11

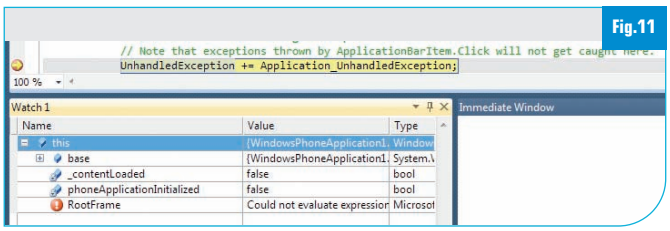


Fig.12

Fig.13

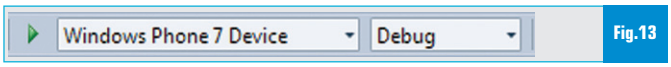


Fig.15

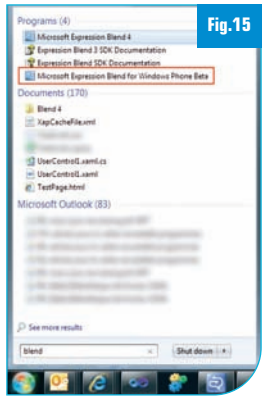


Fig.16

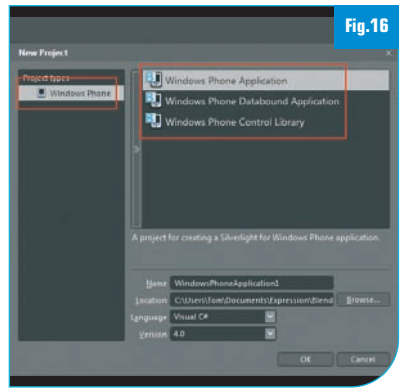


Fig.14

Fig.17

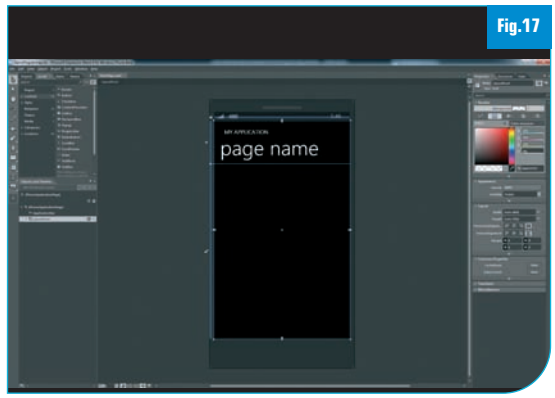
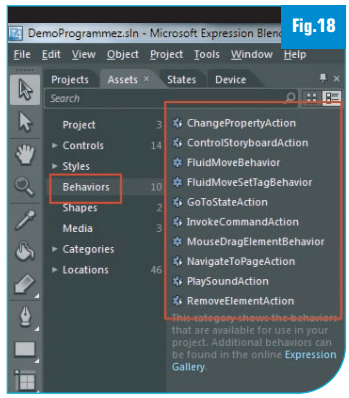


Fig.18



Une autre des fonctionnalités que l'on apprécie dans les développements Silverlight pour Windows Phone (mais bien entendu aussi disponible pour du Silverlight « classique ») est la possibilité d'utiliser des données en « design-time ». En effet, afin de faciliter la collaboration entre le designer et le développeur (qui, parfois, sont la même et unique personne), Expression Blend offre la possibilité d'afficher des données fictives, qui ne seront présentes que lorsque l'application sera visualisée dans un outil de design tel qu'Expression Blend, Cider (l'éditeur WPF/Silverlight disponible avec Visual Studio), etc. : [Fig.20]

Vous avez la possibilité de créer une nouvelle source de données fictive à partir de rien ou bien, pour avoir un rendu plus réaliste, créer votre source de données à partir d'une de vos classes ! Chacune des propriétés générées est entièrement personnalisable (type, longueur, etc.) : [Fig.21]

Une fois la source de données créée, il suffit de la faire glisser/déposer sur l'interface graphique afin que le DataContext de l'élément cible, en mode design, soit correctement renseigné et permette d'offrir des données en mode design : [Fig.22]

ALM et bonnes pratiques avec Windows Phone

Comme beaucoup de nouvelles technologies arrivant sur le marché, Windows Phone offre un très grand nombre de fonctionnalités aux développeurs : manipulations de l'ApplicationBar, prise de photos, gestion de l'Isolated Storage, etc. Cependant, il est toujours bon de se demander comment faire pour avoir à éviter de réinventer la roue lorsque l'on parle de développement d'applications. Fort heureusement, la technologie utilisée pour le développement d'applications Windows Phone est Silverlight. Ainsi, toutes les techniques acquises jusqu'à maintenant sont toujours valables. Dès lors, l'utilisation de pattern de développement tel que le pattern MVVM (Model View View Model) est toujours d'actualité (si ce n'est plus !). La mise en place de tests unitaires, points très importants d'une application mais souvent négligée par bon nombre de développeurs, est également possible avec Silverlight pour Windows Phone ! Enfin, qui dit exécution de tests unitaires pense bien souvent (mais pas toujours !), à usine de développement logiciel. Qui n'a jamais rêvé d'avoir à sa disposition une chaîne complète d'intégration continue avec :

- Récupération des dernières sources sur le serveur
- Exécution automatique des tests unitaires
- Eventuellement, exécutions automatiques des tests d'interfaces graphiques (cette fonctionnalité n'est pas encore disponible pour Silverlight mais devrait être prochainement mise à disposition)
- Compilation des sources
- Dépôts des binaires sur un répertoire partagé
- Envoi d'un email à un (ou plusieurs) développeurs/testeurs.

Ce scénario, qui peut paraître idéal, est cependant possible avec l'exécution d'applications Silverlight sur Windows Phone. En effet, la grande problématique à l'heure actuelle concerne la version des outils de développement à installer sur le serveur. Fort heureusement, Justin Angel (développeur chez Microsoft Corporation dans l'équipe Silverlight) a posté, sur son blog, un article expliquant comment être en mesure de réaliser de l'intégration continue, pour Windows Phone, sans avoir besoin d'installer le moindre outil sur le serveur ! Pour voir le post en question, c'est ici : http://justinangel.net/TFS2010WP7ContinuousIntegration. Ensuite, toutes les techniques de développement que vous aviez acquises pour le moment pourront être réutilisées dans vos projets : l'objectif de Microsoft est de fournir une expérience de développement similaire à ce que les développeurs connaissaient en Windows Forms, ASP.NET, etc.

Conclusion

Microsoft a complété son offre en matière d'outillage pour le développement sur la plateforme Windows Phone 7. Un point d'importance est mis sur la facilité de prise en main de ces outils, faisant ainsi en sorte qu'un développeur ayant l'habitude des technologies clientes telles que WPF ou des technologies RIA telles que Silverlight puisse passer sans difficulté au développement Windows Phone 7. Enfin, la disponibilité d'outils annexes tels que l'émulateur ou encore les outils de déploiement sur périphériques réels viennent encore faciliter les développements et permettent d'augmenter la productivité des développeurs et designers sans nuire à la qualité du code qu'ils produisent ! Vous connaissez à présent tous les outils qui vous permettront de développer des applications fiables, robustes et pérennes. Il ne vous reste plus qu'à les utiliser !

Fig.19

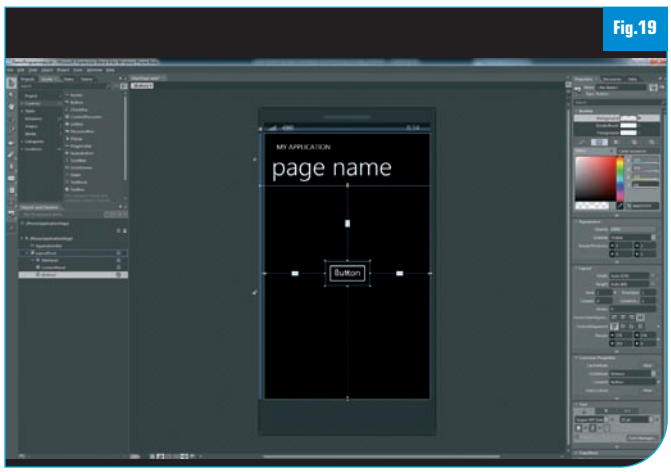


Fig.20

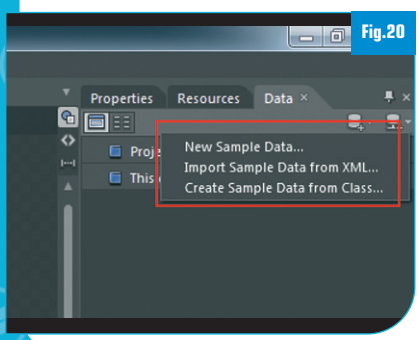


Fig.21

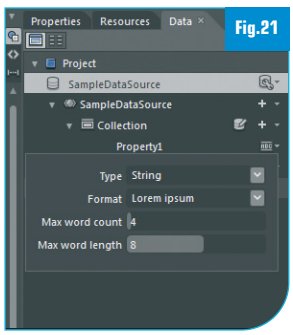


Fig.22

■ Julien **Corioland**
MVP Client App Dev,
Consultant / Formateur
Access It

■ Thomas **Lebrun**
Consultant /
Formateur,
MVP Client App Dev
Access It

Votre première application Windows Phone 7

Nous allons concevoir le projet *VeloToulouse*, une application pour connaître la liste des points de location de vélos à Toulouse et le nombre de vélos disponibles.

Cette application est relativement simple, composée uniquement de deux écrans :

- Un écran principal, comportant la liste de toutes les stations, des stations préférées et des stations proches de nous,
- Un écran de détail d'une station, comportant le nombre de vélos disponibles ainsi qu'une carte des alentours de la station.

Mise en place de l'application

Lorsque vous créez votre projet, la solution contient déjà un certain nombre d'éléments, et notamment :

- Un fichier MainPage.xaml contenant la page qui sera affichée par défaut,
- Un fichier ApplicationIcon.png qui est l'image qui sera affichée dans la liste des applications,
- Un fichier Background.png utilisé comme arrière-plan de la tile si l'application est punaisée sur l'écran d'accueil,
- Un fichier SplashScreenImage.jpg qui est utilisé lors du chargement de l'application [Fig.1].

Nous allons commencer par personnaliser ces éléments en important nos propres icônes et notre splash screen. Ceux-ci peuvent être réalisés avec n'importe quel outil de création graphique en respectant les résolutions et formats indiqués dans le tableau. Il suffit ensuite de les ajouter au projet. Vous devez cependant prêter attention à deux détails importants :

- Dans les propriétés de ces fichiers, Build Action doit être positionné à Content,
- Si les fichiers images de l'icône et du fond de la Tile peuvent être nommés comme bon vous semble, l'écran de chargement doit obligatoirement être à la racine et s'appeler SplashScreenImage.jpg.

Element graphique	Résolution	Format
Icône (ApplicationIcon.png)	62x62px	PNG
Fond de la Tile (Background.png)	173x173px	PNG
Ecran de chargement (SplashScreenImage.jpg)	480x800px	JPEG

Si vous changez les noms de fichiers, vous devez, dans les propriétés du projet, préciser quels sont les fichiers image à utiliser.
Revenons à notre fichier MainPage.xaml, vous pouvez noter trois éléments intéressants : le StackPanel *TitlePanel* contient le titre de l'application ainsi que le nom de la page. Ce StackPanel est suivi d'une Grid, *ContentGrid*, pour le contenu de la page. Pour notre application, nous allons conserver cette organisation, cependant vous pouvez tout à fait ne pas utiliser cette hiérarchie et utiliser la vôtre. Le dernier élément intéressant est un commentaire : c'est un code exemple d'utilisation de l'ApplicationBar, une *barre de menus* pour les applications Windows Phone. Nous y reviendrons plus tard. Nous pouvons déjà exécuter notre application et voir les premiers résultats [Fig.2 et 3].

Nos vélos en direct du web

Nous allons tout d'abord créer une classe Station pour décrire une station, puis une classe StationService pour pouvoir interroger le service web.
La classe Station comporte un certain nombre de propriétés contenant toutes les caractéristiques d'une station, et implémente INotifyPropertyChanged. Pourquoi ? Tout simplement car le service web de VeloToulouse ne nous retourne pas les informations sur la disponibilité en même temps que la liste des stations. Il faut donc appeler une méthode afin d'obtenir ces informations. L'implémentation de cette interface et l'utilisation sur nos propriétés permet d'utiliser les capacités de Binding de Silverlight pour mettre à jour automatiquement ces informations dans l'interface graphique. Cette classe contient également une méthode MapImageUrl renvoyant l'Url d'une carte de la zone (générée à l'aide du service Google Static Maps. Vous pouvez tout à fait utiliser Bing Maps, comme l'a fait Patrice Lamarche dans sa version de VeloToulouse). Passons maintenant à notre classe StationsService : c'est elle qui va effectuer les requêtes auprès du service web pour créer une liste de Stations. Silverlight possède tous les éléments

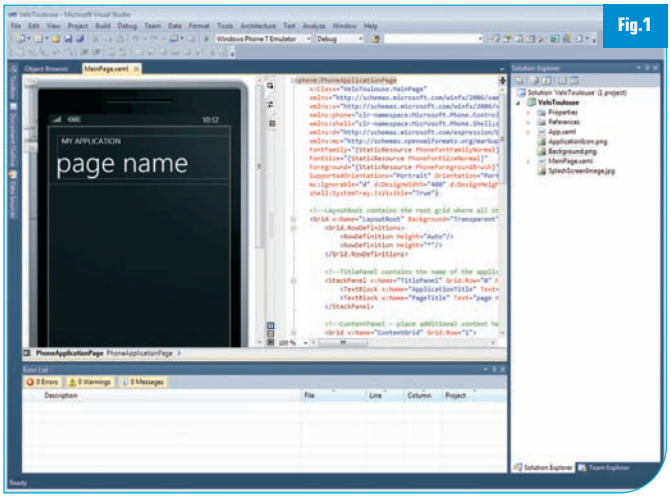


Fig.1

Fig.2

Fig.3

nécessaires pour interroger facilement des services web. Celui qui nous intéresse est un service REST ayant un format de réponse XML. Il nous suffit donc d'effectuer une requête http, puis de parler avec le document XML de résultat pour reconstruire nos objets. Pour effectuer une requête http et récupérer le résultat, nous pouvons utiliser la classe WebClient. Le code suivant permet d'effectuer une requête vers l'Url contenue dans la variable StationUrl, puis d'appeler la méthode LoadCompleted.

```
var webClient = new WebClient();
webClient.OpenReadCompleted += LoadCompleted;
webClient.OpenReadAsync(new Uri(StationUrl));
```

Nous allons maintenant parser le résultat XML afin de créer notre liste de stations. Connaissant la structure de notre réponse XML, le plus simple est d'écrire une requête LINQ afin de récupérer l'ensemble des informations et de retourner une liste de stations.

```
<carto>
<markers>
<marker name="00001 - POIDS DE L'HUILE"
   number="1"
   fullAddress="12 RUE DU POIDS DE L'HUILE"
   lat="43.60413468702105"
   lng="1.445420780739279" />
</markers>
</carto>
```

Nous avons choisi dans notre classe de stocker cette liste dans une ObservableCollection<Station> or, la requête LINQ nous retournant simplement une liste, nous devons parcourir celle-ci afin d'ajouter chacun des éléments à notre observableCollection.

```
private void LoadCompleted(object sender, OpenReadCompleted
EventArgs e)
{
 if (e.Error != null)
   return;
Stream s = e.Result;
XDocument xdoc = XDocument.Load(s);

List<Station> res = (xdoc.Descendants("marker").Select(
         i => new Station{
           Id = i.Attribute("number").Value,
           Nom = i.Attribute("name").Value,
           Adresse = i.Attribute("fullAddress").Value,
           Latitude = i.Attribute("lat").Value,
           Longitude = i.Attribute("lng").Value
           })).ToList();

res.ForEach(item => Stations.Add(item)); }
```

Dernière précision, cette classe implémente le design pattern Singleton. Cela nous permet d'accéder à une instance de la classe unique simplement via StationService.Current, et éviter,entre autres, de faire de multiples appels au web service inutilement.
Nous pouvons maintenant revenir à notre code XAML afin d'afficher notre liste de stations. Pour cela, nous allons ajouter à l'intérieur de notre grille de contenu une ListBox. Il faudra lui préciser deux informations : où aller chercher les données et quoi afficher. Si vous découvrez Silverlight, regardez le code qui suit : il suffit de préciser le nom de la propriété à afficher (avec DisplayMemberPath) et réaliser une opération de *Binding* – ou « connexion de données »- en précisant le nom de l'objet liste contenant les données, et le tour est joué. Nous verrons plus loin dans cet article comment personnaliser l'affichage de ces données.

```
<ListBox x:Name="listStations" ItemsSource="{Binding Stations}"
DisplayMemberPath="Nom" />
```

Vous vous posez peut-être la question suivante : d'où provient l'objet Stations utilisé dans l'ItemsSource ? Revenons dans le code Behind (par exemple, en appuyant sur F7) : il faut donner à votre page l'instance d'un objet contenant toutes les propriétés dont elle a besoin, appelé le Data Context. Nous avons déjà l'objet idéal pour remplir ce rôle : StationService. Il suffit donc de préciser que le DataContext est l'instance de notre service, puis effectuer un appel à la méthode GetListeStationsAsync() qui va récupérer les données depuis le service web.

```
public MainPage()
{
    InitializeComponent();

    this.DataContext = StationsService.Current;
    StationsService.Current.GetListeStationsAsync();
}
```

Il est tout à fait possible de préciser un DataContext sur un autre élément que la page, par exemple sur une Grid, tout comme il est possible d'initialiser directement côté code l'ItemSource de notre liste. Cependant, les bonnes pratiques de Silverlight tendent à ne pas lier fortement l'élément qui va afficher les données de celui qui les fournit. C'est d'ailleurs une séparation reprise en MVVM, un moyen efficace d'architecturer son application Silverlight.
Vous pouvez exécuter le programme à ce stade et voir la liste des stations s'afficher.
L'une des qualités les plus importantes des applications mobiles est la réactivité. De ce point de vue, et à ce stade, notre application se porte plutôt bien. L'affichage de l'écran principal est immédiat, et les stations s'affichent une fois la liste récupérée. Il manque une étape : informer l'utilisateur qu'on est bien en train de charger les données. Cela peut simplement s'effectuer via une ProgressBar. Nous n'avons pas d'indicateur précis sur la récupération des stations, nous pouvons donc utiliser la propriété *IsIndeterminate* de la ProgressBar. Quand celle-ci est à true, la ProgressBar affiche une animation que l'on retrouve dans d'autres applications Windows Phone 7. Il suffit d'afficher cette ProgressBar au début du chargement et de ne plus l'afficher une fois le chargement terminé.

```
<Grid x:Name="LayoutRoot" Background="Transparent">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="*"/>
```

```
    </Grid.RowDefinitions>
    <StackPanel x:Name="LoadingPanel" Grid.Row="0" Visibility
="Collapsed">
      <ProgressBar x:Name="MainProgress" />
      <TextBlock Text="Chargement des stations..." />
    </StackPanel>
    <StackPanel x:Name="TitlePanel" Grid.Row="1" Margin="24,
24,0,12">
      <TextBlock x:Name="ApplicationTitle" Text="VELOTOULOUSE"
Style="{StaticResource PhoneTextNormalStyle}"/>
      <TextBlock x:Name="PageTitle" Text="stations" Margin="
-3,-8,0,0" Style="{StaticResource PhoneTextTitle1Style}"/>
    </StackPanel>
    <Grid x:Name="ContentGrid" Grid.Row="2">
            <ListBox x:Name="listStations"
            ItemsSource="{Binding Stations}"
            DisplayMemberPath="Nom">
      </ListBox>
    </Grid>
  </Grid>
```

```
    public MainPage()
    {
      InitializeComponent();

      this.DataContext = StationsService.Current;
      StationsService.Current.GetListeStationsCompleted += new
 EventHandler(service_GetListeStationsCompleted);
      MainProgress.IsIndeterminate = true;
      LoadingPanel.Visibility = System.Windows.Visibility.Visible;
      StationsService.Current.GetListeStationsAsync();
    }

    void service_GetListeStationsCompleted(object sender, EventArgs e)
    {
      if (e == null)
      {
        MessageBox.Show("Erreur lors de la récupération des données");
      }
      MainProgress.IsIndeterminate = false;
      LoadingPanel.Visibility = System.Windows.Visibility.Collapsed;

    }
```

D'une station à l'autre, il n'y a qu'une page

Nous allons maintenant créer un affichage détaillé pour chaque station. Pour cela, nous allons créer une nouvelle page (Clic droit sur votre projet -> Add -> New Item -> Windows Phone Portrait Page) StationDetail.xaml. Pour un premier pas, nous allons simplement afficher le nom de la station, la carte et le nombre de vélos disponibles en ajoutant le code suivant dans la ContentGrid.

```
<StackPanel>
  <TextBlock Text="{Binding Nom}" />
  <Image Source="{Binding MapImageUrl}" />
  <StackPanel Orientation="Horizontal">
    <TextBlock Text="Nombre de vélos disponibles : " FontSize="
{StaticResource PhoneFontSizeMedium}" />
    <TextBlock Text="{Binding VelosDisponibles}" FontSize="
{StaticResource PhoneFontSizeMedium}"  />
  </StackPanel>
</StackPanel>
```

Comme nous l'avons fait pour notre liste de stations, il faut, depuis le Code Behind, récupérer la station, rafraîchir le nombre de vélos disponibles, et configurer le DataContext. Ces opérations sont effectuées au chargement de la page, il faudra donc brancher cet EventHandler soit dans le code, soit dans le XAML.

```
private void Page_Loaded(object sender, RoutedEventArgs e)
{
  Station station;
  string stationId = "";
  NavigationContext.QueryString.TryGetValue("id", out stationId);
  if (!string.IsNullOrEmpty(stationId))
  {
    station = StationsService.Current.GetStationById(stationId);
    if (station != null)
```

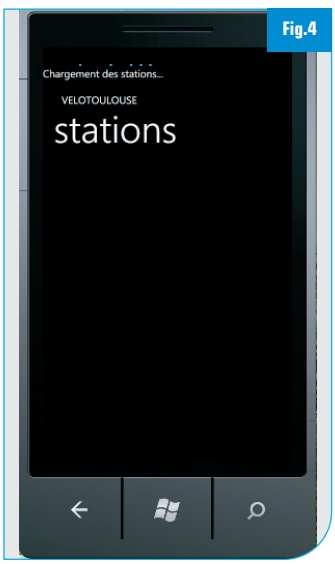


Fig.4

Fig.5

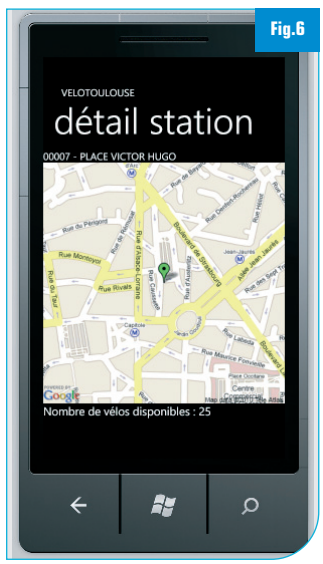


Fig.6

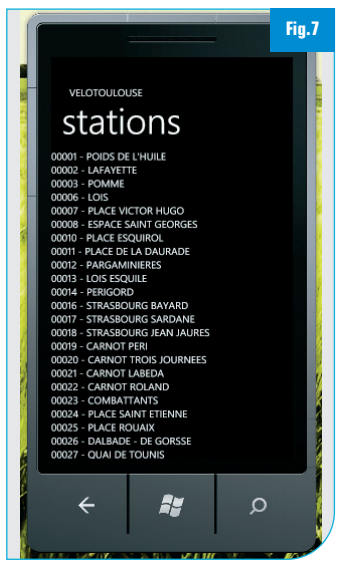


Fig.7

```
    {
      station.RefreshAsync();
      ContentGrid.DataContext = station;
    }
  }
}
```

Si vous regardez bien ce code, vous pouvez voir que l'on récupère l'ID de la station par une QueryString. Mais comment faire passer un paramètre dans l'URL alors qu'on est dans une application Silverlight Out of Browser ? On en vient au dernier point à implémenter : comment passer d'un élément sélectionné dans notre liste à la fiche d'une station ?
Nous allons pour cela utiliser le NavigationService de Silverlight. Il permet de passer d'une page à l'autre simplement via son URL, ici relative par rapport à la racine de notre projet. Comme on peut le voir dans le code suivant – levé lors d'un changement de sélection dans la liste des stations de notre MainPage, on peut simplement passer des paramètres comme on le ferait à une page web.

```
private void listStations_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    NavigationService.Navigate(
              new Uri("/StationDetail.xaml?id=" + ((Station)
listStations.SelectedItem).Id, UriKind.Relative));
}
```

Il nous est désormais possible d'afficher notre liste de stations, puis d'afficher le détail d'une station. Par défaut, en appuyant sur la touche Back du téléphone depuis le détail d'une station, nous revenons à la page précédente [Fig.6].
Il reste un détail qui ne nous saute pas aux yeux lorsque l'on teste l'application dans l'émulateur, mais qui est pourtant essentiel : la gestion de l'orientation. Tous les téléphones Windows Phone 7 sont équipés d'un accéléromètre, et Silverlight nous fournit tous les éléments dont nous avons besoin pour gérer de manière simple et efficace une application pouvant s'exécuter avec différentes orientations. Il y a deux façons d'aborder les choses :

- On veut simplement étirer notre interface,
- On veut modifier la position des éléments pour mieux correspondre à l'orientation (par exemple déplacer une zone qui se situe en dessous en mode portrait vers le côté en mode paysage).

La première approche est très simple : il suffit, dans votre code Xaml, de préciser SupportedOrientations="PortraitOrLandscape"

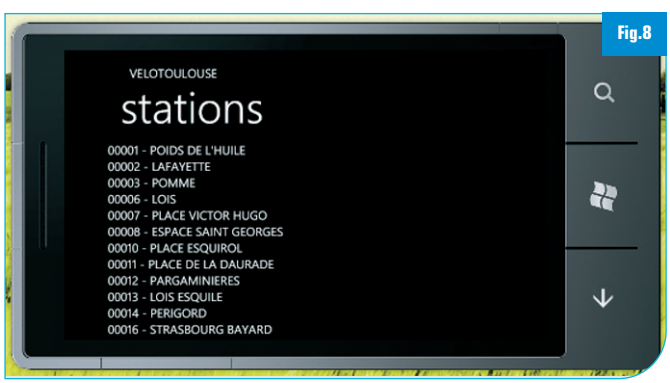


Fig.8

dans votre PhoneApplicationPage, puis de profiter des fonctionnalités de mise en page offertes par Silverlight. Après tout, dans cette approche, le changement d'orientation équivaut à un redimensionnement d'une fenêtre ou d'une page sur une application Desktop, ce que Silverlight gère très bien. Pour ce cas, deux éléments de Silverlight sont intéressants : le StackPanel – permettant d'empiler des contrôles – et le ScrollViewer –permettant de scroller au doigt une zone de contenu [Fig.7 et 8].
La seconde approche est plus poussée mais permet de maîtriser complètement l'apparence de son interface. Les pages proposent un événement OrientationChanged, auquel vous pouvez vous abonner pour ainsi changer le positionnement de vos éléments. Prenons la page de détail d'une station. Nous voulons afficher la carte sur la droite et non pas en dessous du texte en mode paysage. Pour cela, nous allons modifier notre présentation pour que les éléments soient positionnés dans une grille 2X2. Le texte ne bougera pas : il sera toujours dans la case 1A (1re ligne, 1re colonne). En mode portrait, la carte sera affichée en 2A, alors qu'en paysage, elle sera en 1B. Il suffira d'écrire le code suivant.

```
private void PhoneApplicationPage_OrientationChanged(object
sender, OrientationChangedEventArgs e)
{
  if((e.Orientation & PageOrientation.Portrait) == Page
Orientation.Portrait)
  {
    Grid.SetRow(MapImage,1);
    Grid.SetColumn(MapImage,0);
  }
  else
  {
    Grid.SetRow(MapImage, 0);
    Grid.SetColumn(MapImage, 1);
  }
}
```

C'est une manière simple et efficace de gérer l'orientation d'une manière un peu plus aboutie. Vous pouvez tout à fait avoir un comportement plus évolué, par exemple, en jouant sur la visibilité pour passer d'une présentation à une autre totalement différente.

Comme je la veux, quand je la veux.

Le téléphone est un objet personnel, de nombreux utilisateurs le personnalisent pour qu'il leur corresponde : ils personnalisent le thème, changent le fond d'écran et choisissent les applications qu'ils installent. On va prolonger cet esprit en permettant aux utilisateurs de choisir leurs stations préférées. Mais avant cela, faisons un point sur les possibilités de stockage qui vous sont offertes. La première place pour stocker vos données, c'est votre projet. Tout comme n'importe quelle application .net, vous avez la possibilité d'embarquer dans votre application des ressources : textes, images, fichiers, etc…Vous devez simplement avoir deux points en tête avec cette solution : vous ne pouvez pas modifier ces fichiers et vous devez en limiter la taille (n'oubliez pas que votre application est susceptible d'être téléchargée et installée via des réseaux cellulaires). Vous n'avez pas accès au système de fichiers du téléphone, mais à un emplacement mémoire qui vous est réservé : l'IsolatedStorage. Vous pouvez donc créer, lire et

modifier des fichiers dans cet espace simplement. Vous avez également accès à un Dictionnaire dans lequel vous pouvez stocker vos objets simplement. C'est la solution idéale pour stocker nos stations favorites. Nous allons tout d'abord modifier la classe StationService pour exposer une collection de stations dans laquelle nous allons stocker les stations favorites de l'utilisateur. Nous allons également coder deux méthodes pour charger cette liste depuis les settings, et sauvegarder cette liste.

```
public void LoadFavoris()
{
 if (!IsolatedStorageSettings.ApplicationSettings.Contains("favoris"))
   IsolatedStorageSettings.ApplicationSettings["favoris"] =
new ObservableCollection<Station>();

 StationsFavorites = (ObservableCollection<Station>)Isolated
StorageSettings.ApplicationSettings["favoris"];

}
public void SaveFavoris()
{
 IsolatedStorageSettings.ApplicationSettings["favoris"]  =
 StationsFavorites;
 IsolatedStorageSettings.ApplicationSettings.Save();
}
```

Nous allons également ajouter une propriété StationFavorite à notre classe Station. Retournons maintenant sur le détail d'une station. Nous allons utiliser l'Application Bar (ou AppBar) afin de proposer deux boutons pour ajouter et supprimer la station courante dans la liste des favoris. Nous sommes déjà passés à côté de cette fonctionnalité au début de l'article : rappelez-vous le code XAML qui était commenté lors de la création du projet. Le code suivant permet d'afficher l'AppBar avec nos deux boutons.

```
 <phone:PhoneApplicationPage.ApplicationBar>
   <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
     <shell:ApplicationBarIconButton x:Name="appbar_btnAdd"
IconUri="/Images/appbar.favs.addto.rest.png" Text="Ajouter au
favoris" Click="appbar_button1_Click"></shell:ApplicationBarIconButton>
     <shell:ApplicationBarIconButton x:Name="appbar_btnRemove"
IconUri="/Images/appbar.favs.rest.png" Text="Supprimer" Click=
"appbar_button2_Click"></shell:ApplicationBarIconButton>
   </shell:ApplicationBar>
 </phone:PhoneApplicationPage.ApplicationBar>
```

```
private void appbar_button1_Click(object sender, System.
EventArgs e)
{
 Station currentStation = (Station) this.DataContext;
 currentStation.StationFavorite = true;
 if  (!StationsService.Current.StationsFavorites.Contains
(currentStation))
 {
     StationsService.Current.StationsFavorites.Add(current
Station);
 }
 StationsService.Current.SaveFavoris();
}
private void appbar_button2_Click(object sender, System.
EventArgs e)
{
 Station currentStation = (Station)this.DataContext;
 currentStation.StationFavorite = false;
 if  (StationsService.Current.StationsFavorites.Contains
(currentStation))
 {
   StationsService.Current.StationsFavorites.Remove(current
Station);
 }
 StationsService.Current.SaveFavoris();
}
```

Nous avons désormais l'enregistrement de nos favoris, il ne nous reste plus qu'à l'afficher ! Au lieu d'afficher un texte du type Station favorite : oui/non, nous allons afficher un texte uniquement si cette station fait partie de la liste des élues. Le problème est que la visibilité n'est pas un booléen. Nous ne pouvons donc pas directement la binder, il faut passer par un converter qui va se charger de caster la valeur au moment du binding. Vous pouvez retrouver le code de ce converter dans les sources de l'article.

```
<TextBlock Text="J'aime cette station" Visibility="{Binding
StationFavorite,Converter={StaticResource  boolToVisibility
Converter}}"  />
```

Nous avons maintenant intégré les favoris dans notre application [Fig.9]. Restons sur cette page : que se passe-t-il si on appuie sur le bouton démarrer, ou si on reçoit un appel ? Notre application sera fermée, le modèle d'exécution de Windows Phone 7 ne permet pas d'avoir des applications qui tournent en arrière-plan. Tout n'est pas perdu pour autant.

Deux solutions sont là pour nous aider à gérer le retour dans l'application et à réveiller l'application en cas de besoin.

La première solution est appelée tombstoning. En quelques mots, lorsque l'utilisateur quitte inopinément votre application, le système lui donne 10 secondes pour enregistrer son état courant. L'utilisateur peut ensuite ressusciter l'application et revenir au même point qu'avant. Il a également la possibilité de lancer une nouvelle instance de l'application. On a deux groupes d'événements disponibles Launching/Closing qui sont activés lorsque l'application est lancée pour la première fois (depuis l'écran d'accueil ou la liste de programmes) et lorsque l'utilisateur quitte l'application, en appuyant sur la touche retour depuis la première page de l'application.

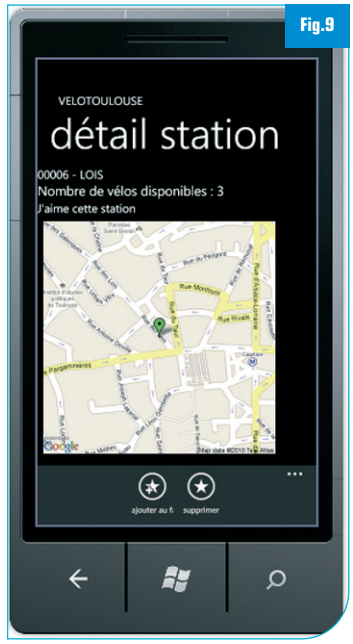


Fig.9

Desactivated/Activated sont eux appelés lors de l'opération de tombstoning et nous permettent de sauvegarder les informations nécessaires à la restauration de notre application. L'application est hibernée lors de l'invocation d'un launcher ou d'un shooser, sur l'appui du bouton Démarrer ou répond à un appel. Une application hibernée peut être relancée sur l'appui de la touche retour sur l'écran d'accueil.

Revenons à VeloToulouse, si vous essayez d'hiberner l'application actuelle et de la ressusciter, vous n'aurez aucun problème sur la liste des stations, cependant, si vous vous situez sur la page de détail, vous verrez un écran presque vide, contenant les titres mais aucune donnée. Le service de Navigation conserve l'historique et la page courante, mais il ne stocke pas les paramètres passés à la page, ainsi que tous les objets que vous avez instanciés en mémoire.

Si vous souhaitez gérer le tombstoning, il vous faudra gérer l'enregistrement de vos données, l'état courant de votre application et de votre page. Nous allons tout d'abord sauvegarder nos données. Pour cela, il existe un objet PhoneApplicationService.Current.State, fonctionnant de la même manière que l'Isolated StorageSettings pour l'application. Nous allons donc sauvegarder nos listes de stations et de stations favorites lors de la désactivation de l'application.

Dans notre cas, nous sauvegardons nos données dans l'état, mais l'Isolated Storage est également une solution intéressante. A vous d'évaluer quelles données peuvent être persistées à long terme, et quelles sont vos données éphémères.

```
private StationsService()
{
  Stations = new ObservableCollection<Station>();
  StationsFavorites = new ObservableCollection<Station>();
  PhoneApplicationService.Current.Deactivated += new EventHandler<DeactivatedEventArgs>(Current_Deactivated);
}
public void WakeUpService()
{
    Stations = (ObservableCollection<Station>) PhoneApplicationService.Current.State["Stations"] ;
    StationsFavorites =(ObservableCollection<Station>) PhoneApplicationService.Current.State["StationsFavorites"];
}
void Current_Deactivated(object sender, DeactivatedEventArgs e)
{
  PhoneApplicationService.Current.State["Stations"] = Stations;
  PhoneApplicationService.Current.State["StationsFavorites"] = StationsFavorites;
}
```

Il nous reste à gérer le passage de paramètres à notre page de détails. Pour cela, nous avions utilisé un paramètre passé à la page. Nous allons donc stocker dans l'état de la page l'Id passé en paramètre. Au chargement de celle-ci, si l'état contient un ID, nous utilisons celui-là et dans le cas contraire celui passé en arguments de la page.

```
private void Page_Loaded(object sender, RoutedEventArgs e)
{
  Station station;
  string stationId = "";

  if (this.State.ContainsKey("Id"))
  {
    stationId = (string) this.State["Id"];
    this.State.Clear();
  }
  else
    NavigationContext.QueryString.TryGetValue("id", out stationId);

  if (!string.IsNullOrEmpty(stationId))
  {
    this.State.Add("Id", stationId);
    station = StationsService.Current.GetStationById(stationId);
    if (station != null)
    {
      station.RefreshAsync();
      this.DataContext = station;
    }
  }
}
```

Peut-on quitter une application Silverlight ? Oui, mais pas directement par Silverlight. Il faut faire appel à XNA et appeler la méthode Exit() sur une instance de la classe Game. Bien que cela semble un peu surprenant, on a ici une information très intéressante : on peut tout à fait utiliser la puissance de XNA dans des applications Sliverlight. Utile pour manipuler des images, des sons, etc…

Les notifications Push nous permettent d'envoyer un message au téléphone depuis un serveur web. Trois types de notifications peuvent être envoyées au téléphone : un message brut, reçu uniquement par votre application, un toast, qui sera affiché à l'utilisateur qui pourra ensuite lancer l'application. Enfin les *tiles notifications* peuvent mettre à jour la tuile de l'application si celle-ci a été punaisée sur l'écran d'accueil.

Un dernier coup de peinture

Pour cette dernière partie, nous allons donner un coup de peinture à notre application. En tout premier, nous allons mettre en place le contrôle pivot sur l'écran d'accueil. Au moment de l'écriture de cet article, ce contrôle n'est pas fourni publiquement par Microsoft. Cependant, une version créée par la communauté est disponible sur Codeplex (http://phone.codeplex.com/).

L'utilisation du contrôle Pivot est très simple. Nous allons simplement afficher deux listes, directement bindées sur les listes de notre service. Bien évidemment, vous pouvez ajouter n'importe quel contrôle à un élément d'un Pivot.

```
<Samples:PivotControl>
  <Samples:PivotItem Header="toutes">
    <ListBox x:Name="listStations"
      SelectionChanged="listStations_SelectionChanged"
      ItemsSource="{Binding Stations}"
      DisplayMemberPath="Nom"/>
  </Samples:PivotItem>
  <Samples:PivotItem Header="favorites">
    <ListBox x:Name="listStationsFavorites"
```

```
      SelectionChanged="listStations_SelectionChanged"
      ItemsSource="{Binding StationsFavorites}"
      DisplayMemberPath="Nom"/>
  </Samples:PivotItem>
</Samples:PivotControl>
```

[Fig.10]

Avant de styliser notre application, nous allons ajouter une dernière fonctionnalité : la liste des stations près de ma position actuelle. En effet, tous les téléphones Windows Phone possèdent des capteurs tels qu'un GPS ou un accéléromètre. Nous allons voir ici comment utiliser l'accéléromètre.
La première étape est d'ajouter une référence vers System.Device. Il faut ensuite utiliser la classe GeoCoordinateWatcher afin de récupérer la position à chaque fois que celle-ci est récupérée.

```
GeoCoordinateWatcher watcher;
watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.Default);
watcher.MovementThreshold = 20; // permet d'ignorer le bruit
watcher.PositionChanged += watcher_PositionChanged;
watcher.Start();
```

Une fois la position courante récupérée, si celle-ci est à moins de deux miles nautiques de notre position courante, on l'ajoute à la liste. La méthode CalculateDistanceBetweenGPSCoordinates est fournie dans le code source de l'article.

```
private void watcher_PositionChanged(object sender, GeoPosition
ChangedEventArgs<GeoCoordinate> e)
{
 GeoPosition<GeoCoordinate> position = watcher.Position;

   foreach (Station station in Stations)
   {
     if (
CalculateDistanceBetweenGPSCoordinates(Convert.ToDouble(station.
Longitude),
Convert.ToDouble(station.Latitude),position.Location.Longitude,
position.Location.Latitude) <= 2)
       StationsNearMe.Add(station);
   }
}
```

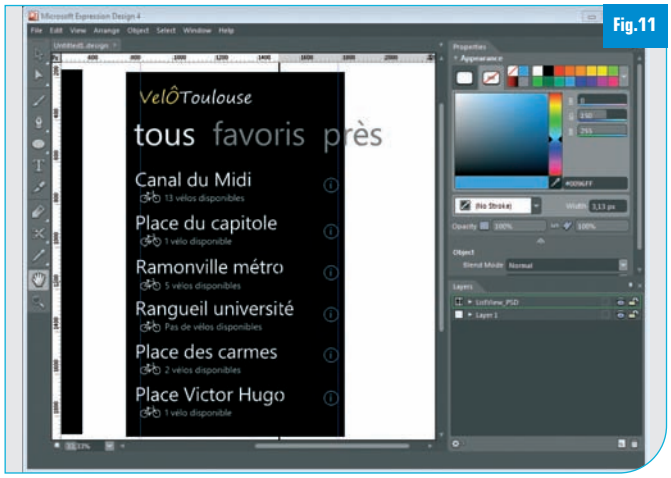


Fig.11

Attention : l'utilisation du GPS est une opération très consommatrice en énergie. Pour une application telle que VeloToulouse, une récupération d'une position à un instant T serait suffisante.
Silverlight est une technologie qui est idéale pour fournir des applications riches graphiquement. Les applications Windows Phone 7 sont des cibles idéales pour tirer parti de ces capacités. Même si le style Metro est très épuré, ce n'est pas pour autant qu'il doit être un frein au développement de l'application. Expression Design 4, non fourni avec les outils de développement Windows Phone, est un logiciel de dessin vectoriel. Il permet à des graphistes (ou aux développeurs ayant une âme d'artiste) de réaliser des maquettes et les graphismes d'applications Windows Phone 7 [Fig.11]. Afin d'aller plus vite, vous pouvez utiliser les Design Templates fournis par Microsoft (http://go.microsoft.com/fwlink/?LinkId=196225), qui sont au format Photoshop, mais importables dans Expression Design.
Une fois le travail effectué dans Expression Design, vous pouvez passer à un autre outil, qui lui, est fourni par défaut : Expression Blend. Cet outil est destiné aux développeurs/intégrateurs qui sont en charge de l'interface graphique. Il permet de réaliser ces tâches beaucoup plus simplement et rapidement qu'avec Visual Studio. Vous pouvez importer ou copier/coller vos éléments graphiques directement d'Expression Design à Expression Blend, ce qui vous permet ainsi de gagner beaucoup de temps. Il est également aisé de passer d'Expression Blend à Visual Studio. Votre solution est visible de la même manière avec les deux outils, et vous pouvez passer d'un outil à l'autre par un simple clic droit.

Fig.10

Les Styles et les DataTemplates vous permettent de réutiliser des éléments de présentation très facilement sur le même écran ou au travers d'une application entière. Dans VeloToulouse, nous pouvons ainsi définir un DataTemplate qui représente l'affichage d'une station dans les différentes listes. Nous pouvons écrire ce DataTemplate dans les ressources de notre page afin qu'il soit accessible de toutes nos listes.
Il suffit ensuite sur chacune de nos trois listes de référencer ce template.

Fin de la course

Nous sommes arrivés au terminus de cet article. Désormais, Il ne vous reste plus qu'à développer vous-même vos propres applications, et à les publier – gratuitement ou de façon payante- sur le Windows Marketplace. A vos claviers !

■ Christopher **Maneu**
Windows Phone 7 Metro Trainer
Consultant @ Bewise

Tirer profit des ressources et des fonctionnalités de Windows Phone 7 dans votre application

La possibilité d'utiliser Silverlight et XNA pour développer une application pour mobile permet de standardiser à travers les 3 écrans et le cloud bon nombre d'API. Pour autant, un téléphone reste un terminal mobile avec du matériel et des fonctionnalités spécifiques dont il serait dommage de ne pas faire usage dans une application. Si vous ne tirez parti d'aucune fonctionnalité du téléphone, quelle est l'utilité de développer votre application spécifiquement sur plateforme mobile ? Windows Phone 7, et plus particulièrement son langage de design, Metro, définit 3 « fils rouges » dans l'expérience utilisateur : « Personal, Relevant, Connected ». Nous allons voir comment utiliser les spécificités de Windows Phone 7 pour que votre application applique ces principes !

L'avantage des mobiles Windows Phone 7 est qu'ils ont une spécification matérielle stricte : on sait donc qu'on disposera toujours des mêmes « tripes » dans le téléphone, même si « l'emballage » change. Dans tous les terminaux Windows Phone 7 on retrouvera donc 3 boutons en façade, un appareil photo numérique 5MP avec Flash, A-GPS, accéléromètre, boussole ainsi que capteurs de proximité et de luminosité, une puce d'accélération graphique, un CPU à 1Ghz minimum, etc. De plus, le système propose un certain nombre d'API qui permettent d'accéder non pas à des fonctionnalités matérielles mais bien aux données stockées sur le téléphone, comme les contacts, la musique, la possibilité de lancer certaines applications natives…

L'API de localisation

La localisation peut-être un casse-tête pour les développeurs, car il existe différentes manières plus ou moins simples de récupérer sa position sur la plupart des terminaux mobiles : soit celui –ci est équipé d'un GPS (c'est le cas de tous les terminaux Windows Phone 7) mais dans ce cas il faut être en extérieur et avoir acquis le signal au préalable (ce qui prend dans le meilleur des cas quelques secondes), soit il n'est pas équipé d'un GPS, et il faut alors faire appel à une base de donnée sur internet, souvent payante pour le développeur qui souhaite l'utiliser, qui convertit les points d'accès wifi des alentours, ou pire car moins précis, l'identifiant de la tour cellulaire à laquelle le terminal est connecté à son opérateur. Gérer intelligemment toutes ces sources d'informations peut être fastidieux pour le développeur, fort heureusement, l'API Windows Phone 7 de localisation abstrait cette complexité. Une seule classe suffit pour bénéficier de toutes ces sources : la classe GeoCoordinateWatcher. Lors de l'instanciation, on spécifie la précision que l'on souhaite : la valeur « Default » évitera l'utilisation du GPS tout en garantissant la meilleure position possible, alors que la valeur High tentera d'utiliser le GPS si un fix peut être obtenu.

```
GeoCoordinateWatcher watcher = new
GeoCoordinateWatcher(GeoPositionAccuracy.High);
```

On peut ensuite ajuster la propriété MovementThreshold qui s'exprime en mètres et permet de déclencher des évènements uniquement après un déplacement d'une certaine distance.

```
watcher.MovementThreshold = 10;
```

Enfin, il faut s'abonner aux évènements StatusChanged et PositionChanged pour récupérer les infos que nous renvoie le service de localisation

```
watcher.StatusChanged += new
EventHandler<GeoPositionStatusChangedEventArgs>(watcher_Status
Changed);
watcher.PositionChanged += new
EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(watcher
_PositionChanged);
watcher.Start();
```

Les coordonnées sont renvoyées avec la précision estimée (en mètres, elle aussi). Malgré une API de haut niveau, on a cependant un contrôle relativement fin sur le type d'information qu'on reçoit et l'utilisation qu'on a des ressources, et donc de la batterie ! Voici comment mettre à jours quelques TextBlocks avec les informations renvoyées par le GPS : comme vous pouvez le constater nous devons faire appel à Dispatcher.BeginInvoke car ces events handlers ne tournent pas dans le thread d'UI.

```
void watcher_PositionChanged(object sender,
GeoPositionChangedEventArgs<GeoCoordinate> e)
{
  Dispatcher.BeginInvoke(() =>
```

```
    {
      tbLatitude.Text = e.Position.Location.Latitude.ToString();
      tbLongitude.Text = e.Position.Location.Longitude.ToString();
      tbSpeed.Text = e.Position.Location.Speed.ToString();
      tbCourse.Text = e.Position.Location.Course.ToString();
      tbVerticalAccuracy.Text = e.Position.Location.VerticalAccuracy
.ToString();
      tbHorizontalAccuracy.Text = e.Position.Location.Horizontal
Accuracy.ToString();
      tbAltitude.Text = e.Position.Location.Altitude.ToString();
    });
}

void watcher_StatusChanged(object sender, GeoPositionStatus
ChangedEventArgs e)
{
 Dispatcher.BeginInvoke(() =>
    {
      tbStatus.Text = e.Status.ToString();
    });
}
```

Vous voilà équipés pour géolocaliser les informations de votre application

L'accéléromètre

L'accéléromètre permet de mesurer la position et les mouvements du téléphone. Il fournit sur 3 axes les mouvements que subit le terminal : X est dans le sens de l'abscisse de l'écran en mode portrait, Y dans le sens de l'ordonnée et Z est la normale de l'écran. On peut donc par défaut trouver la position du téléphone (car l'accéléromètre mesure la gravité), mais aussi la façon dont l'utilisateur agite son terminal, ce qui peut enrichir l'interactivité de votre application, particulièrement si c'est un jeu.
On accède à l'accéléromètre uniquement de façon asynchrone, c'est-à-dire quand l'évènement Accelerometer.ReadingChanged est levé. Il suffit donc de s'y abonner pour recevoir régulièrement les informations qu'il produit. Le code le plus simple qu'on puisse imaginer ressemble donc à cela :

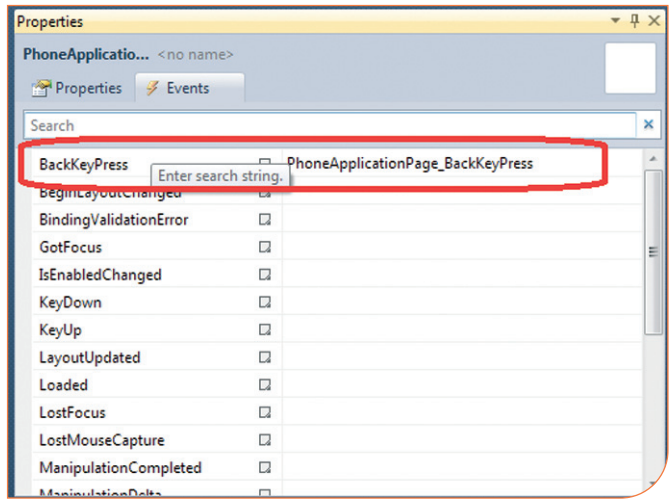
```
Accelerometer accel = new Accelerometer();
accel.ReadingChanged += new EventHandler<AccelerometerReading
EventArgs>(accel_ReadingChanged);
accel.Start();
```

et l'event handler correspondant, encore une fois pour mettre à jour des contenus de TextBlock par exemple :

```
void accel_ReadingChanged(object sender, AccelerometerReading
EventArgs e)
{
    Dispatcher.BeginInvoke(() =>
    {
        tbX.Text = e.X.ToString();
        tbY.Text = e.Y.ToString();
        tbZ.Text = e.Z.ToString();
    });
}
```

Les boutons

Certains développeurs pourraient être tentés de modifier l'usage des boutons hardware du téléphone dans leur application. Cela n'est possible que pour un seul bouton, le bouton Back. Il faut utiliser l'évènement associé à l'appui sur ce bouton avec parcimonie : ce bouton est en effet utilisé à travers toute l'interface et les applications du téléphone comme un moyen de naviguer vers la page précédente. Attention donc à ne pas « perdre » l'utilisateur en en changeant radicalement son fonctionnement !!
L'évènement auquel s'abonner pour récupérer l'appui sur ce bouton « back » est BackKeyPress dans la classe PhoneApplicationPage. Cela permet de définir le comportement du bouton back différemment sur chaque page d'une application.
La plupart du temps, intervenir sur cet évènement permettra d'effectuer un traitement comme une sauvegarde du contexte ou le déclenchement d'une animation avant le retour à la page précédente. Dans certains cas, on peut vouloir annuler complètement ce retour, dans ce cas il faut passer à « true » le booléen « Cancel » de l'argument de l'event handler :




```
private void PhoneApplicationPage_BackKeyPress(object sender,
 System.ComponentModel.CancelEventArgs e)
{
    // Do your stuff here - Start an animation, save some
context or whatever
    // Use this line to cancel the event if you don't want
to navigate back
    e.Cancel = true;
}
```

Les espaces de stockage sur le téléphone

Chaque terminal Windows Phone 7 disposera d'au moins 8Go de stockage. Pour autant, les applications étant sandboxées, l'accès au système de fichier est interdit. Cependant, il faut quand même fournir aux développeurs de quoi faire persister des données d'une exécution à l'autre de l'application : c'est le rôle de l'Isolated Storage, une API héritée de la version web de Silverlight qui permet de faire persister des données sur le PC de l'utilisateur.
Chaque application dispose donc d'un espace de stockage qui lui est propre. La première chose à faire pour s'en servir est donc de récupérer un « handle » vers cet espace de stockage. Ensuite, on

peut utiliser les API classiques de .NET (StreamReader et StreamWriter par exemple) pour y écrire, comme si l'on écrivait dans un système de fichiers classiques. Du coup on peut aussi structurer les données sous forme de dossiers et de fichiers. La limite de volume que chaque application peut utiliser est de 2Go : il faudra donc s'efforcer en tant que développeur d'être « bon citoyen » afin de ne pas pénaliser les autres applications sur le terminal !
Voici un exemple très simple d'écriture dans un fichier :

```
IsolatedStorageFile isf =
IsolatedStorageFile.GetUserStoreForApplication();

using (StreamWriter sw = new StreamWriter(new
IsolatedStorageFileStream(«myfile.txt»,  FileMode.OpenOrCreate,
isf)))
{
    sw.WriteLine(tbxWriteToIS.Text);
}
```

Et l'équivalent en lecture, tout aussi simple :

```
IsolatedStorageFile isf =
IsolatedStorageFile.GetUserStoreForApplication();

using (StreamReader sr = new StreamReader(new
IsolatedStorageFileStream(«myfile.txt», FileMode.Open, isf)))
{
    tbxReadFromIS.Text = sr.ReadLine();
}
```

Les autres espaces de stockage sont ceux qui sont accessibles globalement depuis tout le téléphone à savoir la liste de contacts, et la librairie multimedia qui contient photos, vidéos, et musique. Pour l'instant les API limitent l'utilisation de ces espaces par des applications tierces : il est possible de sauver des emails et des numéros de téléphones dans la liste des contacts, par l'intermédiaire de « choosers » dont nous détaillons le fonctionnement plus tard. Il est également possible de stocker les images dans la bibliothèque d'images plutôt que dans l'isolated storage avec la méthode de SavePicture de la classe MediaLIbrary.

Les Launchers et Choosers

Launchers et choosers font appel à une nouvelle fenêtre pour accéder à des données ou une fonctionnalité du téléphone. La différence entre ces deux types d'API réside dans le fait qu'un Launcher ne retourne pas naturellement à l'application une fois fermé, alors qu'un Chooser revient forcément à l'application avec une valeur de retour et signale ce retour par le déclenchement d'un évènement au niveau de l'application.
Voici la liste des launchers disponibles sur le système : EmailComposeTask, MarketplaceDetailTask, MarketplaceHubTask, MarketplaceReviewTask, MarketplaceSearchTask, MediaPlayerLauncher, PhoneCallTask, SearchTask, SMSComposeTask, et WebBrowserTask. Comme vous pouvez le voir, ces launchers servent donc à lancer des applications natives : que ce soit l'écriture d'un email ou d'un SMS, l'ouverture de la marketplace, y compris sur une page définie, la manipulation du lecteur de musiques et vidéos du système, le lancement du browser, de l'application de recherche, ou le démarrage d'un appel téléphonique. L'appel à un launcher se fait toujours de la même manière : il faut instancier un objet de la classe correspondante, éventuellement fournir les différents arguments dans les champs de l'objet, puis appeler la méthode « Show() » de celui-ci.

```
EmailComposeTask ect = new EmailComposeTask();
ect.To = «pierreca@microsoft.com»;
ect.Subject = «EmailComposeTask Sample»;
ect.Body = «Sent from a sample code explaining the use of
EmailComposeTask»;

ect.Show();
```

Passons maintenant aux choosers : ceux-ci permettent de sélectionner une donnée disponible sur le téléphone ou bien de la sauver : par exemple l'email d'un contact, ou une photo. Les choosers disponibles sont : CameraCaptureTask, EmailAddressChooserTask, PhoneNumberChooserTask, PhotoChooserTask, SaveEmailAddressTask et SavePhoneNumberTask.
Ces choosers fonctionnent de la même manière : on commence par instancier un objet de la classe correspondante, et dans de rares cas, remplir des arguments en en remplissant les champs de cet objet. Enfin, avant d'appeler la méthode Show() il ne faut pas oublier d'ajouter un event handler à l'évènement « Completed », évènement qui sera déclenché quand l'utilisateur reviendra à l'application après en avoir fini avec le chooser, ce qui permet d'en traiter le résultat. Lors de l'appel à cet event handler, il sera possible de consulter la valeur de retour (le champ TaskResult) pour vérifier si la tâche a été effectuée ou annulée par l'utilisateur, qui aurait appuyé sur le bouton back par exemple.

```
private void btnCameraCaptureTask_Click(object sender, Routed
EventArgs e)
{
    CameraCaptureTask cct = new CameraCaptureTask();
    cct.Completed += new EventHandler<PhotoResult>(cct_Completed);
    cct.Show();
}

void cct_Completed(object sender, PhotoResult e)
{
    if (e.Error != null)
        // Do something with the e.Error.Message for example :)
    else
        // Use e.TaskResult and e.OriginalFileName for example
 to use the picture that was just taken
}
```

Les feedback utilisateurs : vibrer et jouer un son ou de la musique

Pour l'utilisateur d'un téléphone mobile le feedback le plus habituel suite à une action est évidemment visuel : une animation, un changement de page, etc. Mais il peut être intéressant d'utiliser d'autres formes de feedback comme le vibreur ou un son (attention cependant à ne pas transformer le téléphone en boîte à musique! votre utilisateur est peut-être dans une salle de classe silencieuse avec son téléphone sous la table !!)

Le vibreur

Rien de plus simple que de faire vibrer le téléphone : il suffit de faire appel à la méthode Vibrate() en lui indiquant une durée (TimeSpan). Voici un exemple pour une vibration de 0 heures, 0 minutes et 2 secondes :

```
VibrateController.Default.Start(new TimeSpan(0, 0, 2));
```

Jouer un son

Il existe deux manières de jouer un son ou de la musique : soit en utilisant une Api qui joue un son court par exemple pour un bruitage, soit en faisant appel au player audio du téléphone. Ces deux API viennent de XNA et plus précisément de l'espace de nommage Microsoft.Xna.Framework.Audio. Avec Windows Phone 7 il est en effet possible d'appeler une API XNA depuis Silverlight et vice versa. La première option consiste à utiliser l'API SoundEffect qui permet de jouer simplement un son ou un bruitage. L'avantage de cette Api est qu'elle est simple à utiliser et permet de jouer des sons qui ne font pas partie de la librairie Media. Par exemple pour jouer un son stocké dans les « embedded resources » de l'application :

```
string streamName = «MyProject.tada.wav»;
using (System.IO.Stream stream =
this.GetType().Assembly.GetManifestResourceStream(streamName))
{
    SoundEffect se = SoundEffect.FromStream(stream);
    FrameworkDispatcher.Update();
    se.Play();
}
```

L'inconvénient de cette API réside dans son fonctionnement en mode « Fire & Forget » : il est donc impossible une fois que la lecture du son est lancée, de l'arrêter ou de la mettre en pause par exemple. Pour cela il faudrait utiliser l'API SoundEffectInstance, à peine plus complexe :

```
bool isPlaying = false;
SoundEffectInstance sei;
```

Et plus tard dans le code, par exemple dans le handler d'un bouton play :

```
if (isPlaying == false)
{
    string streamName = «MyProject.tada.wav»;
    using (System.IO.Stream stream =
this.GetType().Assembly.GetManifestResourceStream(streamName))
    {
        SoundEffect se = SoundEffect.FromStream(stream);
        FrameworkDispatcher.Update();
        sei = se.CreateInstance();
        isPlaying = true;
        btnSoundEffect.Content = «Stop»;
        sei.Play();
    }
}
else
{
    sei.Stop();
    isPlaying = false;
    btnSoundEffect.Content = «Play Sound Effect»;
}
```

L'autre méthode consiste à utiliser le lecteur multimedia du téléphone, qui peut tourner en background. Cela peut servir par exemple à jouer de la musique présente dans la bibliothèque de musique du téléphone pendant le déroulement de votre application. On peut par exemple faire appel à la bibliothèque de musique pour choisir une chanson, puis demander à la classe MediaPlayer de la jouer en background : voici par exemple comment jouer la première chanson de la bibliothèque de musique :

```
if (isPlaying == true)
{
    MediaPlayer.Stop();
   btnUseMediaPlayer.Content = «Play Music w/ Media Player»;
    isPlaying = false;
}
else
{
    MediaLibrary ml = new MediaLibrary();
    Song s = ml.Songs.First();
    FrameworkDispatcher.Update();
    MediaPlayer.Play(s);
    isPlaying = true;
    btnUseMediaPlayer.Content = «Stop MediaPlayer»;
}
```

Dans ce cas, le mediaplayer se mettra en marche background et son interface (avec les boutons play, pause etc.) apparaîtra temporairement en haut de l'écran avant de disparaître. Pour la faire réapparaître, il suffira de toucher un bouton de contrôle du volume par exemple.

Conclusion

Toutes ces API spécifiques à Windows Phone 7 permettent d'enrichir l'application avec de nombreuses fonctionnalités. Pour autant il faut rester « bon citoyen » dans l'utilisation de ces ressources qui peuvent consommer de la batterie ou de l'espace mémoire. Ceci étant dit, on pourrait se demander quel serait l'utilité de développer une application pour mobile si elle ne tire parti d'aucune des fonctionnalités mobiles de l'appareil !

Bien que cet article ne soit pas exhaustif il couvre la plupart des fonctions « classiques » qu'on pourrait trouver sur n'importe quel Windows Phone 7. Il est possible que la fonction de vos rêves que vous n'avez pas vu dans cet article existe . Pour autant certaines API manquent encore à la première version du SDK : l'accès à la boussole par exemple. Mais nous avons la garantie de la part de l'équipe Windows Phone que ce SDK a vocation à s'enrichir au fur et à mesure des versions. Gageons que de nombreuses autres API intéressantes et aussi bien structurées que les existantes arriveront dans le futur !

■ Pierre **Cauchois**
Relations avec les développeurs - technologies embarquées et mobiles - Microsoft France
http://blogs.msdn.com/pierreca - @pierreca

Les notifications en push avec Windows Phone 7

Un système de notifications en push permet à une application Windows Phone 7 de s'enregistrer auprès d'un service web afin de recevoir de la part de ce dernier des informations, sans que l'application ait à tourner ou à demander régulièrement des mises à jour d'information. En plus d'être plus élégant qu'un système de polling, c'est un palliatif au fait qu'avec Windows Phone 7 une application ne peut s'exécuter en background : si elle n'est pas à l'écran, il n'y a pas de code qui tourne. Nous allons dans cet article détailler le système de notifications de Windows Phone 7 en prenant pour exemple une application réelle, AHEAD, qui est une application de gestion de projet avec un client PC, un client mobile, et un service web hébergé dans Azure.

Il est courant de créer une application pour téléphone qui va interroger un service web. Une notification push est capable de réveiller une application grâce au service de notification de Microsoft : MPN (Microsoft Push Notification). Il est obligatoire de passer par ce service pour envoyer une notification au terminal, en fait, seul Microsoft communique avec le Windows Phone 7 : et le service web AHEAD (notre projet mobile) communique avec le service MPN : il y a donc 4 acteurs dans notre scénario : le terminal WP7, l'application PC, le service web AHEAD et le service de notifications Microsoft (MPN). Dans le cas de notre application nous allons voir comment :

- Le client Windows Phone 7 s'abonne pour recevoir des notifications en push
- L'application PC communique avec le service hébergé dans le cloud pour créer un nouveau projet
- Lors de la création d'un projet, le service envoie une notification au serveur MPN
- Le service MPN contacte le téléphone Windows Phone 7
- Le téléphone WP7 réveille l'application AHEAD
- L'application AHEAD exploite le service dans le cloud pour récupérer le projet créé

Le service web

Le service web AHEAD hébergé dans Azure contient de nombreuses méthodes, et propose entre autres des méthodes pour qu'un client Windows Phone 7 s'abonne ou se désabonne auprès de celui-ci, et des méthodes pour récupérer tous les projets ainsi que créer un projet :

```
[OperationContract]
void AddProject(string title);

[OperationContract]
List<string> GetAllProjects();

[OperationContract]
void RegisterPhone(string url);

[OperationContract]
void UnRegisterPhone(string url);
```

```
private void RegisterPhone(string url)
{
    if (!listPhone.Exists(url)
    {
        listPhone.Add(channelUri);
    }
}

private void UnRegisterPhone(string url)
{
    if (listPhone.Exists(url)
    {
        listPhone.Remove(url);
    }
}
```

Le service WCF doit être de type basicHttpBinding ou RESTFull pour être exploité par une application Windows Phone 7. Il doit également utiliser la propriété Single pour partager le même contexte entre tous les clients. C'est un point extrêmement important car sinon il ne serait pas possible de maintenir une liste des téléphones enregistrés ! A chaque appel de RegisterPhone par un client Windows Phone 7, le service ajoute dans une liste l'adresse communiquée en paramètre : cette adresse est une URL fournie par le service MPN au client Windows Phone 7. Nous verrons plus tard comment obtenir cette URL, qui identifie de manière unique le terminal et l'application. Cela permet de savoir très exactement quelle notification on envoie à qui.

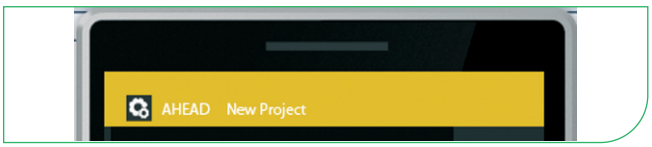
Quand le client PC appelle la méthode AddProject, le service web AHEAD va donc générer un message, qui prendra la forme d'une notification en push en direction du terminal Windows Phone 7, à travers un canal représenté par l'url qu'on a envoyée en paramètre au moment de l'enregistrement du terminal. L'envoi de notification push se réalise grâce aux classes HttpWebRequest et Stream. Ce sont des API .NET classiques, pas besoin de s'étendre outre-mesure sur celles-ci. Vous noterez qu'on formate quelques headers spéciaux, qui pourront ensuite être interprétés par le Windows Phone pour savoir comment traiter la notification. Cette

étape étant remplie nous avons une bonne partie de l'architecture en place : le service web AHEAD permet d'enregistrer un terminal, d'ajouter un nouveau projet et d'envoyer une notification aux terminaux enregistrés quand un nouveau projet a été créé. Il reste à s'occuper de la partie sur le terminal Windows Phone 7.

Le client WP7

Côté client, nous avons plusieurs choses à faire : d'abord, récupérer un identifiant unique auprès du service MPN et l'envoyer au service AHEAD afin de s'enregistrer pour recevoir les notifications. Puis nous devons choisir entre plusieurs types de notifications : *tile*, *raw* et *toast*. Ces 3 types permettent d'observer un comportement différent sur le téléphone, et d'envoyer différents types de messages : Les notifications de type « Tile » vont s'afficher sur l'icône que l'utilisateur peut choisir de placer sur le « Start Screen », l'écran d'accueil caractéristique de Windows Phone 7, qu'on voit souvent composé de carrés bleus. En fait, chaque application peut avoir son icône dynamique sur cet écran, et donc afficher du texte, une image et même un compteur directement sur cet écran d'accueil. C'est extrêmement intéressant car cela permet de pousser du contenu et des informations à l'utilisateur sans qu'il lance l'application ! Cela encourage également l'utilisateur à rentrer dans l'application pour en découvrir plus. A vous de voir ce que vous voulez faire apparaître. L'inconvénient de ces notifications est que l'utilisateur doit avoir pro-activement choisi de placer votre application sur l'écran d'accueil pour voir ces notifications « Tile » !

Les notifications de type « Toast » vont afficher temporairement une petite barre en haut de l'écran. Cette petite barre s'affiche même quand l'utilisateur navigue dans une autre application. Pas besoin de revenir à l'écran d'accueil donc. En revanche, cette petite barre disparaît au bout d'un certain laps de temps. C'est une notification éphémère, ce qui la rend peu intrusive, et qui permet aux applications qui n'ont pas été choisies par l'utilisateur pour figurer sur l'écran d'accueil de quand même signaler que quelque chose est arrivé. Le dernier type de notification est « raw » : il s'agit d'une notification qui ne pourra être reçue que lorsque l'application est lancée. Cela permet par exemple de mettre à jour l'affichage d'une page sans action de l'utilisateur, et c'est aussi le moyen le moins intrusif d'envoyer des données au téléphone : on ne pousse des informations que lorsque l'utilisateur est effectivement en train d'utiliser l'application. A vous de voir quel type de notification convient le mieux à l'usage de votre application, sachant que vous pouvez bien entendu faire usage des 3 ! Pour notre application AHEAD nous choisissons d'utiliser les toasts :



Commençons par créer un canal de notification entre le client Windows Phone 7 et le service MPN.

```
channel = HttpNotificationChannel.Find(channelName);
if (channel != null)
    Debug.WriteLine(channel.ChannelUri.ToString());
else
{
    channel = new HttpNotificationChannel(channelName);
    channel.ChannelUriUpdated += channel_ChannelUriUpdated;
    channel.Open();
    channel.BindToShellToast();
}
```

Si le canal n'existe pas déjà, le client en recrée un. Sinon le service MPN renverra l'identifiant du canal existant en vous signifiant qu'il est déjà ouvert. Le client s'abonne donc à l'événement ChannelUriUpdated pour retrouver l'URL fournie par le service MPN.

```
channel.ChannelUriUpdated += channel_ChannelUriUpdated;
void channel_ChannelUriUpdated(object sender, NotificationChannelUriEventArgs e)
{
    Debug.WriteLine(e.ChannelUri);
}
```

C'est avec cette URL renvoyée par le service MPN que le client va s'enregistrer auprès du service web AHEAD.

```
string baseUri = «http://localhost/Service/RegisterPhone?url={0}»;
string theUri = String.Format(baseUri, e.ChannelUri.ToString());
WebClient client = new WebClient();
client.DownloadStringAsync(new Uri(theUri));
```

Nous avons maintenant réuni toutes les pièces pour un système de notification : le client WP7 qui s'enregistre auprès du service MPN et envoie son URL au service web AHEAD, l'application PC qui va créer un nouveau projet et bien entendu l'envoi d'une notification par le service web AHEAD à travers le service MPN vers le téléphone. Reste à savoir quoi faire quand on reçoit une notification ! Pour cela, nous nous abonnons côté client WP7 à l'événement HttpNotificationReceived :

```
void httpChannel_HttpNotificationReceived(object sender, HttpNotificationEventArgs e)
{
    //Code à réaliser
}
```

Reste à savoir ce que vous voulez faire, mais ça c'est votre boulot ! Pour aller plus loin avec les notifications push, rendez-vous sur http://download.microsoft.com/documents/France/MSDN/2010/WP7_NotificationsTypePush_FR.pdf

■ Julien **Dollon**

LA QUESTION QUI TUE : je suis développeur d'application mobile, comment choisir où héberger mon service web qui permettra aux terminaux de s'enregistrer et qui enverra au service MPN les notifications ?

Comme c'est une question qui tue, il n'y a pas de réponse simple. Tout dépend du volume de notification que vous souhaitez traiter ! Sur une application à faible portée, un petit serveur mutualisé suffira largement pour traiter les quelques requêtes par jour… mais si vous brassez un gros volume de notifications, alors il faudra peut-être opter pour un serveur dédié. D'autant plus si vous commencez à avoir besoin d'une base de données derrière ! Cette base de données devra-t-elle être reliée à une infrastructure existante ? Si c'est le cas votre système de notification mérite peut-être une place dans votre datacenter, sinon chez un hébergeur ?

Avez-vous regardé Windows Azure ? L'offre cloud de Microsoft permet de payer à l'usage et de scaler facilement votre service, dans l'éventualité d'un succès fulgurant de votre application (c'est tout le mal qu'on peut vous souhaiter !)

Ecrire un contrôle pour Silverlight

Avec WPF et Silverlight (pour desktop ou pour le téléphone), un développeur peut créer deux types de contrôles distincts qui répondent à des problèmes différents : un contrôle utilisation ou un contrôle personnalisé. Dans cet article, nous allons créer notre propre contrôle.

Le contrôle personnalisé (ou « templated control » dans les outils Silverlight) répond à un problème technique, d'interactivité : par exemple un contrôle permettant d'afficher une liste d'éléments sous la forme d'un carrousel, un contrôle permettant de saisir une date… Un contrôle personnalisé permet aussi de supporter des scénarios d'utilisation plus poussés : DataBinding, animation, séparation de l'apparence et du comportement, et surtout support des styles et templates.

Note : *la saisie d'une adresse et la saisie d'une date ne sont pas équivalentes, car le type « Date » correspond à un type primitif du Framework .Net et donc son éditeur ne dépend pas d'une entité métier d'une application spécifique, mais peut être utilisé dans toute application nécessitant de manipuler des dates.*

Nous aborderons plusieurs notions fondamentales de Silverlight, puis on se concentrera exclusivement sur un exemple de templated control : le Spinner.

SILVERLIGHT – QUELQUES RAPPELS POUR LES NON-INITIÉS

Les lecteurs déjà familiers avec les notions de DataBinding, d'animation, de styles et de templates de contrôles peuvent directement aller à la partie « Le contrôle Spinner ». Pour les autres, un rapide rappel s'impose !

DataBinding

Un des concepts les plus importants de Silverlight est certainement le DataBinding. Ce n'est pas vraiment une notion nouvelle (les développeurs Windows Forms et ASP.Net utilisent eux aussi des systèmes de DataBinding pour lier des contrôles graphiques à des sources de données), mais la manière dont elle a été implémentée en Silverlight et WPF est totalement différente. Cet article n'a pas pour but d'être exhaustif sur les possibilités offertes par le moteur de Binding de Silverlight, mais de le présenter succinctement.

DataBinding – à quoi ça sert ?

Le but du DataBinding, est de déclarer que la valeur d'une propriété d'un contrôle dépend de la propriété d'un autre objet (autre contrôle, objet CLR classique, …). La définition d'un binding en xaml est très simple :

```
<StackPanel x:Name=»LayoutRoot» >
  <TextBox x:Name=»theTextBox» />
  <TextBlock Text=»{Binding Text, ElementName=theTextBox}» />
</StackPanel>
```

Automatiquement, le TextBlock va afficher le texte contenu dans la textbox située au-dessus au fur et à mesure que l'utilisateur tape du texte. Cet exemple est très basique, mais il faut savoir que les sources de données possibles sont quasi illimitées. Par exemple, imaginons que le DataContext de notre contrôle contient un objet C# contenant une propriété de type List<Person>. Pour afficher cette liste des personnes, on pourrait faire ainsi :

```
<ItemsControl ItemsSource=»{Binding People}»>
  <ItemsControl.ItemTemplate>
    <DataTemplate>
      <StackPanel>
        <TextBlock Text=»{Binding LastName}» FontWeight=»Bold» />
        <TextBlock Text=»{Binding FirstName}» />
      </StackPanel>
    </DataTemplate>
  </ItemsControl.ItemTemplate>
</ItemsControl>
```

Vous pouvez noter l'utilisation d'un DataTemplate permettant de définir comment chaque élément est affiché. Il s'agit là aussi d'un des gros atouts de Silverlight.

DataBinding – côté contrôle, comment ça se passe ?

Du côté du contrôle, pour qu'une propriété supporte le binding, il faut la déclarer sous forme d'une DependencyProperty. Voici un exemple de propriété définie sous la forme d'une DependencyProperty (le type du contrôle est MyControl, et la propriété s'appelle MyInt) :

```
public int MyInt
 {
 get { return (int)GetValue(MyIntProperty); }
 set { SetValue(MyIntProperty, value); }
 }

 public static readonly DependencyProperty MyIntProperty =
   DependencyProperty.Register(«MyInt», typeof(int), typeof
(MyControl), new PropertyMetadata(0));
```

Note : *la syntaxe peut sembler fastidieuse, mais Visual Studio 2010 contient un code Snippet permettant d'accélérer l'écriture de ces propriétés : tapez « propdp » puis pressez 2 fois la touche tabulation.*

Styles et templates de contrôles

Avec la plupart des frameworks d'interfaces graphiques, développer un contrôle personnalisé revient à écrire une classe comprenant du code de dessin (affichage du contrôle), et de comportements (évènements claviers, souris, touch…). Avec Silverlight et WPF, Microsoft a introduit un nouveau modèle : un contrôle est composé d'une classe définissant l'état et le comportement du contrôle, tandis que l'affichage est pris en charge par un Template écrit en Xaml. Ceci procure deux énormes avantages :

- Le xaml est bien plus adapté à l'expression d'éléments graphiques que C#, et est beaucoup plus facilement manipulable

par des outils de Design (Expression Blend par exemple). Quiconque a déjà vécu l'écriture de CustomControl Windows Forms ne peut qu'être d'accord avec moi

- Un contrôle est généralement fourni avec un style / template d'affichage par défaut… mais rien n'empêche le développeur qui l'utilise de créer une copie de ce template et de le modifier, afin d'obtenir l'affichage qu'il souhaite tout en gardant le comportement d'origine.

L'association d'un template à un contrôle est abordée plus loin dans cet article

Le contrôle Spinner

Le Spinner permet de sélectionner un élément d'une liste sous la forme d'une liste défilante qui apparaît quand le contrôle prend le focus.

- Quand un élément est sélectionné, il vient automatiquement se placer au centre du contrôle.
- Quand le défilement perd son inertie, l'élément qui est alors le plus près du centre devient l'élément sélectionné.
- Quand le contrôle perd le focus, seul l'élément sélectionné est visible. [Fig.2]

Ce contrôle existe dans Windows Phone (il est notamment utilisé dans le paramétrage de la date et l'heure du téléphone et des alarmes), mais il n'est pas disponible dans le SDK Silverlight. La première chose à faire lors de la création d'un templated control est de choisir soigneusement quel contrôle de base utiliser. Dans le cas du spinner, le but est d'afficher une liste d'éléments et de permettre la sélection d'un élément. C'est exactement ce que permet de faire le contrôle ListBox (mais avec une représentation graphique et une interaction utilisateur légèrement différente). On va donc dériver de ce contrôle (qui gère déjà le DataBinding, le templating via la propriété ItemTemplate, la virtualisation graphique, et la synchronisation des propriétés SelectedItem et SelectedIndex).

Note : *avec WPF, on aurait plutôt choisi de dériver du contrôle Selector. Il s'agit du contrôle de base de la ListBox qui implémente entièrement le comportement de listing / selection, mais qui n'a aucun style par défaut*

Note : *La virtualisation graphique est une technique permettant d'afficher des listes de données quasi-infinies sans dégradation des performances. Dans le cas d'une ListBox par exemple, seuls les éléments réellement visibles à l'écran sont affichés. Quand l'utilisateur scrolle, les éléments visuels qui disparaissent sont réutilisés pour afficher les éléments de la liste qui apparaissent.*

Il nous restera alors à modifier le comportement initial de la ListBox pour obtenir un Spinner fonctionnel.

Création de la librairie de contrôles

Une librairie de contrôles Silverlight contient 2 types d'éléments : des classes comprenant le comportement des contrôles, et des styles par défaut à appliquer à ces contrôles. Les classes sont donc des fichiers C# classiques, alors que les styles doivent être présents dans un dictionnaire de ressources spécial ayant pour chemin « [racine du projet]/Themes/generic.xaml ». Ce chemin est très important et est défini par convention de nommage : [Fig.3]
La première chose que l'on doit faire est de lier un contrôle à son style par défaut. Pour ceci, nous allons modifier dans le constructeur de nos contrôles la propriété DefaultStyleKey, afin que le style correct soit systématiquement utilisé :

```
[StyleTypedProperty(Property = «ItemContainerStyle»,
    StyleTargetType = typeof(SpinnerItem))]
 public class Spinner : ListBox
 {
    public Spinner()
    {
      DefaultStyleKey = typeof(Spinner);
      [...]
    }
    [...]
 }
```

Vous pouvez aussi remarquer la présence d'un contrôle SpinnerItem. En effet, un ItemControl manipule des « ItemContainers », dérivant de ContentControl, et utilisés pour afficher chaque élément de la liste. La ListBox fonctionne avec des ItemContainers de type ListBoxItem afin de gérer les VisualStates de sélection. Dans le cas du Spinner, les éléments non-sélectionnés doivent aussi être masqués lorsque le contrôle perd le focus : nous devons donc créer un contrôle SpinnerItem dérivant de ListBoxItem.

Note : *Vous pouvez remarquer l'attribut StyleTypedProperty sur le contrôle Spinner. Il s'agit d'un attribut spécifique au design-time indiquant à Expression Blend le type des ItemContainers permettant à un designer de les styliser.*

Dans le fichier generic.xaml, nous allons alors pouvoir définir les styles / templates par défaut de nos contrôles (le markup présenté ici a été simplifié, le contenu complet se trouve sur le support accompagnant le magazine avec l'intégralité du code source) :

```
<ResourceDictionary  xmlns=»http://schemas.microsoft.com/winfx/2006/xaml/presentation»
          xmlns:x=»http://schemas.microsoft.com/winfx/2006/xaml»
          xmlns:SStuff_PhoneControls=»clr-namespace:SStuff.PhoneControls»
          xmlns:d=»http://schemas.microsoft.com/expression/blend/2008»
          xmlns:mc=»http://schemas.openxmlformats.org/markup-compatibility/2006»
          mc:Ignorable=»d»>
  <Style TargetType=»SStuff_PhoneControls:SpinnerItem»>
    <Setter Property=»Template»>
      <Setter.Value>
        <ControlTemplate   TargetType=»SStuff_PhoneControls:SpinnerItem»>
```

Fig.2

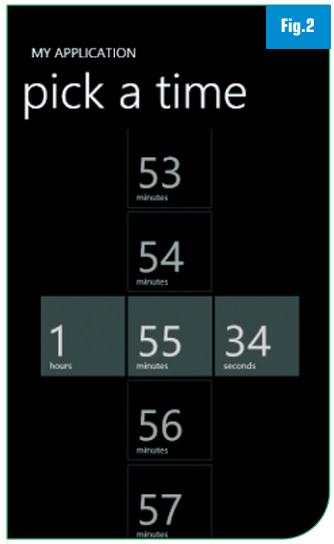
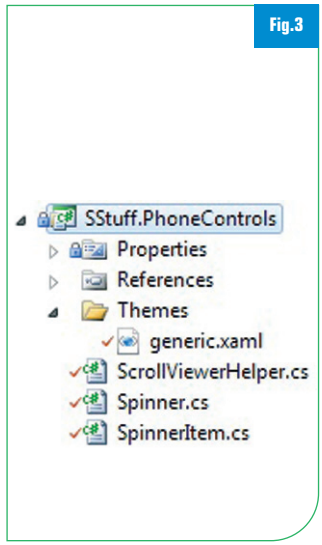


Fig.3



```
        […]
      </ControlTemplate>
    </Setter.Value>
  </Setter>
  […]
 </Style>
 <Style TargetType=»SStuff_PhoneControls:Spinner»>
  […]
  <Setter Property=»Template»>
    <Setter.Value>
      <ControlTemplate  TargetType=»SStuff_PhoneControls:
Spinner»>
        <ScrollViewer x:Name=»ScrollViewer»
          BorderBrush=»{TemplateBinding BorderBrush}»
          BorderThickness=»{TemplateBinding BorderThickness}»
          Background=»{TemplateBinding Background}»
          Foreground=»{TemplateBinding Foreground}»
          Padding=»{TemplateBinding Padding}»>
          <Grid>
            […]
          </Grid
        </ScrollViewer>
      </ControlTemplate>
    </Setter.Value>
  </Setter>
  <Setter Property=»ItemsPanel»>
    <Setter.Value>
      <ItemsPanelTemplate>
        <VirtualizingStackPanel />
      </ItemsPanelTemplate>
    </Setter.Value>
  </Setter>
  […]
 </Style>
</ResourceDictionary>
```

Quelques remarques sur ces styles :

- Le contrôle ListBox définit un TemplatePart nommé ScrollViewer de type System.Windows.Controls.ScrollViewer lui permettant d'interagir avec le ScrollViewer de son template. Il nous faut donc un élément dans le template par défaut du Spinner respectant ce TemplatePart.
- L'utilisation de TemplateBindings permet de lier une propriété d'un élément du template à une DependencyProperty du contrôle auquel il s'applique.
- Nous utilisons un ItemsPanelTemplate de type VirtualizingStackPanel pour permettre le binding à des sources de données contenant beaucoup d'éléments (le cas classique étant la sélection de dates dont l'année peut aller de 0001 à 9999).

Génération des SpinnerItems

Fort heureusement, la ListBox gère déjà entièrement la virtualisation des éléments visuels. Le modèle est très facilement extensible, et permet de générer n'importe quel type de contrôle à la place de ListBoxItem. Pour ce faire, le contrôle ListBox possède 4 méthodes virtuelles que l'on peut surcharger dans des contrôles dérivés :

- bool IsItemItsOwnContainerOverride(object item) : Permet d'indiquer si un élément de la liste nécessite un container (si l'élément lui-même est un SpinnerItem, il n'a pas besoin d'être incorporé dans un ItemContainer)
- DependencyObject GetContainerForItemOverride() : Crée un ItemContainer pour le donner au système de virtualisation
- void PrepareContainerForItemOverride(DependencyObject element, object item) : Permet d'exécuter du code lorsqu'un container est associé à un élément de la liste.
- void ClearContainerForItemOverride(DependencyObject element, object item) : Permet d'exécuter du code lorsqu'un élément est dissocié de son container.

Dans le cas du Spinner, nous allons simplement surcharger les 2 premières méthodes afin de créer des SpinnerItems à la place des ListBoxItem.

Communication entre le contrôle et les éléments de son template

Afin d'implémenter le comportement du contrôle, ce dernier doit pouvoir communiquer avec son template. Pour cela, 3 moyens sont à notre disposition :

- Les TemplateBindings : Il s'agit de bindings particuliers permettant de lier une propriété d'un élément du template à une propriété du contrôle associé.
- Les VisualStates : permet de définir des états visuels et de leur associer des animations. Le passage d'un état à un autre est effectué dans le code du contrôle, alors que le rendu visuel est défini dans le template.
- Les TemplateParts : il s'agit de contrats définis par le contrôle, demandant au template de contenir un élément nommé (voir le cas du template part « ScrollViewer » un peu plus haut). Pour récupérer une référence à un contrôle nommé du template, on surcharge la méthode OnApplyTemplate et on appelle « GetTemplateChild() ».

Ces 3 possibilités sont illustrées dans le code et le markup du Spinner, disponible sur le site www.programmez.com, mais voici tout de même un exemple de récupération du TemplatePart ScrollViewer (qui nous permet ensuite de synchroniser Scrolling et sélection des éléments) :

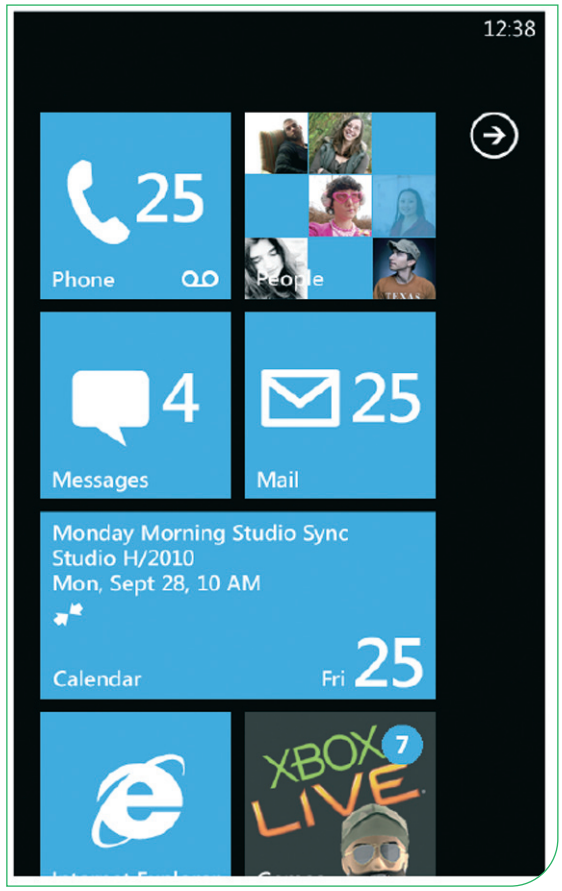
```
public override void OnApplyTemplate()
 {
   base.OnApplyTemplate();
   // we get a reference to the associated scroll viewer,
and create a ScrollViewerHelper on top of it
   // for handling the end of inertia
   _scrollViewer = GetTemplateChild(«ScrollViewer») as Scroll
Viewer;
   _scrollViewerHelper  =  new  ScrollViewerHelper(_scroll
Viewer);
   _scrollViewerHelper.VerticalScrollInertiaStopped  +=  On
VerticalScrollInertiaStopped;
 }
```

Conclusion

Le reste du code n'a absolument rien de spécifique à la création de contrôles (on pourrait tout à fait le retrouver dans le code d'un UserControl par exemple), et n'est donc pas traité en détail dans cet article. Vous pouvez cependant récupérer l'intégralité du code source sur site www.programmez.com.

Simon **Ferquel** - *Consultant Access it IDF, MVP Client App Dev*

Introduction à Metro



Windows Phone 7 est présenté comme la renaissance de la plateforme mobile Windows. Cela passe par de nombreux points techniques, ergonomiques et visuels. Redémarrer de zéro a permis d'améliorer de nombreux points de Windows Mobile 6.5, dont la cohérence et qualité étaient relatives.

Ainsi, un nouveau langage visuel a été développé pour l'occasion, baptisé Metro. Metro se veut une référence en termes de conception d'interfaces graphiques et d'interactivité. L'application de ces principes ergonomiques passe par une liste de recommandations ainsi que des fils rouges auxquels se référer lors de n'importe quelle étape d'un projet.

Vous ne serez cependant pas désorientés : il ne s'agit que d'une formalisation de bonnes pratiques de création et de conception. Les plus observateurs auront également remarqué que Metro est une évolution du travail commencé avec Windows Media Center et la branche de produits Zune (les baladeurs mais aussi le logiciel).

Au-delà du choc initial provoqué par l'aspect épuré, espacé, voire vide, vous remarquerez que le contenu est mis en avant : les images, les textes, les informations qui font l'intérêt de l'application que vous manipulez prennent le pas sur les décorations et enjoliveurs inutiles, qui peuvent parfois distraire l'œil de l'utilisateur de sa tâche première. Ce parti pris fort entre en opposition directe avec les plates-formes concurrentes qui imitent souvent des objets de la vie courante tels que des étagères en bois, des calepins avec des couvertures de cuir : **Metro est authentiquement numérique et favorise le contenu face au contenant.**

Une inspiration évidente

Comme son nom l'indique, Metro s'inspire de la signalétique des transports. Cette signalétique a pour avantage d'être universelle, simple, directe et non ambigüe. En effet, vous pouvez vous trouver à Paris, New-York, Moscou, Tokyo ou Hong-Kong, et arriver à comprendre les plans et lignes de métro, sans même comprendre la langue. La signalétique des transports est également conçue pour être lisible et compréhensible en mouvement, d'un coup d'œil et sans avoir à se concentrer : ces conditions correspondent parfaitement à l'utilisation d'un téléphone mobile. De telles qualités reposent sur des éléments très simples tels que la couleur, la taille, le poids et la famille des polices ainsi que les formes des éléments graphiques (ronds, carrés, ellipses).

Une disposition équilibrée

Pour arriver à conserver un impact important, il est primordial d'arriver à équilibrer la disposition de l'information. En design, quelques règles très simples permettent d'agencer des éléments tels que la grille, ainsi que l'espace négatif. Il n'est pas évident dans un premier temps de résister à la tentation de concentrer un maximum d'information dans un minimum d'espace. Souvent, une partie de cette information n'est pas de premier ordre et peut être reléguée à une interaction de l'utilisateur, un clic par exemple ou en utilisant le *progressive disclosure*. Sa disparition permet d'agencer différemment le noyau dur de l'information. Il est désormais possible d'agrandir la taille des titres, d'espacer les éléments, et d'octroyer un espace vide qui fait ressortir l'information présente. Afficher moins d'information permet de mieux présenter celle retenue.

Animations

Un autre aspect important de Metro, qu'il est impossible de cerner sans voir un téléphone en vidéo ou l'avoir entre ses mains est le mouvement. En effet, l'aspect simple de l'interface graphique est compensé en partie par l'intégration d'animations subtiles lors d'interaction ou lors de déplacements dans l'interface. Ces animations peuvent prendre la forme de translations, rotations, fade-in, fade-out, qui permettent une immersion accrue de l'utilisateur dans les tâches qu'il souhaite accomplir. Par exemple, la consultation d'une tâche dans la vue mensuelle du calendrier a pour effet de zoomer sur la case du jour afin d'afficher les détails des tâches.

Panorama et pivot

L'ensemble de ces principes converge sur Windows Phone 7 vers deux éléments graphiques aidant à disposer le contenu affiché par

une application : le panorama et le pivot. Tous deux tirent parti d'une navigation horizontale interne à la page, c'est-à-dire que les différentes sections composant une page sont agencées de gauche à droite, et qu'un mouvement glissé du doigt permet de passer d'une section à une autre. Une fois la dernière section atteinte, il reste possible de naviguer vers la droite, la section suivante rebouclant vers la première section. Les similarités s'arrêtent ici.

Panorama

Le panorama est utilisé dans tous les Hubs fournis avec le système : le Hub People (contacts), Images, Xbox LIVE, Office, … Sa principale fonction est d'afficher un contenu riche et engageant pour plonger l'utilisateur dans un *univers* unique. Ses différentes sections ont une même hauteur mais leur largeur peut varier si le contenu qu'elles doivent afficher nécessite plus d'espace, comme dans le cas des photos récentes du Hub Images par exemple. Il est parfois utile de considérer un panorama comme étant composé de 3 calques indépendants : titre, image de fond et contenu. L'image de fond, et notamment sa vitesse de défilement, différente de celle du contenu et de celle du titre, contribue à l'immersion dans l'expérience du Hub. Une immersion renforcée par l'affichage d'une petite partie de la section suivante, invitant l'utilisateur à découvrir le contenu supplémentaire et à naviguer de section en section. [Fig.1]

Les données affichées dans un panorama n'ont pas vocation à être exhaustives. En effet, dans la grande majorité des cas, les panoramas affichent des informations sélectionnées suivant différentes caractéristiques : leur fraîcheur, leur pertinence, leur fréquence de consultation ou d'utilisation. Par exemple, les fiches personnelles affichées dans la section « Récent » du Hub contact affichent les contacts avec lesquels j'ai pu interagir récemment, soit par appel, soit par SMS, soit par un réseau social tel que Twitter ou Facebook. Il est également possible de pousser du contenu à un utilisateur, qu'il soit publicitaire, adapté à son utilisation ou à ses goûts, etc. Le panorama peut également contenir un section-menu listant les différentes fonctionnalités de l'application qui auront pour effet de naviguer vers une autre page. Ainsi, il est en général conseillé de n'utiliser le panorama que comme page d'accueil d'une application. Le reste des pages de l'application étant soit de simples pages, soit des pages de type pivot.

Fig.1

Pivot

Les pages de type pivot se retrouvent également dans de nombreuses applications natives de l'OS, dans la page de réglage des paramètres comme dans l'application mail ou le calendrier [Fig. 2 et 3]. Il est possible de tracer un parallèle entre le pivot et un mécanisme d'interface graphique très utilisé en informatique depuis plusieurs décennies : les onglets.

Le principe est d'afficher les titres des différentes sections disponibles dans cette page en ligne en haut de la page, le titre le plus à gauche indiquant la section visible. Un geste de glissement horizontal ou la sélection d'un autre titre de section ayant pour effet de changer le contenu affiché.

Ceci permet de catégoriser des groupes hétérogènes de données facilement mais peut également servir de filtre sur un même jeu de données, comme par exemple dans l'application mail : voir tous ses messages, uniquement les nouveaux, ou ceux ayant déjà été étiquetés comme importants, etc.

Plusieurs différences sont à observer entre le pivot et le panorama : pas de visualisation d'une mince partie du contenu de la section suivante, pas d'image de fond défilant en parallèle de la navigation dans le contenu. Le pivot est un bon choix pour afficher une masse de données plus importante que le panorama, et peut être utilisé conjointement. Comme dans le hub People par exemple où la sélection d'un contact récent présenté dans le panorama, permet d'accéder à la fiche du contact, de type pivot et affichant toutes les informations reliées à un contact, de ses numéros de téléphone et son adresse jusqu'à ses plus récentes contributions à votre réseau social. [Fig.4]

Fig.2

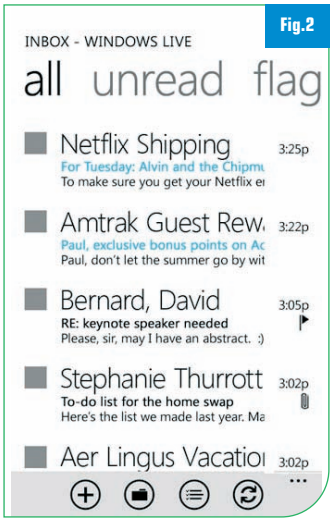


Fig.3

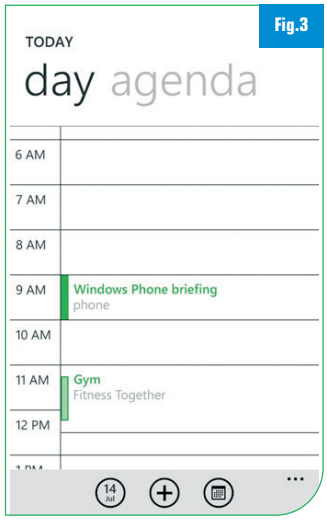


Fig.4

UI & SILVERLIGHT

Windows Phone 7 tire parti d'une technologie de présentation, Silverlight. Cette technologie, initialement dédiée aux interfaces riches (RIA), et déjà dans sa version 4, a été portée sur le nouveau téléphone. La grande majorité des fonctionnalités est disponible, tout en incorporant des spécificités propres à Windows Phone 7 : des styles de contrôles adaptés, une gestion des modes paysage et portrait ainsi qu'une gestion des mouvements multi-touch.

Les fondamentaux de Silverlight

Silverlight est un cousin proche de WPF (**W**indows **P**resentation **F**oundation), qui a redéfini la conception d'applications sur plateformes Microsoft, notamment en séparant la définition de l'interface graphique du code .Net s'occupant des traitements, mais également en apportant une gestion vectorielle des éléments. L'interface graphique est ainsi **déclarée** par un fichier XAML (e**X**tensible **A**pplication **M**arkup **L**anguage) compatible XML. Celui-ci définit tous les éléments graphiques, les conteneurs de positionnement (Grid, StackPanel, Canvas), les zones de texte (TextBlock), les animations des éléments, … Hormis les cas les plus complexes, il est rare de manipuler ces éléments dans du code C#. En effet, Silverlight met à disposition une technique très pratique pour connecter des données à leur zone d'affichage, le DataBinding. Il suffit de déterminer avec une syntaxe particulière dans le fichier XAML quel objet fournit l'information.

Utilisation de patrons d'affichage (templating)

Un des avantages de Silverlight et de WPF est de pouvoir facilement personnaliser n'importe quel élément graphique d'une page, qu'il soit un bouton, une boîte de saisie de texte, une barre de défilement ou une case à cocher (checkbox). Les précédentes technologies telles que Windows Forms étaient limitées à cet égard et toute personnalisation était fastidieuse.

Ainsi, une séparation forte a été introduite entre l'aspect de l'élément graphique et sa logique interne. Par exemple, l'aspect par défaut du bouton Windows Phone 7 est celui-ci : [Fig.5]

Style de bouton

Un bouton définit un certain nombre d'états dans lesquels il peut se trouver : *Normal, MouseOver* (inutilisé sur le téléphone puisqu'il n'y a pas de pointeur), *Pressed, Disabled, Unfocused* et *Focused.* Il est ensuite possible de définir quels éléments sont affichés dans un état particulier, de changer la couleur de fond, effectuer des rotations, virtuellement n'importe quelle modification sur les éléments graphiques. Silverlight fournit un outil de gestion de ces états appelé VSM (**V**isual **S**tate **M**anager), dont une des tâches est d'interpoler les animations à jouer entre les différents états.

De plus, le bouton expose des évènements indiquant les interactions qu'il subit, comme Click par exemple. Ceux-ci permettent de créer la logique interne à l'application et par exemple de connecter la pression d'un bouton avec une requête web.

Dans l'image [Fig.5], le deuxième bouton partage toutes les caractéristiques du bouton standard. Sa manipulation dans le code ne révèle aucune surprise, par exemple l'évènement Click conserve le même comportement. Son aspect graphique est différent : la bordure est plus fine et grise, une légère ombre portée est visible et donne une impression de profondeur, et le contenu du bouton reste affiché dans l'espace au centre.

Style d'éléments de liste

Des éléments graphiques plus complexes, tels que des listes peuvent également être restylés. Dans ce cas, Silverlight propose de changer plusieurs patrons d'affichage (templates) qui définissent l'aspect de différentes parties de l'élément graphique ListBox.

Le rôle d'une ListBox est d'afficher le contenu d'une liste d'éléments, un cas de figure rencontré très fréquemment. Pour ce faire, elle expose une propriété définissant la source de cette liste (ItemsSource) ; il faut ainsi connecter cette propriété à une liste d'éléments, étape appelée « binding ». Il est possible de distinguer plusieurs éléments d'affichage hiérarchisés : la zone dans laquelle seront affichés les éléments de sa source, une barre de défilement qui permet de voir l'ensemble de ces éléments s'il est impossible de les afficher dans l'espace disponible sur la page.

La disposition de ces éléments est définie par un patron de disposition (**ItemsPanel**). Par défaut, les éléments sont empilés verticalement (en utilisant un StackPanel), mais dans certains cas il sera préférable d'utiliser un patron disposant les éléments en ligne et sachant revenir à la ligne lorsque l'espace horizontal vient à manquer (WrapPanel disponible dans le Silverlight Toolkit http://silverlight.codeplex.com/). [Fig.6]

Le patron d'affichage le plus important de l'élément graphique liste est sans aucun doute celui dédié à l'affichage individuel des éléments (**ItemTemplate**). Par défaut, la liste ne sachant pas comment afficher l'élément, elle utilisera la simple représentation en

Fig.5

Bouton standard sur fond noir et style de bouton personnalisé

Fig.6

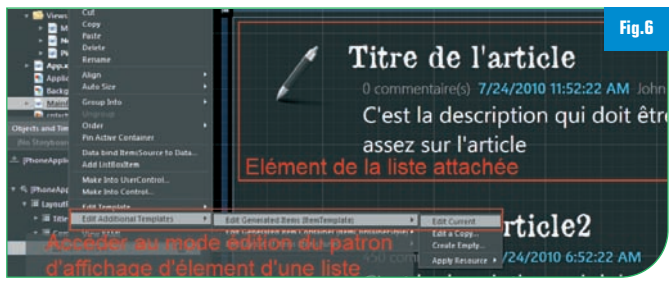


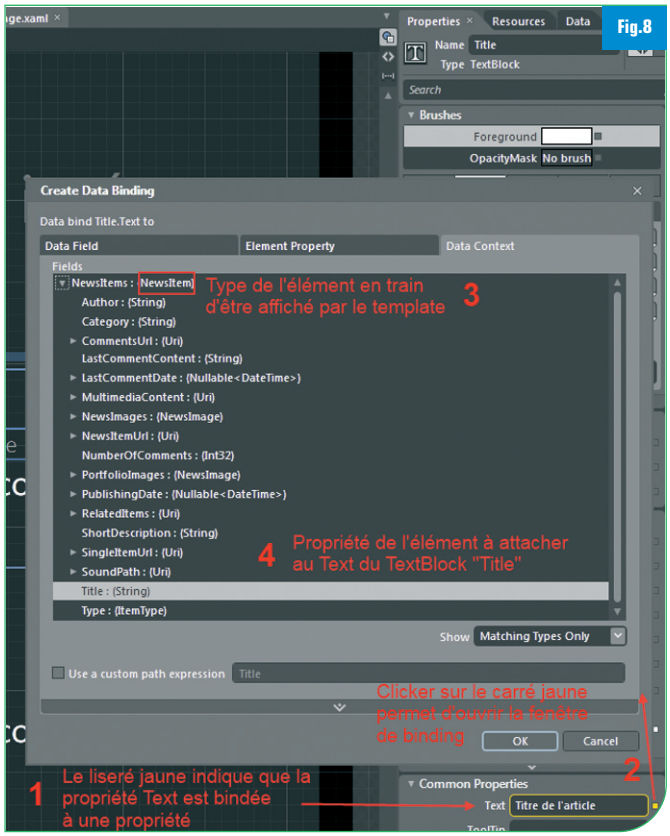
Fig.7

chaîne de caractères de l'élément (en exécutant la méthode ToString qui le plus souvent affiche le type qualifié de l'objet, MonApplication.Model.Article par exemple). [Fig.7]
Une fois rentré dans le mode d'édition du patron d'affichage de l'élément, vous pouvez décider de la meilleure façon de l'afficher. Le concept d'article (objet métier NewsItem) est très simple et comporte plusieurs informations (matérialisées dans le code par des propriétés) : titre, image d'illustration, description courte, auteur, date, nombre de commentaires.

```
    public class NewsItem
    {
  public string Title { get; set; }
  public DateTime? PublishingDate { get; set; }
  public List<NewsImage> NewsImages { get; set; }
  public string Author { get; set; }
  public int NumberOfComments { get; set; }
  public string ShortDescription { get; set; }

  public string Id { get; set; }
public string Category { get; set; }
public Uri NewsItemUrl { get; set; }
public Uri CommentsUrl { get; set; }
…
    }
```

Après avoir disposé les éléments sur la surface de design suivant leur importance, image à gauche, titre en haut avec une police de grande taille, etc., intervient l'étape de connexion entre les éléments graphiques et les informations de l'objet métier (binding) : [Fig.8]. Si les données présentes dans votre objet métier ne sont pas directement affichables, plusieurs solutions s'offrent à vous :

- créer des propriétés supplémentaires sur votre objet métier, la plupart du temps en lecture seule, qui s'occupent de transformer la valeur d'origine vers une représentation graphique acceptable
- utiliser un convertisseur de valeurs personnalisé



Convertisseur de valeurs

Un convertisseur de valeurs permet de transformer virtuellement n'importe quelle valeur d'origine vers une valeur compatible avec la propriété de destination de l'élément graphique sur lequel la donnée est attachée. Nous pourrions imaginer vouloir afficher un fond d'une couleur particulière suivant la catégorie de l'article : bleu pour les nouvelles internationales, vert pour les nouvelles sportives, gris pour l'économie… Dans ce cas, la valeur d'origine est la catégorie de l'article, mais la valeur transformée est un code couleur. Le convertisseur fonctionne dans les deux sens : affichage d'une donnée du modèle objet vers l'interface graphique (méthode Convert de l'interface IValueConverter) mais également la saisie d'une donnée depuis l'interface graphique et devant être traitée par l'application (méthode ConvertBack).
Après l'avoir déclaré dans les ressources de l'application, le plus souvent dans App.xaml, le convertisseur est ensuite utilisé dans le code XAML lors de l'étape de connexion des données (binding). Les convertisseurs peuvent également prendre un paramètre, à considérer comme une indication supplémentaire sur la façon dont on doit exécuter la transformation.
Voici un exemple simple de convertisseur permettant d'incorporer une valeur dans une chaîne formatée :

```
 public class StringFormatConverter : IValueConverter
 {
   public object Convert(object value, Type targetType,
object parameter,
    CultureInfo culture)
   {
     return string.Format(parameter as string, value);
   }

   public object ConvertBack(object value, Type targetType,
object parameter,
  CultureInfo culture)
   {
     throw new NotImplementedException();
   }
   }
```

La méthode ConvertBack n'est pas implémentée car cette conversion ne fonctionne que dans un sens pour afficher une donnée sur la page.
Ce convertisseur trivial agit sur la valeur en train d'être connectée et applique le formatage de texte passé en paramètre. La valeur peut être de n'importe quel format, du moment qu'elle peut s'afficher correctement en chaîne de caractères. Vous pouvez observer un exemple de son utilité dans le patron d'affichage de l'article plus haut, afin d'afficher la mention « 6 commentaire(s) ». Dans notre modèle métier, l'article NewsItem possède une chiffre indiquant le nombre de commentaires (NumberOfComments, typé int). Avec une connexion de données simple :

```
<TextBlock Text=»{Binding NumberOfComments}»/>
```

seul le chiffre serait représenté. Il serait alors nécessaire d'ajouter un autre élément texte (TextBlock) pour afficher « commentaire(s) ». En utilisant le convertisseur que nous avons créé de cette façon :

```
<TextBlock Text=»{Binding NumberOfComments, Converter={Static
Resource StringFormatConverter}, ConverterParameter=(\{0\})
commentaire(s)}»/>
```

nous obtenons le résultat attendu sans avoir à rajouter d'autre élément texte. A noter qu'une telle transformation par code se ferait ainsi :

```
string.Format(«({0}) commentaire(s)», value);
```

(Il est impératif d'échapper les accolades ouvrantes et fermantes (« \{ » et « \} ») lors du passage de paramètre dans le binding XAML pour ne pas interférer avec les accolades faisant partie de la syntaxe de binding)

LE WORKFLOW DESIGNER-DÉVELOPPEUR

Le data-binding ainsi qu'un certain nombre de fonctionnalités de Silverlight fait partie des techniques mises en place pour réaliser la nouvelle vision du processus de développement prônée par Microsoft. Ce workflow designer-développeur a pour but d'exploiter au mieux les compétences des différents acteurs du projet, et de mieux répartir leurs interventions lors de la vie du projet. Les designers s'occupent du design, de la charte graphique et de l'aspect créatif, les développeurs s'occupent du code, de sa robustesse, de la logique métier ainsi que des infrastructures d'accès aux données (e.g. webservices).

Un travail séquentiel

Historiquement, la création graphique et l'élaboration de l'ergonomie d'une application ont lieu lors des premières étapes du projet, en parallèle de l'analyse fonctionnelle des besoins. Une fois les écrans « désignés », souvent de façon statique (image), ils sont découpés et envoyés au développeur. Ce dernier fait ensuite de son mieux pour reconstituer l'aspect de l'écran tout en codant les fonctionnalités demandées et les éléments interactifs : les boutons, les listes, … Fréquemment, la sensibilité du développeur aux détails subtils du design est faible, et la conséquence directe est une approximation, voire un gouffre, entre l'aspect final de l'application interactive et les maquettes statiques initialement réalisées. Cette différence peut se caractériser par des tailles et styles de polices incorrects, des marges non respectées, des alignements approximatifs, … Ces divergences peuvent être détectées et corrigées lors de la recette créative de l'application qui a généralement lieu lors des dernières étapes du projet, mais elles exigent des allers-retours fréquents entre développeur et designer, à un moment où ce dernier n'est plus nécessairement disponible. Le constat est que le développeur passe un temps non négligeable à interpréter des signaux explicites et implicites d'une image statique. Ce travail ayant déjà été effectué par le designer.

Un travail itératif et continu avec Expression Blend

La suite d'outils Expression est apparue avec WPF en 2005. Un de ceux-ci a été conçu pour répondre à cette problématique forte de création, et tirer parti de la nouvelle architecture de présentation utilisant XAML : Expression Blend. Cet outil est désormais compatible avec Silverlight, et une version gratuite est disponible pour le développement d'applications Windows Phone 7.

Bien que cet outil ait été pensé comme un point d'intégration entre la partie technique pure incombant au développeur, et la partie créative incombant au designer, chacune des deux parties gagnerait à en apprendre les fonctionnalités qui ne lui sont pas directement dédiées. Par exemple, un développeur pourrait gagner un temps non négligeable en utilisant quotidiennement Blend en complément de Visual Studio lorsqu'il manipule les fichiers XAML de l'application. Une version plus basique de la surface de design restant disponible dans ce dernier outil.

La séparation de l'interface graphique de sa logique est matérialisée par deux fichiers : un fichier XAML, par exemple MainPage.xaml, et un fichier de code associé, par exemple MainPage.xaml.cs. Ainsi il est généralement accepté que le fichier XAML tombe sous la responsabilité du designer tandis que le fichier C# tombe sous celle du développeur. L'un et l'autre peuvent cependant intervenir brièvement sur l'un ou l'autre fichier.

Le principe de l'élément graphique impersonnel (faceless control) permet de changer l'aspect complet d'un élément graphique sans changer ses fonctionnalités ni ses états. Cette étape de « lifting » est réalisable sans toucher une ligne de code, en utilisant des patrons d'affichage personnalisés (templates) et des styles définis dans le fichier XAML. Le designer peut ainsi travailler tranquillement sur l'aspect des boutons, la façon dont il faut présenter les éléments d'une liste ainsi que leur disposition, sans interférer avec le travail du développeur connectant l'application mobile à un service web.

Naturellement, certains pré-requis sont nécessaires, et ces points d'intégration doivent être fixés au plus vite pour fluidifier les interactions designer-développeur :

- définir au plus vite les objets métier, leurs propriétés et leurs types ; il est capital que le designer soit capable de comprendre ces unités d'information lorsqu'il réalise l'interface graphique et connecte ces informations aux éléments de l'interface (binding) ; cette responsabilité incombe au développeur car cet objet métier est matérialisé par une classe C# ; l'étape de binding peut cependant être réalisée plus tard
- définir la source des données : il est à la charge du développeur de choisir comment exposer les données au designer ; plusieurs choix s'offrent à lui : l'utilisation directe de la propriété DataContext de la page, ou bien l'utilisation d'un framework MVVM (**M**odel-**V**iew-**V**iew**M**odel) qui se chargera de lui simplifier la tâche pour le data-binding tout en clarifiant l'architecture de l'application
- constituer un jeu de données de test, actif uniquement lors de l'édition des pages sous l'outil de design Blend afin que le designer puisse travailler avec du contenu lors de la création des patrons d'affichage

Guillem **Mazarico**
Backélite

Mon premier jeu mobile avec XNA

Dans ce dossier nous allons apprendre à élaborer un jeu 2D. Nous verrons les différentes étapes de réalisation d'un « frogger », fameux jeu où l'on doit faire traverser une route à une grenouille en évitant différents obstacles.

Avant de commencer le développement proprement dit, je tiens à préciser que toutes les images utilisées dans cet article sont tirées d'un kit de démarrage disponible sur http://creators.xna.com, le site officiel de Microsoft pour cette technologie. Commençons par créer notre projet. Tout d'abord, ouvrez Visual studio 2010. Ensuite faites fichier -> nouveau projet. Là, choisissez Windows Phone Game (4.0) et donnez un nom à votre projet.
Cliquez sur OK. Vous devriez avoir une classe Game1.cs qui se crée automatiquement et qui contient un squelette primaire de votre jeu. Nous n'allons pas nous occuper tout de suite de cette classe puisqu'elle nous servira à gérer les différents composants de notre jeu.
Commençons par créer l'environnement de jeu qui sera composé d'une seule image de fond représentant le décor. Cet environnement sera défini dans une nouvelle classe. Faites un clic droit sur Frogger dans la barre d'outils sur votre droite puis ajouter, et enfin nouvel élément [Fig.2].
Une fois dans l'assistant choisissez Game Component, nommez-la comme vous voulez, Decor.cs par exemple, puis faites ajouter. Une nouvelle classe s'affiche sur votre écran.
Decor.cs aura pour but de permettre l'affichage du décor. Nous devons donc transformer notre classe qui est une Game Component en DrawableGameComponent : il suffit de changer l'héritage de la classe comme ceci.

```
public classLevel: Microsoft.Xna.Framework.GameComponent
```

devient :

```
public classLevel: Microsoft.Xna.Framework.DrawableGameComponent
```

Déclarons maintenant les variables dont nous allons avoir besoin :

```
Game1 game;//servira à stocker une référence vers la classe de départ

SpriteBatch spriteBg;
Texture2D txtBg;//texture du fond
```

Maintenant, modifions le constructeur de notre classe Decor de manière à lui envoyer les informations dont il aura besoin sur le jeu.

```
public Decor (Frogger game)
    : base(game)
{
    this.game = game;
}// end Level
```

Nous allons maintenant préparer un spritebatch. Le SpriteBatch permet de traiter par lot l'affichage des sprites, et ainsi améliorer les performances.. Nous allons assigner une texture à un spritebatch puis nous allons choisir ou et comment afficher ce dernier. Leur initialisation se fait dans la méthode initialize comme ceci.

```
public override void Initialize()
{
    spriteBg = new SpriteBatch(m_graphics.GraphicsDevice);
decorRect = new Rectangle(0, 0, game.GraphicsDevice.Display
Mode.Width, game.GraphicsDevice.DisplayMode.Height);
    base.Initialize();
}// end Initialisation
```

Nous profitons de la méthode initialize pour créer un rectangle dans lequel afficher notre texture. Le décor prenant tout le fond, le rectangle a la taille de l'écran de notre appareil. Il ne faut pas oublier que dans XNA tout comme DirectX le point 0,0 se situe en haut à gauche de l'écran.
Ensuite, nous allons charger les textures : commençons par les ajouter au projet. Faites un clic droit sur le projet FroggerContent dans la barre d'outils, puis ajoutez *éléments existants*. Vous pouvez bien entendu créer des dossiers pour ranger vos textures. Chargez maintenant les textures grâce à la fonction LoadContent.

```
protected override void LoadContent()
{
 txtBg = game.ContentManager.Load<Texture2D>(«Textures/Decor»);
 base.LoadContent();
}// end LoadContent
```

Il n'est pas nécessaire de spécifier l'extension du fichier. Nous n'utiliserons pas la méthode update puisque le décor ne change pas au cours du temps, vous pouvez la supprimer si vous voulez.
Nous allons maintenant modifier la fonction Draw afin d'afficher notre texture.

```
void Draw(GameTime gameTime)
{
 spriteDecor.Begin();
 spriteDecor.Draw(txtDecor, decorRect, Color.White);
```

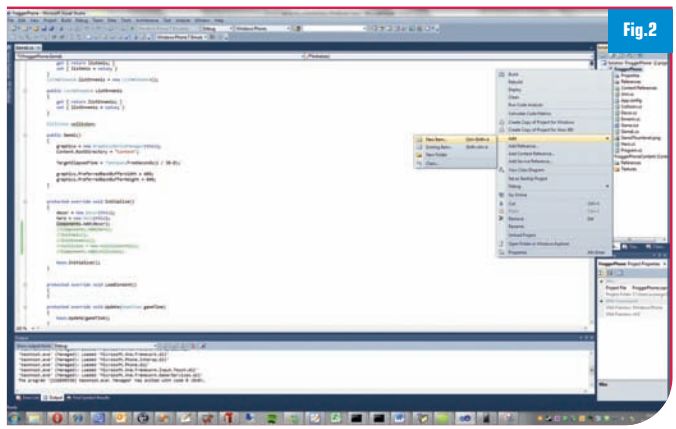


Fig.2

```
  spriteDecor.End();

}// end DrawBg
```

Pour dessiner un sprite il y a 3 étapes :
1 - begin
2 - on le dessine avec la texture voulue dans un rectangle
3 - on lui donne une teinte.
Cette dernière option est très utile pour avoir des ennemis de couleurs différentes sans avoir à charger des textures en plus. Nous en avons fini avec notre classe Decor : il faut maintenant l'ajouter aux éléments du jeu. Cela se passe dans le fichier Game1.cs. Notre classe principale servira de chapeau aux autres classes, c'est ici que nous allons définir tous les éléments généraux de notre jeu, comme par exemple la taille du backbuffer.

```
public Game1()
{
  graphics = new GraphicsDeviceManager(this);
  Content.RootDirectory = «Content»;
  graphics.PreferredBackBufferHeight = 800; //hauteur du backbuffer
  graphics.PreferredBackBufferWidth = 480; //largeur du backbuffer
}
```

Cette résolution est la résolution standard pour les écrans des Windows Phone, vous pouvez la réduire afin de gagner en performance de manière tout à fait invisible (tant que vous conservez le format), le hardware se chargeant d'upscaler votre jeu à la résolution native du téléphone pour vous. Pour appeler notre Décor dans notre classe principale nous avons besoin de créer une variable Decor, de l'instancier, puis de l'ajouter aux composants de notre jeu.

```
protected override void Initialize()
{
  decor = new Decor(this);
  Components.Add(decor);
  base.Initialize();
}
```

Une des particularités d'XNA est qu'une fois qu'un GameComponent ou un DrawableGameComponent est ajouté en tant que composant dans une classe Game, les méthodes Update et Draw se synchronisent entre elles. Après compilation et déploiement de votre jeu sur l'émulateur votre fond devrait s'afficher.

Déplacement et animation d'un personnage

Réaliser l'animation d'un élément est relativement simple, c'est le même que celui des dessins animés. Nous allons afficher les images composant l'animation les unes à la suite des autres. Nous allons donc utiliser une image, une « planche de sprites », qui va contenir toutes les étapes de notre animation. Il nous faudra découper la partie de l'image nous intéressant et l'afficher dans notre spritebatch.
Voici un exemple de planche de sprite : [Fig.5]
On commence par créer la classe représentant notre héros. Les étapes de chargement de l'image étant les même que pour le décor je ne vous les réexplique pas. Voici les variables dont nous allons avoir besoin pour notre personnage :

```
Game1 game;
private Vector2 pospers; // position du personnage

public Vector2 Pospers
{
  get { return pospers; }
  set { pospers = value; }
}
Texture2D txtPerso_up; //texture du personnage se déplaçant vers le haut
Texture2D txtPerso_down; //texture du personnage se déplaçant vers le bas
Texture2D txtPerso_left; //texture du personnage se déplaçant vers la gauche
Texture2D txtPerso_right; //texture du personnage se déplaçant vers la droite
Texture2D txtPerso_idle; //texture du personnage ne bougeant pas
Texture2D txtPerso; //texture tampon
Texture2D txtPerso_Dead; //texture du personage mort
SpriteBatch spriteHero;
bool dead = false;

public bool Dead
{
  get { return dead; }
  set { dead = value; }
}
Rectangle rectCol = new Rectangle(); //rectangle utilisé pour la gestion des
collisions

public Rectangle RectCol
{
  get { return rectCol; }
  set { rectCol = value; }
}
```

Et celles nécessaires à l'animation :

```
float Timer = 0f;//timer de l'animation
float Interval = 1000f / 10f;//intervalle de changement de l'image
int FrameCount = 5;//nombre de frame dans notre animation
int CurrentFrame = 0;//frame de départ
Rectangle SourceRect;//rectangle découpé dans notre image
Rectangle DestRect;//rectangle ou nous allons afficher l'image
```

Voici la fonction d'animation extrait (voir **code complet** sur le site) :

```
void TimerSprite(GameTime gameTime, int SpriteWidth, int Sprite
Height, int ImgWidht, int ImgHeight)
```

Notre fonction prend en paramètre un gamertime qui nous permet de connaître le temps écoulé depuis le lancement de notre jeu, il nous permettra de gérer la vitesse de notre animation. Il prend ensuite la taille du sprite contenant l'animation, puis la taille du rectangle dans lequel l'image va être affichée. Le timer est incrémenté grâce au gametime et à chaque fois que celui-ci dépasse un intervalle donné, nous passons à la frame (partie du sprite) suivante. Quand nous arrivons à la dernière frame nous repartons au début de notre animation. SourceRect représente le rectangle que nous devrons découper dans notre sprite, et DestRect représente le rectangle ou l'image sera affichée. Maintenant que nous avons notre fonction d'animation et que nous avons initialisé nos textures et nos sprites, nous allons passer à la fonction Draw. Notre fonction d'affichage utilisera deux sous-fonctions différentes. La premiè-

re correspondant à l'affichage de notre personnage mort, la deuxième à l'affichage de notre personnage vivant.

```
void Draw(GameTime gametime, Texture2D txtPerso)
{
   spritePerso.Begin();
 if (!dead)
     spriteHero.Draw(txtPerso, DestRect, SourceRect, Color.White);
   else
     spriteHero.Draw(txtPerso, pospers, Color.White);

   spritePerso.End();
}// end DrawPerso
```

Le premier Draw est une surcharge de la fonction de départ, celle-ci prend en paramètre un rectangle de plus, permettant de découper une partie de la texture pour l'afficher.
Passons maintenant à la fonction initialize où nous initialiserons la position de notre personnage, ici en bas de l'écran et au milieu.

```
public override void Initialize()
{
  pospers.X = game.GraphicsDevice.DisplayMode.Width / 2;
  pospers.Y = 750;
  spriteHero = new SpriteBatch(game.GraphicsDevice);

  base.Initialize();
}
```

Pour déplacer notre grenouille nous allons avoir besoin de récupérer les points sur l'écran ou le joueur appuiera. Nous allons utiliser la fonction suivante qui nous renvoie une collection contenant tous les points de contact de l'écran.

```
TouchCollection touches = TouchPanel.GetState();
```

Si la collection n'est pas vide nous récupérons les coordonnées du premier point d'appui (le jeu n'étant pas multitouch). Nous devons maintenant modifier les coordonnées de la position de notre personnage en fonction de l'endroit où l'on appuie à l'écran, tout en prenant garde à ce qu'il ne sorte pas de l'écran (Code source complet sur notre site).
Selon l'endroit de l'écran sur lequel nous appuierons, nous déplacerons la position de notre personnage dans la direction voulue, puis nous assigneront à txtperso le bon sprite d'animation et enfin nous appellerons la fonction d'animation du personnage avec les valeurs voulues.

Les ennemis ou comment implémenter un comportement automatique

Nous venons de voir comment déplacer et animer un sprite a l'écran en utilisant l'écran tactile du téléphone. Nous allons maintenant implémenter la classe représentant nos ennemis. Dans le cas du frogger, ce sont juste des personnages qui apparaissent de chaque côté de l'écran pour essayer de vous écraser, mais nous allons tenter de faire en sorte de varier leur vitesse afin de rajouter un peu de piquant. Tout d'abord créez votre classe ennemie, comme vous l'avez fait pour le héros et le décor.
Commençons par nous attaquer au constructeur de notre classe ennemi, il sera de la forme suivante :

```
public Ennemi(Game1 game, int coordX, int coordY, float vitesse,
string direction)
  : base(game)
{
  this.game = game;
  pospers.X = coordX;
  pospers.Y = coordY;
  this.direction = direction;
  this.vitesse = vitesse;
}
```

Les principales différences avec le constructeur de notre classe héros sont les suivantes. Tout d'abord, nous appelons notre classe avec deux coordonnées qui seront les coordonnées de départ de notre ennemi. Ensuite, nous avons un entier représentant la vitesse avec laquelle se déplacera l'ennemi. Et enfin une chaîne de caractère permettant de savoir si notre ennemi se déplacera vers la gauche ou vers la droite. Nous pourrons ainsi créer une multitude d'ennemis différents avec un seul constructeur de classes.
Dans notre initialize Nous allons mettre à jour la vitesse, en fonction de sa direction, et la position à laquelle notre ennemi devra réapparaître quand il aura traversé l'écran.

```
public override void Initialize()
{
  spritePerso = new SpriteBatch(game.GraphicsDevice);
  if (direction == «g»)
  {
    posInit.X = game.GraphicsDevice.DisplayMode.Width;
  }
  if (direction == «d»)
  {
    vitesse = (-vitesse);
    posInit.X = 0;
  }
  base.Initialize();
}
```

Passons maintenant au déplacement de nos ennemis, comme nous avons vu précédemment, la méthode update est la méthode qui nous permet de mettre à jour nos variables en fonction du temps. Nous allons faire en sorte que les coordonnées soient

Fig.5

mises à jour en fonction de la vitesse que nous avons choisie.
Nous devrons aussi tester si nos ennemis sortent de l'écran, si c'est le cas, nous les renvoyons à leur position de départ. Ainsi il y aura toujours des ennemis pour vous barrer le chemin. Voici le code permettant de réinitialiser la position de nos ennemis.

```
pospers.X = pospers.X - (vitesse);
if (direction == «g»)
{
 if (pospers.X < 0 - SpriteWidth)
   pospers.X = posInit.X;
}
if (direction == «d»)
{
 if (pospers.X > m_graphics.PreferredBackBufferWidth + Sprite
Width)
   pospers.X = posInit.X;
}
```

Une fois ceci fait, n'oubliez pas d'ajouter vos ennemis à votre jeu. Comme nous en avons beaucoup, je les ai stockés dans une liste afin de pouvoir accéder facilement à leurs données. Voici la fonction que j'utilise, vous pouvez bien entendu ajouter autant d'ennemis que vous le voulez (Code complet sur le site) :

```
public void InitEnnemis()
 {
   listEnnemis.Add(new Ennemi(this, GraphicsDevice.Display
Mode.Width, GraphicsDevice.DisplayMode.Height - 150, 1, "g"));
   listEnnemis.Add(new Ennemi(this, GraphicsDevice.Display
Mode.Width, GraphicsDevice.DisplayMode.Height - 275, 3, "d"));
   listEnnemis.Add(new Ennemi(this, GraphicsDevice.Display
Mode.Width, GraphicsDevice.DisplayMode.Height - 525, (float)
2.5, "d"));
   listEnnemis.Add(new Ennemi(this, GraphicsDevice.Display
Mode.Width, GraphicsDevice.DisplayMode.Height - 400, 2, "g"));

   foreach (Ennemi ennemi in ListEnnemis)
   {
     Components.Add(ennemi);
   }
 }
```

Ébauche d'un moteur physique

Dans un jeu, le moteur physique régit toutes les interactions entre nos différents éléments. Ici il servira à vérifier si notre héros rencontre un ami ou un ennemi. Avant de commencer la gestion des collisions nous allons d'abord créer les points d'arrivée de notre grenouille, en effet dans le frogger vous devez passer par cinq points situés tout en haut du niveau pour passer au suivant. Ici ce seront des amis à aller sauver.
Commençons par créer une classe Ami. Ensuite nous faisons exactement comme pour le héros ou les ennemis, mais sans logique de déplacements, Juste un affichage a une position donnée, vous pouvez les animer ou non. Le sprite change lorsque vous les avez sauvés, mais comme cela ne représente pas une nouveauté je ne vous détaillerai pas le code, vous le retrouverez dans l'archive finale. N'oubliez pas d'ajouter vos amis au jeu principal, de la même manière que pour les ennemis.
Il faut maintenant passer à la gestion des collisions. La première chose à faire pour pouvoir savoir si deux éléments se rencontrent va être de connaître leur position. Pour cela, nous allons créer dans chaque classe d'élément interactif, un rectangle auquel nous pourrons accéder depuis notre classe Game1.

```
 public Rectangle RectCol
 {
   get { return rectCol; }
   set { rectCol = value; }
 }
```

La prochaine étape consiste à mettre à jour ce rectangle de collision, de manière à connaître à chaque instant leur position exacte. Nous allons mettre à jour les coordonnées du rectangle dans la méthode update, comme ceci :

```
rectCol = new Rectangle((int)pospers.X, (int)pospers.Y, 40, 24);
```

Ici, je ne gère que la collision sur la moitié de la hauteur du personnage, de manière à simuler le fait qu'il se tient debout. Nous allons gérer les collisions dans une classe collision qui sera un gamecomponent, puisqu'elle n'aura pas besoin d'affichage. Pour que les collisions aient un effet, il faut que nous puissions tuer notre grenouille. Plaçons-nous donc dans notre classe héros nous allons utiliser le booléen dead, initialisé à faux.

```
public int Dead
{
    get { return dead; }
    set { dead = value; }
}
```

Il nous faut aussi empêcher notre héros de bouger s'il est mort. Il nous faut rajouter une condition (!Dead) dans la méthode update ou nous gérons ses déplacements. Nous avons déjà pris en charge l'affichage du sprite de circonstance.
Maintenant que nous sommes revenus dans notre classe Collision, Nous allons vérifier si les rectangles de notre héros et celui de nos ennemis ou amis se superposent. Nous allons utiliser la méthode intersect() qui détermine automatiquement si les pixels de deux rectangles se confondent.
Dans notre méthode Update nous parcourons nos deux listes, amis et ennemis de la classe principale. Si notre héros rencontre un ami, le héros sera téléporté à ses coordonnées de départ et l'ami sera sauvé. S'il rencontre au contraire un ennemi, il sera tué.
Une fois la classe de collision ajoutée à votre classe Game1, vous avez un jeu simple mais complet. Mais, nous n'avons vu qu'une toute petite partie des possibilités que nous offre XNA. Le framework propose bien d'autres facilités comme une gestion des sons, de l'accéléromètre, et même des connexions réseaux (même si elle est limitée sur Windows Phone 7).

■ Pierre-Yves **Gardette**
Fondateur d'XNA-France
http://www.xna-france.com
Microsoft Student Partner

Créer un univers en 3D

Dans cet article nous allons examiner les notions de base de la programmation 3D et la manière de développemer un univers 3D avec XNA. Nous verrons tout particulièrement la gestion d'un terrain en trois dimensions et comment s'y déplacer.

NOTIONS DE BASE SUR L'AFFICHAGE D'UN OBJET

Un triangle, des vertices

Si en 2D afficher un objet revient à «plaquer» une image à l'écran, en 3D la donne est différente. Les objets ne sont pas dessinés à partir d'une image mais plutôt à partir de vertices (un vertex, des vertices). Un vertex peut être vu comme un emplacement dans l'espace auquel on associe des propriétés (couleur, normale, position …). Le pipeline 3D va relier ces points pour former un objet et va plaquer cet objet à l'écran. Le développeur a pour tâche de bien placer ces points et d'indiquer au device comment les relier.

L'image [Fig.1] illustre cela. Nous voyons un tube et un cube en trois dimensions. On remarque qu'ils sont tout simplement formés à partir de points qui sont reliés entre eux. La forme géométrique la plus simple en 3D est le triangle. Trois vertices sont nécessaires pour la former. Toute forme géométrique en 3D est formée de triangles. Le Framework Xna dialogue sur ce principe avec la carte graphique. Il « raisonne » en ternaire ; Il lui faut trois points pour former la plus petite figure géométrique possible : le triangle. Il lui faut X triangles pour former un objet 3D plus évolué.

Les effets

Le pipeline 3D a pour principale tâche d'afficher sur un écran en 2 dimensions une scène en 3 dimensions. Les instructions qui réalisent cette tâche se présentent sous la forme d'un fichier Effet. Il est très simple de développer un fichier Effet. Un langage existe, le HLSL, relativement proche du C et Microsoft offre gracieusement un compilateur. Mais nous n'en n'aurons pas besoin pour cet article, dans la mesure où le Framework Xna propose n ensembles de fichiers pré compilés et prêts à l'emploi pour nos jeux.

Pour passer d'un objet 3D à un ensemble de pixels sur un écran 2D, le fichier Effet utilise 3 matrices. Une matrice est une sorte de boîte magique mathématique, particulièrement véloce pour effectuer des calculs trigonométriques (si importants en 3D). La première matrice utilisée est appelée par convention World. Cette matrice applique une transformation à ou aux objets à afficher (par transformations nous entendons, translation, rotation, et homothétie). La seconde matrice, View, précise les propriétés de la « caméra » avec laquelle on visionne la scène 3D. La troisième, Projection indique les caractéristiques de l'écran sur lequel la scène va être affichée. Avec ces trois matrices, le fichier Effet pourra effectuer son rendu.

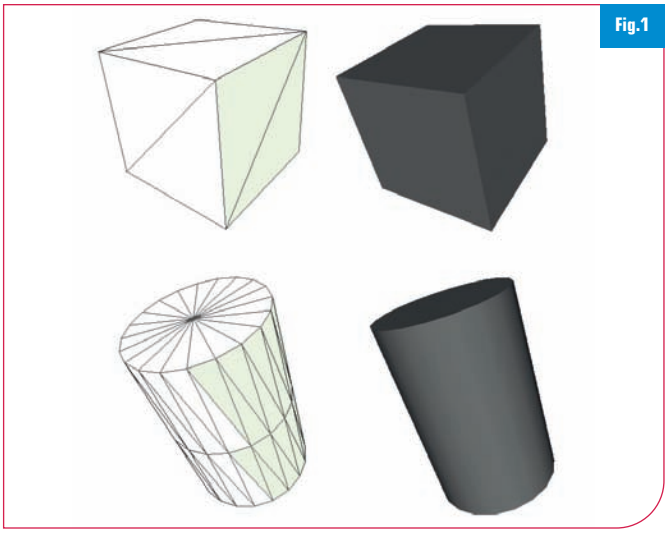


Fig.1

La pratique !

Commençons par créer un fichier Effet en ajoutant le membre suivant à la classe Game1 :

```
BasicEffect _effet;
```

Ajoutez ensuite à la méthode LoadContent de la classe Game1 l'instruction :

```
_effet = new BasicEffect(this.GraphicsDevice);
```

Le fichier Effet est créé et prêt à l'emploi. Nous avons vu au tout début qu'un vertex était un point dans l'espace auquel on associait un ensemble d'informations liées à son affichage dans l'espace. Le but de ce tutorial est d'afficher un triangle coloré à l'écran. Chaque point dans l'espace (vertex) doit donc être associé à une position et une couleur. Le développeur peut créer sa propre structure de données encapsulant toutes ces informations mais le framework Xna en possède une qui répond à nos besoins : la structure VertexPositionColor.

Un triangle est composé de trois points, il faut donc créer un tableau de trois cases de type VertexPositionColor. Ajoutez la déclaration suivante en début de classe :

```
VertexPositionColor[] vertices;
```

Puis, dans la méthode Initialize nous allons créer trois vertices et l'affecter aux trois cases de ce tableau.

```
vertices = new VertexPositionColor[3];
vertices[2].Position = new Vector3(-0.5f, -0.5f, 0f);
vertices[2].Color = Color.Red;
vertices[1].Position = new Vector3(0, 0.5f, 0f);
vertices[1].Color = Color.Green;
vertices[0].Position = new Vector3(0.5f, -0.5f, 0f);
vertices[0].Color = Color.Yellow;
```

Ici le code est plutôt simple : trois vertices sont créés. Chacun deux se voit affecter une position et une couleur. Nous sommes

dans l'espace 3D. Une position correspond donc à trois composantes X, Y, Z (repère euclidien orthonormé dans l'espace). Modifions maintenant la méthode Draw en ajoutant une instruction de dessin :

```
this.graphics.GraphicsDevice.DrawUserPrimitives(PrimitiveType.
TriangleList, vertices, 0, 1);
```

DrawUserPrimitive prend en premier paramètre le type de reliaison à faire entre les vertices pour former des triangles. Ici, avec TriangleList, la carte graphique utilisera les vertices du tableau passé en paramètre par triplets pour former ses triangles. Les deux derniers paramètres précisent l'index à partir duquel le tableau de vertices doit être lu et le nombre de triangles à afficher. Lancez la compilation. Vous devriez obtenir une erreur indiquant plus ou moins clairement qu'un fichier Effet n'a pas été spécifié et que la carte graphique ne sera pas en mesure d'afficher notre modèle. Remplacez donc l'instruction précédente par le code suivant :

```
foreach (EffectPass pass in this._effect.CurrentTechnique.
Passes)
{
       pass.Apply();
      this.graphics.GraphicsDevice.DrawUserPrimitives(Primitive
Type.TriangleList, vertices, 0, 1);
}
```

Un fichier Effet peut abriter plusieurs « techniques » de rendu d'objet 3D, chacune pouvant lui donner un aspect différent. Ici nous prenons la technique par défaut de ce fichier. Appliquer un effet peut demander plusieurs couches ou «passes». Ici la passe en cours s'appliquera pour tous les objets situés en aval de l'appel à Apply(). Mais ce n'est pas fini, nous devons encore préciser les trois matrices que nous avons présentées en début d'article. Préfixez la boucle foreach ci-dessus par les 4 instructions suivantes :

```
this._effect.VertexColorEnabled = true;
this._effect.World = Matrix.Identity;
this._effect.View = Matrix.CreateLookAt(new Vector3(0, 0, -2),
Vector3.Zero, Vector3.Up);
```

Fig.2

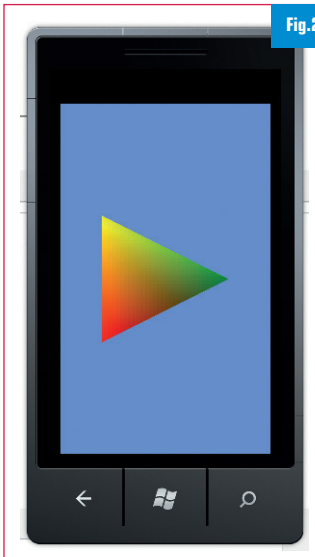


Fig.3

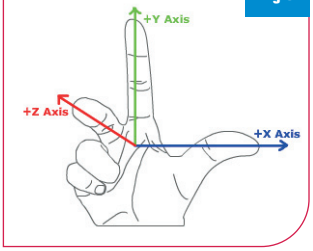
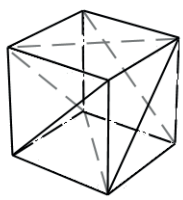


Fig.4



```
this._effect.Projection = Matrix.CreatePerspectiveFieldOf
View(MathHelper.PiOver4, this.GraphicsDevice.Viewport.Aspect
Ratio, 0.1f, 100f);
```

Nous autorisons tout d'abord le fichier Effet à afficher les couleurs des vertices. Pour la matrice World, nous indiquons qu'il n'y aura pas de transformation (Identity est l'élément neutre d'une transformation). Pour la matrice View nous utilisons la méthode CreateLookAt de la classe Matrix afin de préciser :
la position de la caméra (ici la caméra est simplement reculée de deux unités en profondeur) l'endroit vers lequel on regarde (ici on regarde vers l'origine (0, 0, 0)) l'angle de la caméra avec l'horizontale (ici angle droit).
Enfin la matrice Projection spécifie un angle d'ouverture de 45° (PI/4), le ratio de l'écran (largeur de l'écran sur sa hauteur), et enfin deux valeurs importantes : la distance à la caméra à partir de laquelle on peut voir un objet, et la distance au-delà de laquelle un objet n'est plus visible (front plane et far plane). Là encore une méthode de la classe Matrix nommée CreatePerspectiveFieldOfView permet de créer une matrice facilement. A noter que le ratio permet de garder un aspect cohérent pour notre scène en cours d'affichage quelle que soit la résolution de la zone d'affichage. Si vous relancez le programme vous obtenez l'écran suivant : [Fig.2].

Un p'ti cube ?

Le *monde de jeu* se trouve dans un espace 3D orthonormé. A l'intérieur de celui-ci tout point est situé par l'intermédiaire de trois composantes : sa position par rapport à l'abscisse X, sa position par rapport à l'ordonnée Y, sa position par rapport à la cote Z. En repère 2D X et Y sont facilement identifiables : l'axe X est horizontal et Y vertical. En 3D «XNA» on se base plutôt par rapport à un repère dit de «main droite» : [Fig.3]
Ce nom vient du fait que vous pouvez reproduire ce repère à l'aide de votre main droite. Le pouce représente l'axe X, l'index l'axe Y et le majeur l'axe Z. A l'intérieur du *monde de jeu* ou World Space nous placerons désormais nos objets en utilisant des coordonnées 3D liées à cette représentation. Ainsi Z croît lorsqu'il se rapproche de l'observateur. L'axe X est orienté vers la droite, enfin l'axe Y croît avec la hauteur.
Pour afficher un cube, il va nous falloir pas moins de 12 triangles pour former un cube ! (un cube = 6 faces, une face = 2 triangles). Nous commencerons donc à réaliser l'affichage d'une face, puis de deux, et enfin le cube en entier. Nous appliquerons une transformation propre au cube pour terminer. L'image suivante explicite les douze triangles qui forment le cube [Fig.4].
Les traits noirs représentent les faces, les traits gris les triangles. C'est ce résultat que nous devons reproduire. La première étape consiste à modifier notre triangle afin de le rendre rectangle (triangle à angle droit). Modifiez la déclaration des vertices ainsi :

```
vertices = new VertexPositionColor[3];
vertices[0].Position = new Vector3(-10f, -10f, 10f);
vertices[0].Color = Color.Green;
vertices[1].Position = new Vector3(-10f, 10f, 10f);
vertices[1].Color = Color.Red;
vertices[2].Position = new Vector3(10f, 10f, 10f);
vertices[2].Color = Color.Yellow;
```

Les trois points du triangle on la même profondeur (10f) ceci, afin de rendre la définition des positions des vertices plus simple à imager. L'image ci-dessous résume le travail que nous venons de faire (les numéros en rouge correspondant à l'index des vertices dans le tableau vertices) : [Fig.5].
Les dimensions de notre triangle sont bien plus importantes que pour notre premier triangle. Il faut donc replacer la caméra en conséquence. Modifiez les instructions d'affectation aux matrices View et Projection ainsi :

```
this._effect.View = Matrix.CreateLookAt(new Vector3(0, 10, 50),
Vector3.Zero, Vector3.Up);
this._effect.Projection  =  Matrix.CreatePerspectiveFieldOf
View(MathHelper.PiOver4, this.GraphicsDevice.Viewport.Aspect
Ratio, 0.1f, 1000f);
```

Nous nous plaçons en face du triangle que nous venons de créer à 50 unités de distance et avec une légère vue de haut de 20 unités. Si vous lancez le programme à ce stade vous obtenez le résultat : [Fig.6].
Ajoutons une difficulté en créant un nouveau triangle collé au premier de manière à former une face carrée ! Modifiez la création du tableau de vertices en donnant une taille de 6 et en ajoutant les trois vertices suivants :

```
vertices[3].Position = new Vector3(-10f, -10f, 10f);
vertices[3].Color = Color.Green;
vertices[4].Position = new Vector3(10f, 10f, 10f);
vertices[4].Color = Color.Red;
vertices[5].Position = new Vector3(10f, -10f, 10f);
vertices[5].Color = Color.Yellow;
```

Nous avons maintenant 6 vertices. Deux triangles isocèles à angle droit et donc une face. Schématiquement nous avons réalisé ceci : [Fig.7]. Nous n'affichons donc plus une seule primitive (triangle) mais deux. Il faut donc aussi modifier l'instruction de dessin dans la méthode Draw :

```
this.graphics.GraphicsDevice.DrawUserPrimitives(PrimitiveTy
pe.TriangleList, vertices, 0, 2);
```

Ajoutons maintenant une nouvelle face :

```
//triangle 3, face droite
vertices[ 6 ].Position = new Vector3(10f, -10f, 10f);
vertices[ 6 ].Color = Color.Green;
vertices[7].Position = new Vector3(10f, 10f, 10f);
vertices[7].Color = Color.Red;
vertices[ 8 ].Position = new Vector3(10f, 10f, -10f);
vertices[ 8 ].Color = Color.Yellow;

//triangle 4, face droite
vertices[9].Position = new Vector3(10f, -10f, 10f);
vertices[9].Color = Color.Green;
vertices[10].Position = new Vector3(10f, 10f, -10f);
vertices[10].Color = Color.Red;
vertices[11].Position = new Vector3(10f, -10f, -10f);
vertices[11].Color = Color.Yellow;
```

Les deux triangles ajoutés correspondent dans l'image à : [Fig.8].
Nous affichons maintenant 4 triangles. Pourtant à l'affichage nous obtenons le même résultat que précédemment, que s'est-il passé ? Ce problème est lié à la position de la caméra qui empêche de voir la nouvelle face créée. Modifiez sa position ainsi :

```
this._effect.View = Matrix.CreateLookAt(new Vector3(20, 30,
50), Vector3.Zero, Vector3.Up);
```

Scène de crime et indices

36 vertices pour faire un cube (voir code source)… c'est énorme. Encore plus si on regarde le tableau de vertices sous cet angle : [Fig.10]. On remarque rapidement de nombreuses redondances et que seules 8 positions sont nécessaires pour créer un cube. Mais la contrainte du Triangle List nous oblige à utiliser un vertex pour chaque triangle qui l'utilise. L'image suivante se passe de commentaires : [Fig.11].
Le vertex en surbrillance rouge, qui correspond au vertices 8, 10, 13 et 35 appartient à 3 faces dans ce cube. Il est ainsi relié à un triangle de la face supérieure (rose), à deux triangles de la face frontale (jaune) et un triangle de la face droite (bleue).

Il sera donc à ce stade de nos connaissances répliqué 4 fois ! C'est autant de place de perdue dans la mémoire de la carte graphique lorsque nous appelons la méthode DrawUserPrimitive en lui passant un tableau aussi inutilement gros.

Les indices permettent de spécifier la réutilisabilité d'un point. Le système est assez comparable au jeu «reliez les points» : [Fig.12].
Appliquons ce système au cube. Le tableau de vertices n'aura maintenant que 8 cases correspondant au 8 vertices uniques que nous avons déterminés plus haut :

Fig.5

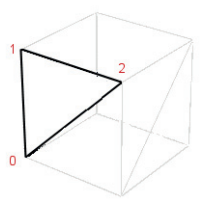


Fig.7

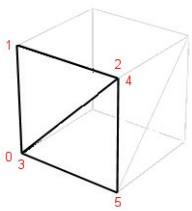


Fig.6

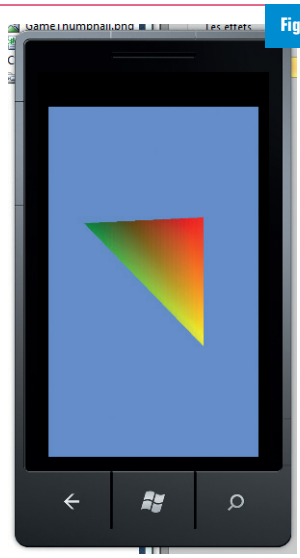


Fig.8

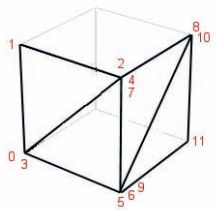
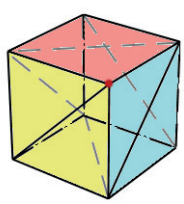


Fig.11



```
vertices[0].Position = new Vector3(-10f, -10f, 10f);
vertices[1].Position = new Vector3(-10f, 10f, 10f);
vertices[2].Position = new Vector3(10f, 10f, 10f);
vertices[3].Position = new Vector3(10f, -10f, 10f);
vertices[4].Position = new Vector3(10f, 10f, -10f);
vertices[5].Position = new Vector3(10f, -10f, -10f);
vertices[ 6 ].Position = new Vector3(-10f, -10f, -10f);
vertices[7].Position = new Vector3(-10f, 10f, -10f);
```

En image cela nous donne : [Fig.13].
Maintenant nous devons relier ces points. Non pas comme le père Noël de la [Fig.12], en reliant tous les points par ordre croissant, mais en définissant des triplets correspondant à des triangles. Avant de passer à la pratique faisons un peu de théorie. Le premier triplet sera (0,1,2), c'est-à-dire le premier vertex, le second, et le troisième (nous les donnons dans le sens des aiguilles d'une montre car la carte graphique par défaut n'affiche pas les triangles dont les vertices sont lus dans le sens inverse). Le premier triangle est relié : [Fig.14]
Passons au second triangle le triplet sera : (0,2,3). Nous donnons ici deux indices (0 et 2) qui ont déjà été utilisés avant. Un bel exemple du gain en termes de nombre de vertices [Fig.15].
Avec seulement 8 vertices nous créons une forme à 12 primitives (12 triangles) ! La mémoire de la carte graphique vous remerciera par de meilleures performances. Reprenez le code du projet précédent et commencez par définir deux nouvelles variables :

```
VertexBuffer vertexBuffer;
IndexBuffer indexBuffer;
```

Ces deux objets représentent des buffers. Ils sont destinés à recueillir respectivement des vertices et des indices. Ces buffers sont plus efficaces qu'un simple tableau pour gérer des ensembles de données. La carte vidéo sait très bien les appréhender et place leur contenu dans sa propre mémoire afin d'y accéder rapidement. Nous devrons donc remplir vertexBuffer avec les vertices de notre cube et remplir indexBuffer avec les indices vers les vertices dans un ordre approprié permettant de lister les primitives de ce cube. Nous allons créer une méthode pour charger les buffer de vertices et d'indices. Elle se présente ainsi :

```
private void InitializeVerticesAndIndices()
{
    vertices = new VertexPositionColor[ 8 ];
    vertices[0].Position = new Vector3(-10f, -10f, 10f);
    vertices[0].Color = Color.Yellow;
    vertices[1].Position = new Vector3(-10f, 10f, 10f);
    vertices[1].Color = Color.Green;
    vertices[2].Position = new Vector3(10f, 10f, 10f);
    vertices[2].Color = Color.Blue;
    vertices[3].Position = new Vector3(10f, -10f, 10f);
    vertices[3].Color = Color.Black;
    vertices[4].Position = new Vector3(10f, 10f, -10f);
    vertices[4].Color = Color.Red;
    vertices[5].Position = new Vector3(10f, -10f, -10f);
    vertices[5].Color = Color.Violet;
    vertices[ 6 ].Position = new Vector3(-10f, -10f, -10f);
    vertices[ 6 ].Color = Color.Orange;
    vertices[7].Position = new Vector3(-10f, 10f, -10f);
    vertices[7].Color = Color.Gray;

    short[] indices = new short[36]{
    0,1,2, //face devant
    0,2,3,
```

Fig.10

```
vertices[0].Position = new Vector3(-10f, -10f, 10f);
vertices[1].Position = new Vector3(-10f, 10f, 10f);
vertices[2].Position = new Vector3(10f, 10f, 10f);
vertices[3].Position = new Vector3(-10f, -10f, 10f);
vertices[4].Position = new Vector3(10f, 10f, 10f);
vertices[5].Position = new Vector3(10f, -10f, 10f);
vertices[ 6 ].Position = new Vector3(10f, -10f, 10f);
vertices[7].Position = new Vector3(10f, 10f, 10f);
vertices[ 8 ].Position = new Vector3(10f, 10f, -10f);
vertices[9].Position = new Vector3(10f, -10f, 10f);
vertices[10].Position = new Vector3(10f, 10f, -10f);
vertices[11].Position = new Vector3(10f, -10f, -10f);
vertices[12].Position = new Vector3(10f, -10f, -10f);
vertices[13].Position = new Vector3(10f, 10f, -10f);
vertices[14].Position = new Vector3(-10f, 10f, -10f);
vertices[15].Position = new Vector3(10f, -10f, -10f);
vertices[16].Position = new Vector3(-10f, 10f, -10f);
vertices[17].Position = new Vector3(-10f, -10f, -10f);
vertices[18].Position = new Vector3(-10f, -10f, -10f);
vertices[19].Position = new Vector3(-10f, 10f, -10f);
vertices[20].Position = new Vector3(-10f, 10f, 10f);
vertices[21].Position = new Vector3(-10f, -10f, -10f);
vertices[22].Position = new Vector3(-10f, 10f, 10f);
vertices[23].Position = new Vector3(-10f, -10f, 10f);
vertices[24].Position = new Vector3(-10f, 10f, 10f);
vertices[25].Position = new Vector3(-10f, 10f, -10f);
vertices[26].Position = new Vector3(10f, 10f, -10f);
vertices[27].Position = new Vector3(-10f, 10f, 10f);
vertices[28].Position = new Vector3(10f, 10f, -10f);
vertices[29].Position = new Vector3(10f, 10f, 10f);
vertices[30].Position = new Vector3(-10f, -10f, -10f);
vertices[31].Position = new Vector3(-10f, -10f, 10f);
vertices[32].Position = new Vector3(10f, -10f, 10f);
vertices[33].Position = new Vector3(-10f, -10f, -10f);
vertices[34].Position = new Vector3(10f, -10f, 10f);
vertices[35].Position = new Vector3(10f, -10f, -10f);
```

Fig.12

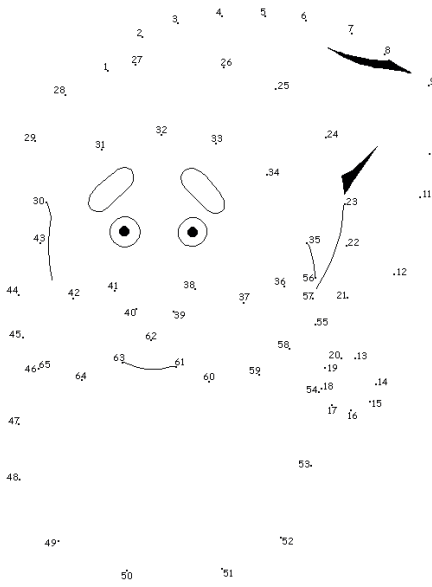


Fig.13

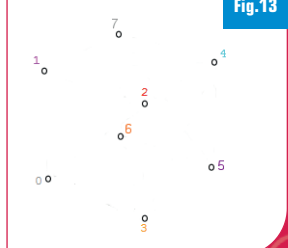


Fig.14

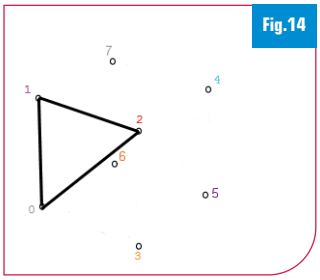
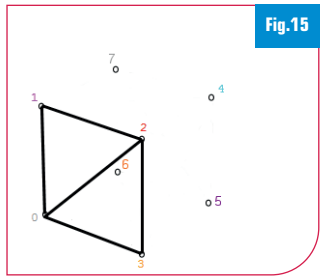


Fig.15



```
    3,2,4, //face droite
    3,4,5,
    5,4,7, //face arrière
    5,7,6,
    6,7,1, //face gauche
    6,1,0,
    6,0,3, //face bas
     6,3,5,
    1,7,4, //face haut
     1,4,2};
   this.vertexBuffer = new VertexBuffer(this.graphics.GraphicsDevice, typeof(VertexPositionColor), 8, BufferUsage.WriteOnly);
    this.vertexBuffer.SetData(vertices);

   this.indexBuffer = new IndexBuffer(this.graphics.GraphicsDevice, typeof(short), 36, BufferUsage.WriteOnly);
    this.indexBuffer.SetData(indices);
}
```

Les buffers sont créés en deux instructions : on les crée puis on les remplit. On passe généralement au constructeur, le device (carte graphique) où stocker les données, le type de données à stocker, la taille du tableau et son utilisation. L'indexbuffer est chargé avec un tableau de short. On y trouve 12 fois 3 entiers qui correspondent aux triplets formant des triangles.
Ne reste donc plus qu'à modifier l'affichage du cube. Nous devons spécifier au device le vertexbuffer et l'index buffer que nous utilisons et utiliser une autre méthode conçue pour l'affichage de formes 3D à l'aide d'indices. Ajoutez ces deux instructions à la méthode Draw juste après le Clear :

```
this.GraphicsDevice.SetVertexBuffer(this.vertexBuffer);
this.GraphicsDevice.Indices = this.indexBuffer;
```

Modifiez l'appel à DrawUserPrimitiveAinsi :

```
this.graphics.GraphicsDevice.DrawIndexedPrimitives(PrimitiveType.TriangleList, 0, 0, 8, 0, 12);
```

Nous sommes toujours en type de reliaison TriangleList mais en mode indexé (avec indices).

Un premier terrain

A ce stade, nous pouvons générer un premier terrain. Il faut « simplement » créer une grille composée de tiles collés les uns aux autres. Regardez cette image de terrain en 3D : [Fig.16].
A première vue rien ne laisse imaginer qu'il s'agit d'une grille de tiles. Mais si maintenant nous ajoutons un quadrillage, nous avons : [Fig.17]. Qu'est-ce qu'une tile ? Tout simplement une face carrée. Et notre cube en contient 6 ! Nous allons donc créer un ensemble de carrés comme pour notre cube mais collés les uns aux autres. Ajoutons à la classe Game1 trois valeurs indiquant la taille d'une tile, le nombre de tiles en largeur, et en hauteur :

```
int tileSize = 2;
int terrainTilesWidth = 100;
int terrainTilesHeight = 100;
```

Modifions ensuite la création des vertices et indices ainsi :

```
private void InitializeVerticesAndIndices()
{
    List<VertexPositionColor> vertices = new List<VertexPositionColor>();

    for (int j = 0; j < terrainTilesHeight; j++)
    {
        for (int i = 0; i < terrainTilesWidth; i++)
        {
           vertices.Add(new VertexPositionColor(new Vector3(i * tileSize, 0, j * tileSize), Color.Red));
           vertices.Add(new VertexPositionColor(new Vector3(i * tileSize, 0, j * tileSize + tileSize), Color.Green));
           vertices.Add(new VertexPositionColor(new Vector3(i * tileSize + tileSize, 0, j * tileSize), Color.Blue));
           vertices.Add(new VertexPositionColor(new Vector3(i * tileSize + tileSize, 0, j * tileSize + tileSize), Color.Yellow));
        }
    }
   this.vertexBuffer = new VertexBuffer(this.graphics.GraphicsDevice, typeof(VertexPositionColor), vertices.Count, BufferUsage.WriteOnly);
    this.vertexBuffer.SetData(vertices.ToArray());
```

Pour chaque tile nous créerons 4 vertices correspondant aux quatre sommets du carré. Le carré a en plus pour taille tileSize et nous utilisons cette valeur pour bien placer le carré. En outre nous donnons aux quatre sommets une couleur différente. Passons aux indices :

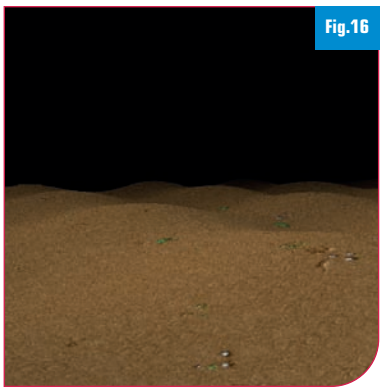


Fig.16

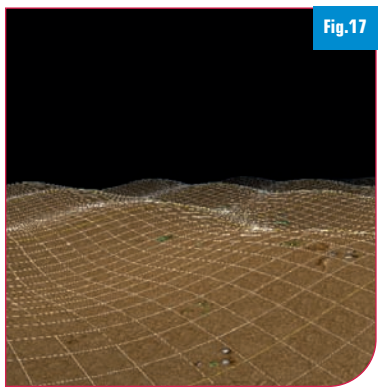


Fig.17

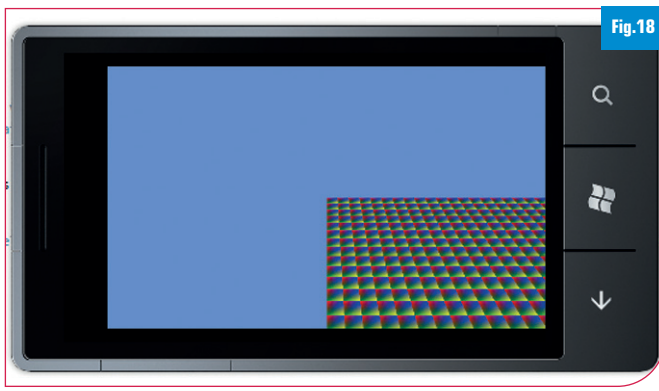


Fig.18

```
    List<short> indices = new List<short>();
    for (int i = 0; i < terrainTilesWidth * terrainTiles
Height; i++)
        {
                short index = (short)(i*4);
                indices.Add( (short)(index+2));
                indices.Add( (short)(index+1));
                indices.Add((short)(index));
                indices.Add( (short)(index+2));
                indices.Add( (short)(index+3));
                indices.Add( (short)(index+1));
        }
   this.indexBuffer = new IndexBuffer(this.graphics.Graphics
Device, typeof(short), indices.Count, BufferUsage.WriteOnly);
    this.indexBuffer.SetData(indices.ToArray());
}
```

Pour chacune des tiles, nous indiquons comment créer les deux triangles à l'aide de 6 reliaisons. Terminons en modifiant l'appel à DrawIndexedPrimitive :

```
this.graphics.GraphicsDevice.DrawIndexedPrimitives(Primitive
Type.TriangleList, 0, 0, terrainTilesHeight * terrainTiles
Width * 4, 0, terrainTilesHeight * terrainTilesWidth*2);
```

et en modifiant la matrice View pour replacer la caméra :

```
this._effect.View = Matrix.CreateLookAt(new Vector3(0, 30,
50), Vector3.Zero, Vector3.Up);
```

Si vous lancez le jeu vous obtenez un joli terrain : [Fig.18]. Encore faut-il appliquer une texture… Une coordonnée de texture permet de spécifier une position dans la texture. Dans la mesure où, dans une texture, l'espace est en 2D, seules deux valeurs sont nécessaires pour spécifier une position : U et V. U représente l'abscisse, à savoir le nombre d'unités à parcourir horizontalement et V qui représente le nombre d'unités verticales (ordonnées) à parcourir. Leurs valeurs oscillent entre 0 et 1 (des valeurs supérieure peuvent être données pour répéter une texture, nous verrons cela plus tard). Le point supérieur gauche de la texture est donc le (0,0) et le point inférieur droit (1,1) pour (U, V). Le schéma ci-dessous explicite tout cela avec la texture que nous allons employer dans notre premier exemple : [Fig.19]. La structure VertexColorPosition que nous utilisons jusqu'ici n'est plus bonne, elle ne contient pas d'information sur le texturing. Changeons-là pour VertexPositionTexture. Dans la création des vertices nous allons remplacer dans le constructeur la couleur associée à chaque vertex par une coordonnée de texture comme l'indique l'image ci-dessus :

Fig.19

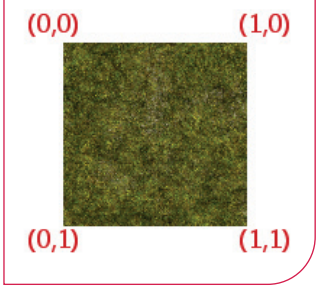


Fig.20

```
vertices.Add(new VertexPositionTexture(new Vector3(i * tile
Size, 0, j * tileSize), new Vector2(0,0)));
vertices.Add(new VertexPositionTexture(new Vector3(i * tile
Size, 0, j * tileSize + tileSize), new Vector2(0, 1)));
vertices.Add(new VertexPositionTexture(new Vector3(i * tile
Size + tileSize, 0, j * tileSize), new Vector2(1, 0)));
vertices.Add(new VertexPositionTexture(new Vector3(i * tile
Size + tileSize, 0, j * tileSize + tileSize), new Vector2(1,
0)));
```

Ajoutez l'image suivante : [Fig.20].
(ground.png Fournie avec le code source) au projet de contenu. Le Xna Game Studio(XGS) va lui associer un identifiant nous permettant de l'utiliser dans notre code source et va évidemment la compiler. Ajoutez le handle suivant à la classe Game1 :

```
Texture2D _groundTexture;
```

Associez ce handle à un objet en chargeant la texture dans la méthode Load ainsi :

```
_groundTexture = this.Content.Load<Texture2D>(«ground»);
```

("ground" étant l'identifiant de la texture généré par le XGS)). Dans Draw, supprimez l'instruction autorisant les vertex colorés :

```
this._effect.VertexColorEnabled = true;
```

Remplacez-là par une autorisant les vertex texturés pour le fichier Effet. Indiquez de même la texture que celui-ci doit utiliser :

```
            this._effect.Texture = this._groundTexture;
            this._effect.TextureEnabled = true;
```

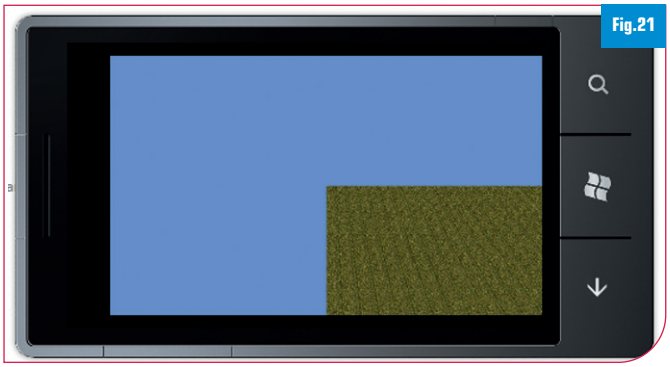
Lancez le jeu, vous obtenez : [Fig.21]. Ajoutons un peu de relief. Pour l'heure la méthode de génération de vertices donne pour l'altitude Y de chaque vertex la valeur 0, notre terrain est donc un large plan plat. Ajoutez cette méthode à Game1:

```
internal float HeightField(float x, float z)
{
    return (float)(5 * Math.Cos(x / 10f) + 2 * Math.Sin(z / 8f));
}
```

Basée sur un calcul trigonométrique elle renvoie une hauteur selon la position qu'on lui passe. Modifiez la création des vertices ainsi :

```
vertices.Add(new VertexPositionTexture(new Vector3(i * tile
Size, this.HeightField(i * tileSize, j * tileSize), j * tile
Size), new Vector2(0,0)));
```

Fig.21



```
vertices.Add(new VertexPositionTexture(new Vector3(i * tile
Size, this.HeightField(i * tileSize, j * tileSize + tile
Size), j * tileSize + tileSize), new Vector2(0, 1)));
vertices.Add(new VertexPositionTexture(new Vector3(i * tile
Size + tileSize, this.HeightField(i * tileSize + tileSize, j
 * tileSize), j * tileSize), new Vector2(1, 0)));
vertices.Add(new VertexPositionTexture(new Vector3(i * tile
Size + tileSize, this.HeightField(i * tileSize + tileSize, j
 * tileSize + tileSize), j * tileSize + tileSize), new
 Vector2(1, 0)));
```

Terminons en ajoutant du réalisme à la scène en utilisant des normales. Une normale est une information qu'on associe à un vertex. Elle permet à la carte graphique de réaliser des calculs liés à l'intensité lumineuse à appliquer à un point donné d'une forme 3D. Une normale doit être perpendiculaire au triangle auquel appartient le vertex. Plus l'angle entre la direction prise par la source lumineuse et la normale est importante, plus faible sera l'intensité lumineuse. Regardez cette boule sommaire, la source lumineuse vient du haut : [Fig.22]. Le calcul d'une normale est très simple et doit être appliqué à tous les vertices du terrain. On calcule la normale d'un triangle de cette manière :

```
    Vector3 side1 = vertice1.Position – vertex2.Position;
    Vector3 side2 = vertice1.Position – vertice3.Position;
    Vector3 normal = Vector3.Cross(side1, side2);
```

Changez la structure VertexPositionTexture par VertexPositionNormalTexture qui ajoute aux vertices une information de normale.

Fig.22

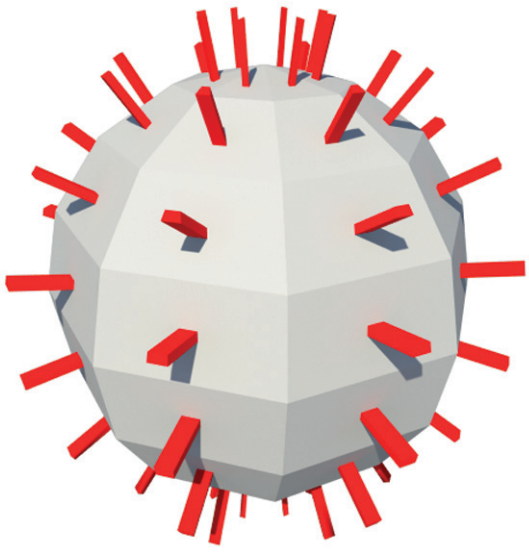
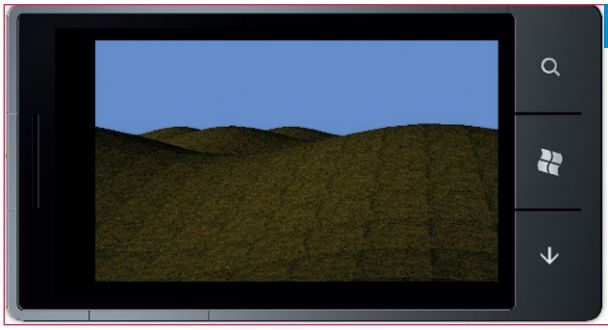


Fig.24



Modifiez la création de vertices (voir code complet sur le site). Nous donnons ici Vector3.Zero car nous ne sommes pas capables à ce stade de calculer une normale. Après la création des indices, parcourez à nouveau les vertices mais dans l'ordre des indices et appliquez la formule que nous avons donnée précédemment :

```
for (int indice = 0; indice < indices.Count; indice += 3)
{
    Vector3 side1 = vertices[indices[indice]].Position - vertices
[indices[indice+1]].Position;
    Vector3 side2 = vertices[indices[indice]].Position - vertices
[indices[indice + 2]].Position;
     Vector3 normal = Vector3.Cross(side1, side2);
     vertices[indices[indice ]].Normal = vertices[indices
[indice + 1]].Normal = vertices[indices[indice + 2]].Normal
= normal;
}
```

Nous parcourons chaque triplet d'indices pour récupérer les trois vertices de chaque triangle et nous calculons la normale que nous réappliquons au vertex concerné.
Modifiez la méthode Draw pour rajouter l'instruction suivante :

```
this._effect.EnableDefaultLighting();
```

Cette instruction génère une source lumineuse automatiquement, la carte graphique va calculer l'intensité lumineuse en chaque point du terrain grâce à nos normales.

Une caméra pour les dominer tous

Pour terminer cet article passionnant nous allons utiliser un composant permettant à l'utilisateur de se déplacer dans un monde à l'aide de son doigt. Utiliser un composant est simple. On le crée, on l'enregistre dans la classe Game et on l'exploite en cours de jeu. Un composant expose les méthodes clés du Game flow d'un jeu : Initialize, Update, et parfois Draw. La classe Game s'occupe d'appeler ces méthodes à votre place. Ajoutez la classe Camera.cs à votre projet. Puis créez un handle :

```
private Camera _camera;
```

Instanciez la classe dans la méthode Initialize et ajoutez-la à la liste des composants gérés :

```
            this. camera = new Camera(this, new Vector3(0,30,
50),  Vector3.Up);
             this.Components.Add(_camera);
```

Désormais, la caméra sera automatiquement gérée par le jeu. Vous n'avez plus qu'à récupérer les membres utiles qu'elle expose comme View et Projection dans la méthode Draw : [Fig.24].

```
this._effect.View = _camera.View;
this._effect.Projection = _camera.Projection;
```

La classe Camera se contente juste d'étudier les actions réalisées avec un doigt sur le touch screen du Windows Phone et déplace la camera en fonction. Maintenant à vous de créer et de jouer… en 3D !

■ Valentin **Billotte** - *CEO – Graphic Stream*
valentin@graphicstream.fr
http://www.graphicstream.fr

Graphic Stream

Migrer vers Windows Phone 7

Nous allons nous intéresser à un aspect important du développement Windows Phone 7. Vous êtes développeur Windows Phone 6.5, Android ou iPhone, comment faire pour migrer efficacement sur Windows Phone 7 ? Toutes les réponses dans les pages qui suivent !

Migration d'une application iOS vers Windows Phone 7

Ce document n'est pas un tutoriel qui explique comment faire telle ou telle tâche sous Windows Phone 7 pour un développeur iOS. C'est plutôt une liste de références, de vocabulaire et de concepts permettant à un développeur iOS de s'y retrouver pendant le développement ou le portage d'une application iOS vers Windows Phone 7.

Les outils de développement et les différentes méthodologies

Voici quelques correspondances au niveau des outils de développement :

iOS	WP7
Outil de développement **xCode**	**Visual Studio 2010**
Langage **Objective C**	Langage **C#**
Designer d'interfaces **Interface Builder**	**Designer Silverlight** inclus dans VS2010 ou **Expression Blend** pour plus de puissance

Et côté méthodologies :

iOS	WP7
Le pattern **MVC** (Modèle-Vue-Contrôleur)	Le modèle **MVVM** (Modèle-Vue-VueModèle) est composé des éléments suivants : • Le Modèle (couche métier) • La Vue (XAML + code behind C#) • La VueModèle (object ViewModel) L'objet **ViewModel** fait le lien entre les vues et le modèle de données.

Le vocabulaire

iOS	WP7
Framework à utiliser dans un projet	Référence à inclure au projet
Fichier **NIB** (interface builder)	Fichier **XAML**
Fichiers .h et .m (Objective C)	Fichier **.CS** (C#)
Modèle **MVC**	Modèle **MVVM**
Window (**UIWindow**)	**Frame**
View (**UIView**)	**Page**
Contrôleur de vue (**UIViewController**)	**Code behind** et **ViewModel**

Les contrôles

Les contrôles sont au centre du développement de l'interface graphique. Voici un tableau qui fait une correspondance entre les principaux contrôles iOS et ceux de WP7.

iOS	WP7
UITableView	ListBox
UITableViewCell	Voir la notion de ListBox.ItemTemplate
UIImageView	Image
UIWebView	WebBrowser
MKMapView	Voir développement carto avec Bing
UITextView	TextBlock
UIScrollView	ScrollViewer StackPanel
UIPickerView	Absent
UIDatePicker	Absent
UISegmentedControl	Absent
UILabel	Label
UIButton	Button
UITextField	TextBox PasswordBox
UISwitch	CheckBox ou RadioButton
UISlider	Slider
UIProgressView	ProgressBar
UIActivityIndicatorView	Voir la propriété Cursor de PhoneApplicationPage
UIPageControl	Voir la notion de control Pivot
UIWindows	PhoneApplicationFrame
UIView	PhoneApplicationPage
UISearchBar	Absent
UINavigationBar UINavigationItem	Voir le système de navigation entre pages
UIToolBar	ApplicationBar
UIBarButtonItem	ApplicationBarIconButton ApplicationBarMenuItem
UITabBar UITabBarItem	Voir la notion de contrôle Pivot
Lecture de vidéos et de fichiers audio	MediaElement

Correspondances entre les événements de vie de l'application

Sous WP7, dès qu'on sort d'une application, elle est fermée et c'est à vous de sauvegarder les données. Ce mécanisme ressemble beaucoup à celui adopté par iOS. Dans les deux plateformes, l'application est notifiée des changements d'état par des événements. Dans iOS c'est la classe **UIApplication** qui génère ces événements, dans WP7 c'est la classe **Application.**

Evénements communs	iOS	WP7
Lancement	applicationDidFinishLaunching	Launching
Désactivation	applicationDidEnterBackground	Deactivated
Réactivation	applicationWillEnterForeground	Activated
Fermeture	applicationWillTerminate	Closing

Les principales fonctionnalités

Localisation par Cell ID, Wifi et GPS

iOS	WP7
Utiliser le framework **CoreLocation**	Ajouter une référence à **System.Device**
Utiliser la classe **CLLocationManager**	Utiliser la classe **System.Device.Location.GeoCoordinateWatcher**
Répondre au protocole **CLLocationManagerDelegate**	Répondre aux événements **PositionChanged et StatusChanged**

Accéléromètre

iOS	WP7
Utiliser le framework **CoreMotion**	Ajouter une référence à **Microsoft.Device.Sensors**
Utiliser la classe **CMMotionManager**	Utiliser la classe **Microsoft.Device.Sensors.Accelerometer**
Utiliser **NSOperationQueue** pour récupérer les informations.	Répondre à l'événement **ReadingChanged**

Lancer un appel téléphonique

iOS	WP7
Ouvrir une URL formatée (tel : numéro) avec la méthode **openURL** de l'objet [**UIApplication sharedApplication**]	Ajouter une référence à Microsoft.Phone Utiliser la classe **Microsoft.Phone.Task.PhoneCallTask**

Prendre une photo

iOS	WP7
Utiliser **UIImagePickerController**	Ajouter une référence à **Microsoft.Phone** Utiliser la classe **Microsoft.Phone.Tasks.CameraCaptureTask**

Sélectionner des informations dans le carnet d'adresses

iOS	WP7
Utiliser le framework AddressBookUI	Ajouter une référence à **Microsoft.Phone**
Utiliser **ABPeoplePickerNavigationController** pour sélectionner un contact dans le carnet d'adresses.	Utiliser la classe **Microsoft.Phone.Tasks.PhoneNumberChooserTask** pour récupérer le numéro de téléphone d'un contact. Utiliser la classe **Microsoft.Phone.Task.EmailAddressChooserTask** pour récupérer l'adresse mail d'un contact.

Ouvrir une page web externe

iOS	WP7
Ouvrir l'url demandée avec la méthode **openURL** de l'objet [**UIApplication sharedApplication**]	Ajouter une référence à **Microsoft.Phone** Utiliser la classe **Microsoft.Phone.Tasks.WebBrowserTask**

Composer un email

iOS	WP7
Utiliser le framework MessageUI	Ajouter une référence à **Microsoft.Phone**
Utiliser la classe **MFMailComposeViewController**	Utiliser la classe **Microsoft.Phone.Task.EmailComposeTask**

Composer un SMS

iOS	WP7
Utiliser le framework **MessageUI**	Ajouter une référence à **Microsoft.Phone**
Utiliser la classe **MFMessageComposeViewController**	Utiliser la classe **Microsoft.Phone.Task.SmsComposeTask**

Stockage des données

Les applications WP7 fonctionnent dans une Sandbox, comme les applications iOS. Sous WP7 le système de fichier est accessible via l'Isolated Storage. Pour le moment pas de SGBD.
Il faut donc assurer la persistance des données via de simples fichiers (texte, XML).

iOS	WP7
Utilisation des fonctions standard de gestion de fichiers, principalement via la classe **NSFileManager.**	Utilisation de **System.IO.IsolatedStorage.Isolated StorageFile** et **IsolatedStorageStream**

Réseau

Seule la communication http asynchrone avec un serveur est prise en charge sous WP7.

iOS	WP7
Requête http via la classe **NSURLConnection**	Utilisation de la classe **WebRequest** et **WebClient**
Pour la réception d'informations depuis un serveur, les deux systèmes utilisent le **Push Notifications.**	

Analyse et performances

Les deux plateformes proposent des outils d'analyse des applications générées. Chez Apple les outils sont externes, chez Microsoft, ils sont à activer dans l'application.

iOS	WP7
Utilisation principalement de l'application **Instruments**.	Activation de compteurs de performance en plaçant la valeur **true** dans les différentes propriétés de l'objet **Application.Current.Host.Settings** : • **EnabledFrameRateCounter** • **EnabledCacheVisualization** • **EnabledRedrawRegion**

Le développement ou la migration d'une application vers Windows Phone 7 pour le développeur iPhone ne doit pas être vu comme une tâche d'une complexité extrême. Même si le vocabulaire, la plateforme et les outils changent, les concepts de base sont les mêmes et les choix techniques se rejoignent.

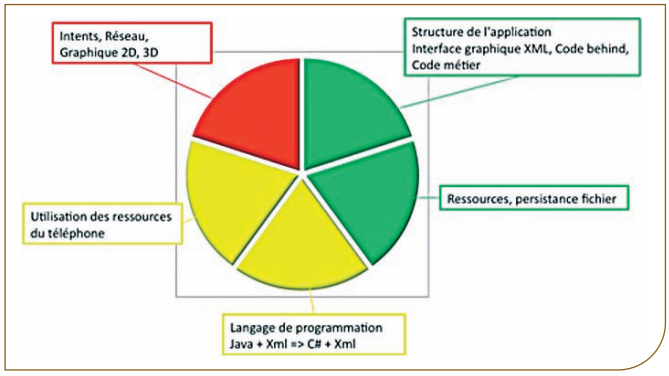
Bien sûr il faudra tout reprendre de zéro (interface et code), seules certaines ressources (images, sons, fichiers xml) seront réutilisables, mais les outils de design sont destinés à nous faire gagner un temps énorme, alors il ne faut pas s'en priver. Côté code, il est important de coller au maximum aux méthodologies qui ont fait leurs preuves (MVC chez Apple et MVVM chez Microsoft).
Objective C et C# sont tous les deux des enfants du C, alors il reste simple de passer de l'un à l'autre.
Le grand plus chez Windows Phone 7 est sa gestion de la mémoire qui permet d'obtenir une application extrêmement saine à ce niveau et d'ailleurs, vivement qu'elle soit également disponible pour le développement iOS.

■ Stéphane **Sibué**
Directeur du développement chez Softélite
(stephane.sibue@softelite.fr)
Microsoft MVP Device Application Development depuis 2003
Webmaster et Fondateur de CodePPC
(www.codeppc.com)

Migration d'une application Android vers Windows Phone 7

Android et WP7 sont très similaires. Ce petit guide vous donnera une appréciation de l'effort de portage et les concepts transverses entre ces 2 technologies.

Effort de migration



Processus de migration

Phase 0 - Familiarisation de la plateforme WP7 Silverlight

Reprenez les exemples fournis et effectuez les tutoriels pour vous familiariser avec la plateforme sur des projets de type Silverlight.

1 Création du projet WP7 par import des ressources du projet Android

- Construire un projet de zéro
- Insérer les ressources graphiques, vidéos, sons, brutes de votre application
- Migrer les ressources textes
- Importer les UI Xml et les migrer pour exploiter les ressources textes WP7
- Migrer le fichier AndroidManifest.xml
- Importer et migrer le code behind des interfaces en commentant le code pour atteindre le stade de compilation

2 Migration de l'UI

Ecran par écran :

- Migrer le code relatif aux activities vers les services et tasks de WP7
- Migrer la navigation vers les autres écrans
- Chiffrer l'effort de migration pour estimer le temps total

3 Migration fonctionnelle

Pour chaque classe business

- Migrer le code fonctionnel de Java vers le C#
- Chiffrer l'effort de migration pour estimer le temps total

4 Optimisation

Tester, repérer et optimiser les fonctionnalités qui pourraient remettre en question l'utilisation de votre logiciel par l'utilisateur (latences, ergonomie, ...)

Conseil : Multithreader votre application uniquement en fin de développement et seulement si c'est nécessaire.

Note : Les séquences de code Android que vous avez dû optimiser doivent être les mêmes que celles en WP7.

Structure de l'application / UI Xml et code behind

Il existe un réel parallèle entre la structure d'une application Android et une application WP7. Comme sous Android, vous retrouverez un fichier manifest, des vues XML avec du code behind, des contrôles avec leurs événements, des pages avec leur cycle de vie ainsi que des ressources de tous types.

Le fichier Manifest

Android	WP7
AndroidManifest.xml	*Manifest.xml*
<manifest xmlns="...">	<Deployement xmlns="...">
<application>	<App>
n/a	<Tasks>
<activity>	<DefaultTask>, <Task>
n/a	<Capabilities>
<uses-permission>	<Capability>

Les contrôles

android.View.;*	*System.Windows.Controls.**
Canvas	Canvas
LinearLayout	StackPanel
TableLayout, GridView	Grid
Button, ImageButton	Button
EditText	TextBox
ImageView	Image
ListView	ListBox
ProgressBar	ProgressBar
RadioButton	RadioButton
ScrollView	ScrollViewer
TextBlock	TextBlock

Les listeners de contrôles

android.view.View - Les listeners	*System.Windows.UIElement*
OnClickListener	MouseDown
OnLongClickListener	n/a
OnKeyListener	KeyUp

Le cycle de vie d'une Activity/Page

android.app.Activity	*Microsoft.Phone.Controls.PhoneApplicationPage*
OnCreate	Launching
OnStart, OnResume, OnRestart	Activated
OnPause, OnStop	Deactivated
OnDestroy, OnFreeze	Closing

Accès aux ressources XML

<TextView android:id="@+id/myTxt" android:text="@string/submit" />	<Application.Resources> <local:LocalizedStrings xmlns:local ="clr-namespace:Global2" x:Key="LocalizedStrings"> </Application.Resources> <TextBlock x:Name="myTxt" Text="{Binding Path= Localizedresources.submit, Source={StaticResource LocalizedStrings}}"/>

Accès aux ressources en code behind

TextView txtView = (ImageView) findViewById(R.id.myTxt); imageView.setText(R.string.submit);	ResourceManager rm = new ResourceManager("SilverlightApplication.LocalizedStrings", Assembly.GetExecutingAssembly()); myTxt.Text = rm.GetString("submit");

Persistance Fichiers / Paramètres de l'application

La persistance de données s'effectue pour le moment dans des **fichiers** accessibles uniquement par l'application et par des **préférences** telles que sous Android. Il n'est pas possible d'utiliser une base de données locale pour l'instant.

Récupérer le contexte du système de fichier de l'application

Android	WP7
n/a	IsolatedStorageFile myStore = IsolatedStorageFile.GetUserStoreForApplication();

Créer un dossier

getDir("ImageFolder");	myStore.CreateDirectory("ImageFolder") ;

Ecrire un fichier

OutputStreamWriter osw = new OutputStreamWriter(openFileOutput("ImageFolder\myFile.txt", Context.MODE_PRIVATE)); osw.write("myText"); osw.flush(); osw.close();	StreamWriter writeFile = newStreamWriter(new IsolatedStorageFileStream("ImageFolder\\myFile.txt", FileMode.OpenOrCreate, myStore)); writeFile.WriteLine("myText"); writeFile.Close();

Lire un fichier

InputStreamReader isr = new InputStreamReader(openFileInput("samplefile.txt")); char[] inputBuffer = new char[lengthOfFile]; isr.read(inputBuffer); String readString = new String(inputBuffer);	StreamReader readFile = new StreamReader(newIsolatedStorageFileStream("ImageFolder\\myFile.txt", FileMode.Open, myStore)); string fileText = readFile.ReadLine();

Récupérer les préférences d'une application

SharedPreferences settings = getSharedPreferences("MyPrefFile", 0);	IsolatedStorageSettings isolatedStore = IsolatedStorageSettings.ApplicationSettings ;

Lire une préférence

String sVal = settings.getString("stringValue", "");	value = (cast)isolatedStore[Key];

Créer un éditeur de préférence

SharedPreferences.Editor editor = settings.edit();	n/a

Ajouter une préférence

editor.putString("stringValue", mMyVal);	isolatedStore.Add(Key, value);

Mettre à jour une préférence

editor.putString("stringValue", mMyVal);	isolatedStore[Key] = value;

Sauver les préférences

editor.commit();	isolatedStore.Save();

Les fonctionnalités du téléphone

Récupérer l'orientation du téléphone

Android	WP7
android.hardware.SensorManager android.hardware.SensorEventListener	*Microsoft.Devices.Sensors.Accelerometer*

Récupérer une position GPS

android.Location.LocationManager; android.Location.Location;	*System.Device.Location.GeoCoordinateWatcher*

Appeler un contact

startActivity(new Intent(ACTION_DIAL, Uri.parse("tel:55555555555")));	PhoneCallTask phoneCallTask = new PhoneCallTask(); phoneCallTask.PhoneNumber = "55555555555"; phoneCallTask.DisplayName = "Gage"; phoneCallTask.Show();

Envoyer un SMS

SmsManager sm = SmsManager.getDefault(); sm.sendTextMessage("5555555555", null, "Check out this new app.", null, null);	SmsComposeTask smsComposeTask = new SmsComposeTask(); smsComposeTask.To = "5555555555"; smsComposeTask.Body = "Check out this new app."; smsComposeTask.Show();

MultiMedia

Les fonctionnalités multimédia et les **animations** sont très similaires entre Android et WP7. L'outil de **design d'interfaces** et d'animations **MS Blend** vous permettra d'optimiser le rendu graphique de votre application et d'améliorer considérablement l'**expérience utilisateur**.

Prendre une photo

Android	WP7
Beaucoup de code …	CameraCaptureTask cameraCaptureTask = new CameraCaptureTask(); cameraCaptureTask.Show();

Ouvrir un navigateur

Intent i = new Intent(Intent.ACTION_VIEW); i.setData(Uri.parse("http://msdn.microsoft.com")); startActivity(i);	WebBrowserTask webBrowserTask = new WebBrowserTask(); webBrowserTask.URL = "http://msdn.microsoft.com"; webBrowserTask.Show();

Les animations

android.view.Animation	System.Windows.Media.Animation
AlphaAnimation, RotateAnimation, ScaleAnimation, TranslateAnimation	DoubleAnimation
AnimationSet	Storyboard

Note : les animations et les scénarios sont généralement créés par un designer dans l'outil Blend.

Jouer un son

MediaPlayer mMediaPlayer = MediaPlayer.create(this, R.raw.my_sound); mMediaPlayer.start();	string streamName = "MyProject.tada.wav"; using (System.IO.Stream stream = this.GetType().Assembly.GetManifestResourceStream(streamName)){ SoundEffect se = SoundEffect.FromStream(stream); FrameworkDispatcher.Update(); se.Play(); }

Jouer une vidéo

MediaPlayer mMediaPlayer = new MediaPlayer(); mMediaPlayer.setDataSource(path); mMediaPlayer.setDisplay(holder); mMediaPlayer.prepare(); mMediaPlayer.setOnPreparedListener(this);	MediaPlayerLauncher mediaPlayerLauncher = new MediaPlayerLauncher(); mediaPlayerLauncher.Media = new Uri("MyVideo.wmv", UriKind.Relative); mediaPlayerLauncher.Show();

```
public void onPrepared(MediaPlayer mediaplayer) {
        startVideoPlayback();
}
```

■ Sébastien **Yriarte** - *Consultant Freelance - Tech & Smile*

Migration d'une application Windows Mobile 6.5 vers Windows Phone 7

La migration d'une application existante est une question que tout développeur ayant fait le pari de Windows Mobile dans ses versions précédentes doit se poser avec Windows Phone 7. En effet, même si auparavant d'une version sur l'autre, les changements étaient mineurs et la compatibilité globalement assurée, le changement radical opéré par Windows Phone 7 oblige à étudier de nouveaux runtimes (Silverlight et XNA vs .NET Compact Framework 2.0 et 3.5) et de nouveaux outils (Visual Studio 2010 et Expression Blend 4 vs Visual Studio 2008). Loin de se vouloir exhaustive, cette fiche a pour but d'aider un développeur Windows Mobile 6.5 à évaluer « facilement » la tâche de portage de son application.

Modèle de l'application

Le modèle actuel de votre application va jouer un rôle primordial dans le portage. Si le découpage interface/ métier de votre application Windows Mobile 6.5 était bien pensé, le portage sera d'autant plus aisé. Les couches IHM seront complètement à récrire, mais les couches logique métiers et modèles de données pourront être réutilisée telles quelles ou portées à quelques deltas près.

Windows Mobile 6.5	Windows Phone 7
Le pattern MVC pouvait être utilisé afin de découpler la couche IHM de la couche métier. Ainsi l'application était composée d'un contrôleur régissant les échanges entre la vue et la couche métier. **Si aucun design pattern n'a été suivi, la séparation est plus compliquée, mais c'est aussi l'occasion de restructurer tout ce code** !	Utilisation de Silverlight, le pattern MVVM est plus adapté pour utiliser le data binding et tirer la quintessence de Silverlight. Dans le pattern MVVM, nous avons toujours une couche métier, le « Model », une vue « View » (composée de l'UI en XAML et d'un fichier code-behind contenant la logique de présentation liée à l'UI) ainsi qu'un objet ViewModel associé à cette vue. Le ViewModel fait le lien entre les vues et le modèle de données : son rôle principal est de « présenter » les données du modèle en propriétés (au sens C#) exploitables par l'UI. Les classes du ViewModel devront implémenter l'interface INotifyPropertyChanged qui permet à l'UI Silverlight d'être rafraîchie lorsqu'une donnée du ViewModel est modifiée.

L'expérience utilisateur

Comme vu dans le paragraphe précédent, l'interface est donc la partie à refondre complètement, le cas échéant, l'expérience utilisateur pourra être revue et adaptée à la nouvelle expérience Windows Phone 7.

Les contrôles

La plupart des contrôles de Windows Mobile 6.5 sont disponibles sur Phone 7 avec des propriétés et des possibilités de customisation beaucoup plus évoluées. Le positionnement des contrôles dynamiques sera grandement facilité grâce à des panels évolués comme le stackpanel ou la grid.

Windows Mobile 6.5	Windows Phone 7
	PhoneApplicationFrame
Form	PhoneApplicationPage
MainMenu	Application bar
	System tray
Panel	Canvas, StackPanel, Grid
Button	Button
TextBox	TextBox
PictureBox **Les PNG transparents ne sont pas gérés nativement, les API natives d'Alpha Blend doivent être wrappées**	Image Le format PNG est supporté nativement et son support est accéléré par GPU
ListView **Difficilement personnalisable**	ListBox Très facilement personnalisable à l'aide des Templates disponibles dans Silverlight
ProgressBar	ProgressBar
Utilisation d'un panel et des évènements de reconnaissance geste	ScrollViewer
Label	TextBlock
Composant MediaPlayer importé depuis l'ActiveX MediaPlayer via un wrapper d'objet COM	Le nouveau composant Media Element est directement accessible depuis le code Silverlight et décode nativement les formats de vidéos suivants : H263, WMV, MPEG-4
Navigateur WEB **System.Windows.Forms.WebBrowser**	WebBrowserTask
Prendre une photo **Microsoft.WindowsMobile.Forms.Camera CaptureDialog**	CameraCaptureTask
Appeler un contact **Microsoft.WindowsMobile.Telephony.Phone**	PhoneCallTask
Envoyer un SMS **Microsoft.WindowsMobile.PocketOutlook. SmsMessage**	SmsComposeTask

Les API

Sur Phone 7 fini les spécificités de code et cas particuliers mis en place pour chaque terminal, tout est unifié grâce à un hardware homogène et une API unifiée pour accéder aux différents capteurs du téléphone.

Windows Mobile 6.5	Windows Phone 7
Très fastidieux de récupérer les informations liées au téléphone notamment pour les points suivants : **- Accéléromètre** **- GPS** **- Radio FM** **- Gestion des boutons du téléphone** **Car il est nécessaire de faire appel à des API natives bas-niveau liées au constructeur du téléphone et rarement fournies.**	Gestion unifiée des API : Accéléromètre (Microsoft. Devices.Sensors.Accelerometer) GPS (System.Device.Location. GeoCoordinateWatcher) Orientation de l'écran Radio FM Gestion des boutons du téléphone.
Orientation de l'écran **Microsoft.WindowsCE.Forms.SystemSettings. ScreenOrientation**	Gestion des boutons du téléphone. PhoneApplicationPage.SupportedOrientation

La reconnaissance de gestes

Sur Phone 7, l'accès au code natif depuis le C# n'est plus autorisé : finis les wrappers d'API natives, tout est directement utilisable depuis le C# via des classes avec des méthodes et évènements ; c'est typiquement le cas de la reconnaissance des gestes de l'utilisateur sur l'écran.

Windows Mobile 6.5	Windows Phone 7
Import de code natif pour utiliser l'API Gesture qui n'était pas disponible en code managé. (traitement des messages WM_GESTURE)	Chaque contrôle dispose d'évènements de manipulation : ManipulationStarted ManipulationDelta ManipulationCompleted Avec pour chacun des paramètres tels que la vélocité, l'angle…

Stockage des données en mémoire

Sur Windows Phone 7, le système de fichiers du terminal n'est « accessible » qu'à travers l'Isolated Storage de l'application et aucun SGBD n'est disponible pour l'instant. La persistance des données sera donc mise en œuvre à l'aide de fichiers XML stockés dans la SandBox ou via des solutions tierces comme SQLite ou Perst.NET.

Windows Mobile 6.5	Windows Phone 7
Utilisation de SQLCE	Utilisation de fichiers XML stockés dans l'Isolated storage et utilisation de LINQ pour requêtes sur les données. Sinon solutions tierces comme SQLite ou Perst.NET.
Accès au système de fichier	Accès à l'Isolated Storage System.IO.IsolatedStorage.*
Sauvegarde de settings : **Base de registre** **DB** **Fichiers**	Sauvegarde des settings dans le dictionnaire de l'IsolatedStorage prévu à cet effet : IsolatedStorageSettings Ce dictionnaire permet de sauvegarder des types sérialisables et le « statut » de l'application indépendamment de fichiers ou d'une base de données.

Réseau

L'accès à des données d'un serveur se fera uniquement via un canal http et en mode asynchrone. Pour les évènements asynchrones provenant d'un serveur il est possible d'utiliser des Push Notifications (cf. paragraphe suivant et article sur le sujet).

Windows Mobile 6.5	Windows Phone 7
Sockets	Pas d'accès au socket : pas de TCP ou d'UDP
HttpWebRequest synchrone	Les WebRequest sont asynchrones, il est conseillé d'utiliser l'objet WebClient qui permet de réaliser une requête http et d'obtenir le résultat via une callback. Cela peut donc complexifier un peu le code, mais c'est mieux d'un point de vue réactivité de l'interface.
L'application peut subsister en background et à tout moment prendre le focus ou gérer des évènements comme le réveil de l'application via un SMS entrant.	Système de notifications en push Ce système permet de remonter des informations à une application fermée et d'afficher en permanence un statut à l'utilisateur sur l'icône de l'application par exemple

Cycle de vie d'une application

Windows Mobile 6.5	Windows Phone 7
L'application reste en background et n'est jamais fermée sauf demande explicite ou soft reset du téléphone	Dès qu'on sort de l'application, cette dernière est fermée et les données ne sont pas automatiquement sauvegardées. Le mécanisme dit de « tombstoning » permet de sauvegarder l'état de l'application et lors du lancement de l'application le restituer. Quatre évènements seront déclenchés afin de mettre en œuvre les comportements adéquats : Launching / Activated : récupération et chargement des données Deactivated / Closing : Sauvegarde des données dans un dictionnaire prévu à cet effet (PhoneApplicationService)

Outils de debug et de monitoring des performances

Reste un point important à traiter qui impacte directement l'expérience utilisateur, ce sont les performances. Voici, de quoi réaliser quelques benchs afin de vérifier que votre application sera à la hauteur du résultat attendu, notamment en termes de fluidité.

Windows Mobile 6.5	Windows Phone 7
Outils du pack Power Toys **Remote Performance Monitor** **NETCF CLR Profiler** **Remote Logging Configuration Tool** ...	Les compteurs de performance : (Application.Current.Host.Settings.Enable FrameRateCounter) Compositeur frame rate (fréquence de rafraîchissement de l'écran) UI thread frame rate (fréquence du thread UI) Mémoire utilisée pour les textures Nombre de surfaces allouées (Application.Current.Host.Settings.Enable CacheVisualization) (Application.Current.Host.Settings.Enable RedrawRegions)

Bien que cet article ne soit pas exhaustif, vous avez maintenant les grandes lignes et pièges à éviter pour porter vos applications Windows Mobile 6.5 vers la nouvelle plateforme Windows Phone 7.
Les étapes suivantes résument la démarche à suivre :

- Création du projet WP7
- Ebauche d'un squelette d'application
- Identification des vues et vues-modèles associées aux données à afficher (datacontext)
- Mise en place de la logique de navigation avec passage des données entre les différentes pages
- Import des ressources textuelles dans les différentes langues si l'application est localisée
- Import des ressources graphiques en PNG ou JPG
- Réalisation des différentes pages en gardant bien en tête l'ergonomie Metro (si possible réalisation d'un storyboard de l'application au préalable pour visualiser l'intégralité de l'application), ne pas oublier de tirer parti des nouveaux composants disponibles pour le positionnement des contrôles
- Si besoin, réalisation de composants personnalisés pour répondre aux besoins de l'application
- Mise en place du code relatif aux activités vers les services et tâches Windows Phone 7
- Migration des couches métiers
- Identifier les accès aux serveurs distants et « asynchroniser » les traitements pour coller au modèle Windows Phone 7. Afin de réaliser une application fluide, il est de rigueur de ne pas réaliser de traitements lourds ou bloquants dans le thread UI.
- Identifier le stockage de données et mettre en place des classes sérialisables pour stockage en XML dans l'IsolatedStorage.
- Le filtrage et traitement des données via des requêtes LINQ pourra être réutilisé tel quel
- Appel des couches métiers depuis les vues-modèles
- Tests de l'intégration des différentes couches interface et métiers
- Optimisation

Le monitoring des performances pourra être réalisé via le biais des compteurs de performances vus plus haut.

■ Fabien **Decret**
MVP. Travaille depuis plus de 5 ans chez Adeneo Embedded (Lyon, 69), société spécialisée dans l'intégration de systèmes embarqués et le développement applicatif pour des plateformes 32-bit.
blog : http://fabdecret.blogspot.com/

La MarketPlace de Windows Phone 7

Aujourd'hui, le succès d'une plate-forme mobile de type Smartphone passe par les développeurs et leurs applications. Windows Phone 7 dispose de sa boutique en ligne : la MarketPlace.

Microsoft a voulu trouver le bon compromis entre un système peu onéreux pour le développeur, dont l'application renforcera l'attrait du système, tout en limitant au maximum les applications «poubelles », qu'elles soient inutiles ou tout simplement malveillantes. Voici les quelques chiffres à connaître en tant que développeur pour la Marketplace de Windows Phone 7 :

- Les outils de développement sont gratuits
- Le partage du revenu est de 70% pour le développeur (immuable) et 30% pour Microsoft (et éventuellement ses partenaires opérateurs)
- L'adhésion à Marketplace est gratuite pour les étudiants
- Pour les autres, il en coûtera 75 euros par an, afin de pouvoir poster un nombre illimité d'applications payantes et jusqu'à 5 applications gratuites.

Microsoft a limité volontairement le nombre d'applications gratuites que le développeur peut déposer.
L'objectif est que la Marketplace soit de qualité avec des applications innovantes et répondant aux attentes des utilisateurs mobiles.

Business Model	Explications	Implémentation	Facturation opérateur*	Partage de revenu
Gratuit	Modèle classique d'application gratuite pour l'utilisateur	Microsoft fournit les API nécessaires	NA	Pas de revenu
Application Payante	Modèle classique d'application payante pour l'utilisateur	Microsoft fournit les API nécessaires.	Possible	70% du revenu pour le développeur (quelle que soit la méthode de paiement, carte bleue ou facturation opérateur). 30% du revenu pour Microsoft
Sponsorisé par la publicité	L'application inclut des publicités (pour des produits, d'autres applications …)	Microsoft ne fournit pas les API nécessaires au moment du lancement. Il faut donc passer par des solutions tierces	NA	100% du revenu pour le développeur, les éventuels frais de la solution tierce sont à la charge du développeur
Freemium	L'application propose certaines fonctionnalités gratuitement mais pour accéder à la version complète il faudra que l'utilisateur final paie une certaine somme	Les API dites de « try & buy » sont fournies par Microsoft	Possible	70% du revenu pour le développeur (quelle que soit la méthode de paiement, carte bleue ou facturation opérateur), 30% du revenu pour Microsoft
Version d'essai limitée dans le temps	L'application fonctionne pendant un temps limité, il faut ensuite s'acquitter d'une certaine somme pour continuer à utiliser l'application	Les API dites de « try & buy » sont fournies par Microsoft	Possible	70% du revenu pour le développeur (quelle que soit la méthode de paiement, carte bleue ou facturation opérateur), 30% du revenu pour Microsoft
Abonnement	L'application nécessite un abonnement (ex journaux)	Microsoft ne fournit pas les API nécessaires. Le paiement peut se faire en dehors de Marketplace ou dans l'application via une solution tierce ou implémentée par le développeur	Non	100% du revenu pour le développeur, les éventuels frais de la solution tierce sont à la charge du développeur
Paiement in app Premium download In app purchase	Le contenu de l'application peut s'enrichir via des micropaiements dans l'application	Microsoft ne fournit pas les API nécessaires au moment du lancement. Il faut donc passer par des solutions tierces ou implémentées par le développeur	Non	100% pour le développeur, les éventuels frais de la solution tierce sont à la charge du développeur

** Facturation opérateur signifie que le montant de l'achat fait est directement prélevé sur la facture opérateur de l'utilisateur. Cette méthode permet des taux de conversions bien plus élevés que lors de l'achat par carte bleue.*

L'unique plateforme de distribution de vos applications dans le monde entier

La Marketplace Windows Phone 7 est l'unique moyen d'accès aux applications. Pour le consommateur final, aucun chargement d'application en dehors de la Marketplace n'est possible. Elle est présente sur l'écran d'accueil de tous les utilisateurs de Windows Phone 7 dans le monde.

Avec la Marketplace Windows Phone 7, les développeurs peuvent distribuer leurs applications dans tous les pays du monde où Windows Phone est présent. La découverte des applications est possible par des recherches, catégories et top applications mais également de manière éditoriale où les meilleures applications seront mises en avant dans chacun des pays, à la discrétion des équipes Microsoft.

De la version de test à l'achat !

Windows Phone 7 propose une fonction très intéressante pour le développeur « Essayer et acheter » (Try & Buy). L'appel à la méthode « IsTrial() » renverra un booléen qui vous permettra de savoir si l'utilisateur a choisi l'option « try » ou l'option « buy » dans Marketplace, et vous pourrez donc proposer un mode « trial » ou « lite » de vos applications. Si vos utilisateurs souhaitent aller plus loin, ils peuvent alors acheter la version complète de l'application en un clic. Pour le développeur cela ne représente qu'une seule base de code à maintenir, pour deux versions de l'application. D'autre part Microsoft propose dans certains pays une facturation opérateur (l'achat d'une application est directement effectué sur la facture opérateur sans passer par la carte bleue), cela engendre mécaniquement un taux d'achat beaucoup plus élevé car à aucun moment l'utilisateur n'a besoin de renseigner son numéro de carte bancaire. Avec les mécanismes de notifications en push qui permet d'être très visible sur l'écran d'accueil, et les mises à jour automatisées, vous pourrez augmenter l'usage de vos applications et leur présence dans l'esprit de vos utilisateurs. Enfin Microsoft vous offre la possibilité d'opérer des programmes de bêta pour vos applications afin de gérer vos tests mais aussi de fournir vos applications en avance de phase à la presse, au site de test d'application etc. Des listes d'utilisateurs « bêta » pourront être identifiées et auront un accès privilégié aux dernières versions de votre application avant sa publication générale.

UNE ÉTAPE OBLIGATOIRE : LA CERTIFICATION

Toutes les applications publiées sur Marketplace doivent être vérifiées par Microsoft avant d'être disponibles pour les utilisateurs. 3 types de vérifications sont effectués.

Le premier type de vérification porte sur le type d'application que vous développez. Par exemple, Microsoft n'acceptera pas que vous développiez une application qui s'apparente elle-même à une marketplace permettant d'installer des applications : une espèce de « méta application ». Un autre exemple de règle relative à l'application est que l'utilisateur doit être averti en cas de gros transfert de données, ou si vous devez envoyer certains de ses détails personnels à des tiers ou à un service en ligne. Si vous utilisez la localisation de l'appareil vous devez vous servir du service de localisation « built-in » , et il en va de même pour le service de notifications en push.

Le second type de vérification porte sur le contenu en lui-même. Vous n'avez le droit d'utiliser que du contenu sur lequel vous avez des droits : pas question de streamer de la musique illégalement. Tout contenu diffamatoire, menaçant, haineux ou discriminatoire sera banni. La promotion d'alcool, de tabac ou de drogue est interdite, au même titre que le contenu « adulte » ou la violence. Toute promotion d'activité illégale (jeux d'argents dans certains pays, activités terroristes, consommation de stupéfiants…) est interdite. Enfin, et c'est sûrement la plus évidente, une vérification technique sera imposée : il faudra déjà fournir dans le package (le fichier .xap) tous les éléments nécessaire à la publication de l'application sur le Marketplace : icones, screenshots, etc. Evidemment l'application devra être stable, et ne pas impacter la fonctionnalité « téléphone » du terminal. L'application devra booter et lorsqu'elle est quittée ou interrompue se terminer dans un temps défini. Enfin l'application devra décrire dans son manifeste les « capabilities » dont elle souhaite se servir et s'y tenir. Ces capabilities sont utilisées pour créer la sandbox de l'application, toute application qui ne respecterait donc pas ces capabilities se verrait inopérante et donc refusée à la certification.

Microsoft veut adopter une politique transparente de validation des applications : tous les critères sont donc déjà documentés, publics (sur http://developer.windowsphone.com) et par l'intermédiaire du portail développeur vous pouvez contacter Microsoft pour avoir toutes les clarifications nécessaires sur ces règles. Aucune n'est censée porter à confusion. Dernier point, s'il devait arriver que votre application échoue à la certification, Microsoft fournit un rapport de test complet et détaillé expliquant les points bloquants. Dans tous les cas, tous les tests sont exécutés, commentés, et les étapes de reproduction du bug spécifiées. De quoi repartir sur de bonnes bases avant la prochaine soumission !

En ce qui concerne les business models, quasiment toutes les formes sont autorisées : applications gratuites, payantes et application freemium (version limitée gratuite et payante pour la version complète via l'API Try & Buy). Tout le cycle de la soumission au suivi des paiements est géré via le portail unique, qui fournira également des statistiques sur les ventes de vos applications et vous permettra d'optimiser les éventuelles campagnes de promotion.

Pour aller plus loin :

http://developer.windowsphone.com

■ Antoine **Markarian**
Responsable des relations avec les éditeurs de logiciel - Microsoft France

→ Suite de la p.42

Ajout de la procédure stockée de recherche de proximité

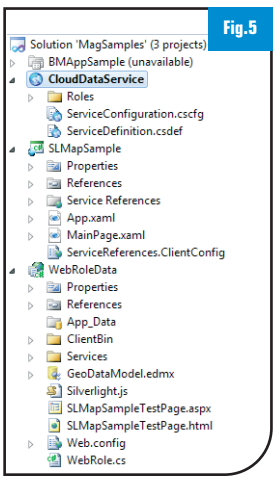
Dans le cadre de l'application réalisée, on souhaite effectuer une recherche à proximité d'un point déterminé grâce aux informations de latitude et longitude passées en paramètres. Le rayon de recherche sera également un paramètre supplémentaire. Enfin, dans le cadre de cet exemple, la fonction prend un paramètre supplémentaire permettant d'exclure les messages de l'utilisateur en cours. Voici le script nécessaire à la création de la procédure stockée exploitant les fonctionnalités des types géographiques :

```
CREATE PROCEDURE [dbo].[FindNearCenter]
 @lat as float, @lng as float, @radius as float,
 @idUser as int
AS
BEGIN
 SET NOCOUNT ON;

 -- Create the point
 DECLARE @center geography;
 SET @center = geography::Point(@lat, @lng, 4326)

 -- Create the Buffer
 DECLARE @centerBuffer geography;
 SET @centerBuffer = @center.STBuffer(@radius);

 --Return all POI in the search area
 SELECT Lat, Lon, Name, DateCreation
 FROM GeoUserInfo
 WHERE (@centerBuffer.STIntersects(GeoPosition)) = 1
 AND  @idUser <> Fk_IdUser
END
```

Mise en œuvre du service hébergé dans Windows Azure

Afin d'exposer les données, on choisit de réaliser un Web Service classique embarqué dans un Web Role. La page affichant l'application Silverlight, ainsi que l'application Silverlight en tant que telle seront également contenues dans ce Web Role hébergé sur la plateforme Windows Azure. Plusieurs éléments sont à considérer dans le cadre d'un hébergement sur Windows Azure, notamment en termes de maintenance ou de configuration, il faut alors habilement utiliser le fichier de configuration du service afin de s'économiser du temps et garantir le meilleur taux de disponibilité des applicatifs. Après avoir créé un projet de type Windows Azure Cloud Service au sein de Visual Studio. On ajoute un Web Role utilisé pour l'exposition des données et l'hébergement de notre application Silverlight. Il est ensuite nécessaire d'ajouter un projet de type Silverlight. On obtient alors l'architecture suivante : [Fig.5].

Fig.5



Modification de la classe du Web Role et bonnes pratiques pour la configuration

Pour chaque service hébergé sur Windows Azure, on retrouve la possibilité de modifier la configuration à travers l'interface en ligne du site d'administration de Windows Azure ou via l'API de management. C'est notamment à travers ce contenu de configuration que l'on va placer les chaînes de connexion à la base SQL. Les modifications apportées dans cette zone nécessitent de recycler le service et il est alors intéressant de modifier la déclaration par défaut générée à la création du projet. Voici le code nécessaire à la prise en compte de ces modifications qu'il faut alors ajouter convenablement dans l'évènement OnStart du Web Role :

```
// This code sets up a handler to update CloudStorageAccount
instances when their corresponding
// configuration settings change in the service configuration file.
CloudStorageAccount.SetConfigurationSettingPublisher((config
Name, configSetter) =>
{
 // Provide the configSetter with the initial value
 configSetter(RoleEnvironment.GetConfigurationSettingValue
(configName));

 RoleEnvironment.Changed += (sender, arg) =>
 {
   if (arg.Changes.OfType<RoleEnvironmentConfigurationSetting
Change>()
     .Any((change) => (change.ConfigurationSettingName == config
Name)))
   {
     // The corresponding configuration setting has changed,
propagate the value
     if (!configSetter(RoleEnvironment.GetConfigurationSetting
Value(configName)))
     {
       // In this case, the change to the storage account
credentials in the
       // service configuration is significant enough that the
role needs to be
       // recycled in order to use the latest settings. (for
example, the
       // endpoint has changed)
       RoleEnvironment.RequestRecycle();
     }
   }
 };
});
```

Dès lors que l'on modifie les propriétés de configuration, le service Azure se verra être recyclé afin d'intégrer ces nouveaux éléments.

Développement du Web Service et accès aux données

Afin de simplifier l'accès aux données dans notre exemple, on utilise Entity Framework qui nous permet de requêter simplement notre base mais également d'appeler la procédure stockée déclarée plus tôt. Il faut ainsi ajouter un élément de type «ADO.Net Entity Data Model » dans le projet du Web Role en indiquant dans l'assistant de génération les paramètres de connexion vers la base SQL Azure. On retrouve la chaîne de connexion inscrite dans le fichier de configuration de l'application Web, mais cette infor-

mation sera externalisée dans le fichier de configuration du service hébergé dans Windows Azure pour garantir une maintenance optimale. On pourra ainsi modifier en ligne le fichier de configuration sans avoir à redéployer le role.
Voici un extrait du contenu du fichier de configuration :

```
<ServiceConfiguration serviceName=»CloudDataService» xmlns=»
http://schemas.microsoft.com/ServiceHosting/2008/10/Service
Configuration»>
  <Role name=»WebRoleData»>
    <Instances count=»1» />
    <ConfigurationSettings>
      <Setting name=»DiagnosticsConnectionString» value=»Default
EndpointsProtocol=https;AccountName=YOURACCOUNT;AccountKey=
ACCESSKEY» />
      <Setting name=»SqlAzureConnectionString» value=»metadata
=res://*/GeoDataModel.csdl|res://*/GeoDataModel.ssdl|res://
*/GeoDataModel.msl;provider=System.Data.SqlClient;provider
connection string=&quot;Data Source=YOURURL.database.windows
.net;Initial Catalog=GeoSampleDb;User ID=USER;Password=PASSWORD
;MultipleActiveResultSets=False&quot;» />
    </ConfigurationSettings>
  </Role>
</ServiceConfiguration>
```

Il faudra alors initialiser le contexte Entity Model en modifiant le constructeur pour lui passer la chaîne de connexion définie dans le fichier de configuration du service :

```
public GeoDataService()
{
  string constring = RoleEnvironment.GetConfigurationSetting
Value(«SqlAzureConnectionString»);
  this.ctx = new GeoSampleDbEntities(constring);
}
```

On ajoute ensuite un simple Web Service pour exposer nos données, celui-ci expose la méthode qui interroge la procédure stockée à travers le contexte Entity Model précédemment généré :

```
[WebMethod]
[ScriptMethod]
public List<GeoUserInfo> FindNearUserInfo(double lat, double lng,
double radius, int idUser)
{
  return this.ctx.FindNearCenter(lat, lng, radius, idUser).
ToList();
}
```

Il suffit enfin de déployer ce Web Role sur le service hébergé dans Windows Azure.

BING MAPS FOR ENTERPRISE ET LE CONTRÔLE SILVERLIGHT

Après avoir déployé le Web Service d'exposition des données, nous réalisons le client riche Silverlight qui consommera celles-ci à travers un contrôle de cartographie Bing Maps.

Présentation de la technologie et authentification

Pour réaliser cette application, nous utilisons le contrôle Silverlight de la plateforme Bing Maps for Enterprise qu'il sera nécessaire d'installer et de référencer dans le projet. Ainsi, les éléments sont disponibles à travers les liens regroupés à l'adresse suivante : http://www.microsoft.com/maps/developers/. Après avoir installé les éléments, il faut ajouter les références vers les assemblies utilisées situées par défaut dans le dossier : **C:\Program Files (x86)\Bing Maps Silverlight Control\V1\Libraries** Après avoir convenablement référencé ces assemblies, il devient alors possible d'implémenter le contrôle Silverlight de la plateforme Bing Maps. Ce composant par défaut requiert l'utilisation d'une clé d'application qu'il est nécessaire de générer depuis le site : http://www.bingmapsportal.com/ [Fig.6]. Il faut ensuite ajouter le Map Control dans l'application Silverlight et ajouter la propriété spécifiant cette clé, ceci peut être réalisé directement depuis le code déclaratif XAML :

```
<UserControl x:Class=»SLMapSample.MainPage»
  xmlns=»http://schemas.microsoft.com/winfx/2006/xaml/presentation»
  xmlns:x=»http://schemas.microsoft.com/winfx/2006/xaml»
  xmlns:m=»clr-namespace:Microsoft.Maps.MapControl;assembly
=Microsoft.Maps.MapControl»>
  <Grid x:Name=»LayoutRoot» Background=»White»>
    <m:Map x:Name=»map»
      CredentialsProvider=»YOURKEY»></m:Map>
  </Grid>
</UserControl>
```

Application name	Key / URL
Sample	AmumskLy7[illegible]TaJUiQh0 http://localhost:56479/ Evaluation/Trial

Fig.6

A cette étape, on peut tout à fait lancer l'application qui initialisera un contrôle de cartographie Bing Maps de base avec un centre et un style de vue adapté.

Mise en œuvre de la recherche de proximité

Pour mettre en œuvre la recherche de proximité dans le cas présent, on choisit de présenter un scénario relativement simple, l'utilisateur va pouvoir récupérer, à partir de sa position, les dernières activités des utilisateurs situés à proximité. Dans le cadre de cet exemple, nous nous intéressons principalement à la récupération des données, la capture de la position géographique courante n'est pas traitée. Celle-ci sera par ailleurs simulée en capturant le clic au sein du contrôle qui déclenchera la requête des données. L'accès aux données est effectué en appelant le Web Service d'exposition qu'il est nécessaire de référencer depuis le projet Silverlight en utilisant l'adresse du service hébergé. Pour récupérer les coordonnées associées au clic sur la carte, on associe l'événement OnClick qui permettra de récupérer les éléments sur la recherche, de cette manière dans le code d'initialisation de contrôle :

```
// Init map events
this.map.MouseClick += new EventHandler<MapMouseEventArgs>
(map_MouseClick);
```

L'événement associé à ce clic est :

```
void map_MouseClick(object sender, Microsoft.Maps.MapControl.
MapMouseEventArgs e)
```

```
{
  // Get the clicked location
  Location loc = this.map.ViewportPointToLocation(e.Viewport
Point);

  // Async call
  this.client.FindNearUserInfoAsync(loc.Latitude, loc.Longitude,
10000, 1);
// 10 km radius; IdUser=1
}
```

Dans ce même constructeur, on initialise la connexion au Web Service d'exposition des données :

```
// Init the WS client and async events
Uri source = Application.Current.Host.Source;
string wsUri = string.Format(«{0}://{1}:{2}/Services/GeoData
Service.asmx»,
  source.Scheme, source.Host, source.Port);
this.client = new GeoDataService.GeoDataServiceSoapClient
(«GeoDataServiceSoap», wsUri);
this.client.FindNearUserInfoCompleted += new EventHandler
<FindNearUserInfoCompletedEventArgs>(client_FindNearUserInfo
Completed);
```

La récupération des résultats permet d'ajouter les points sur la carte à proximité du point cliqué simulant la position géographique de l'utilisateur, ce traitement est effectué dans l'événement asynchrone :

```
void client_FindNearUserInfoCompleted(object sender, FindNear
UserInfoCompletedEventArgs e)
{
  if(e.Result != null)
  {
    foreach (var curElement in e.Result)
    {
      // Create the pushpin
      Pushpin pin = new Pushpin();
      pin.Location = new Location((double)curElement.Lat, (double)
curElement.Lon);
      ToolTipService.SetToolTip(pin, curElement.Name);

      // Add the element on the map
      this.map.Children.Add(pin);
    }
  }
}
```

L'application réalise le scénario simple que l'on souhaitait, voyons maintenant comme exposer cette fonctionnalité sur le portail Bing Maps.

Création d'une Bing Maps App

Avant de se lancer dans le développement d'une Bing Maps App, il est nécessaire de comprendre ce qu'est une Bing Maps App et saisir l'intérêt d'une telle application.

Introduction aux Bing Maps Apps

Il existe actuellement un grand nombre d'applications et l'ouverture de ce média aux développeurs tiers est une réelle aubaine qui s'est par ailleurs déjà matérialisée par un concours « King of Bing Maps » qui a récompensé les meilleures applications Bing Maps [Fig.7]. Une fois un de ces blocs activés, il devient possible d'exploiter ces applications directement sur le portail. On réalise alors l'opération inverse de l'intégration du contrôle de cartographie dans un site web spécifique [Fig.8]. Ici on ajoute une fonctionnalité directement sur le portail, en utilisant nos propres données, il devient alors possible de parler de Mash-in applicatif [Fig.9]. De cette manière, pour l'entreprise utilisant ce type d'application, il devient possible d'exposer ces données à encore plus de personnes.

Présentation technique

Concrètement, une application Bing Maps ne représente pas de caractéristiques différentes de l'application initiale à la différence près des emplacements pour la présentation des données. Dans la réalisation, le procédé implique l'utilisation des modèles de projet simplifiant notamment la mise en œuvre et le développement.
Quelques spécificités résident dans l'utilisation que l'on peut faire du panneau d'affichage dédié à l'application dont voici le détail : [Fig.10]. Ici l'application est décomposée en plusieurs sous-zones applicatives :

1 Titre et icône

2 Panneau applicatif : permet d'afficher des informations complémentaires voire dans certains cas des éléments graphiques permettant de modifier la requête

3 Zone principale de cartographie : permet d'afficher les résultats cartographiques sur la carte

Il est nécessaire d'adapter le code réalisé vers une application Bing Maps et de modifier le code généré afin d'intégrer le scénario

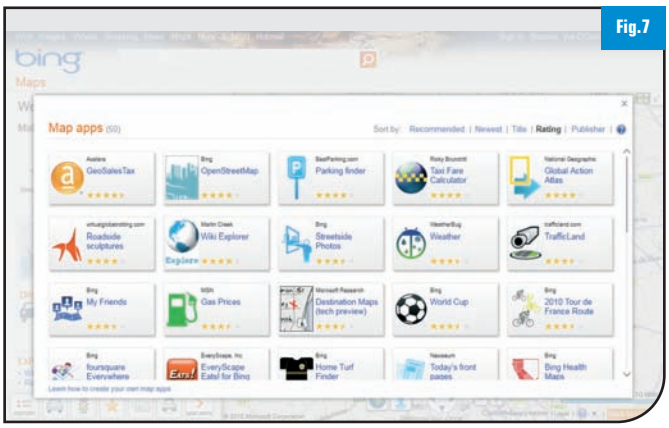


Fig.7

Fig.8

Fig.9

voulu. Pour tester l'application composée, le portail présente une application «Map app test tool» permettant de charger les assemblies générées et les fichiers de configuration directement au sein du portail à travers un assistant. Cette application est accessible en utilisant l'adresse suivante : http://www.bing.com/maps/explore/?developer=1 [Fig.11]

Réalisation de la Bing Maps App

Dans la solution Visual Studio, on ajoute une application de type « Bing Maps App » qui génère un projet type contenant plusieurs classes par défaut qui seront utilisées. Cette application n'est ni plus ni moins qu'une application Silverlight contenant des classes générées et il faut tout d'abord modifier la classe SamplePlugin correspondant à la classe principale de l'application Bing Maps. On ajoute l'association de l'événement de clic à travers la méthode Initialize() :

```
// Associate map click
this.DefaultMap.MouseClick +=
  new System.EventHandler<MapMouseEventArgs>(DefaultMap_MouseClick);
```

Au sein de cette méthode on retrouve la récupération des coordonnées cliquées, l'instanciation du client au Web Service et enfin l'appel asynchrone :

```
void DefaultMap_MouseClick(object sender, MapMouseEventArgs e)
{
  // Get clicked location
  Location curLoc = this.DefaultMap.ViewportPointToLocation
(e.ViewportPoint);

  // Initialize the client with ClientConfig and init async event
  var client = new GeoDataServiceSoapClient();
  client.FindNearUserInfoCompleted +=
    new   System.EventHandler<FindNearUserInfoCompletedEvent
Args>(client_FindNearUserInfoCompleted);
  // Async call
  client.FindNearUserInfoAsync(curLoc.Latitude, curLoc.Longitude,
100000, 1);
}
```

Dans l'événement asynchrone, on réalise à nouveau le traitement d'ajout des punaises :

```
void client_FindNearUserInfoCompleted(object sender, FindNear
UserInfoCompletedEventArgs e)
{
  foreach (var curElement in e.Result)
  {
    // Create a pushpin for the current item
    PointPrimitive temp = PushpinFactoryContract.CreateStandard
Pushpin(
      new Location()
      {
        Latitude = (double)curElement.Lat,
        Longitude = (double)curElement.Lon
      });

    // Add the pushpin on the map
    Entity entity = new Entity(temp);
    this.mainLayer.AddEntity(entity);
  }
}
```

Dès lors, en exécution on obtient le même scénario qui pourra alors être enrichi et qui est cette fois-ci, directement sur le portail Bing Maps.

RÉFLEXIONS ET ÉVOLUTIONS POSSIBLES

Le contrôle Silverlight de la plateforme Bing Maps for Enterprise n'est pas sans limitations techniques, il devient relativement déconseillé d'excéder une limite de 5 000 éléments sur le contrôle afin de ne pas s'exposer à des soucis de stabilité.

Dans le cadre d'un affichage de nombreux points ou d'informations plus riches, il est tout à fait envisageable d'utiliser les différentes technologies de la plateforme Windows Azure en combinant SQL Azure pour les données, les Blobs pour le stockage des tuiles générées depuis le Web Role hébergé et donc garantir les meilleurs performances au sein de l'application. C'est donc sans aucun doute que ces technologies vont continuer de se croiser et être utilisées en combinaison pour répondre à des cas simples comme des cas plus avancés où les temps de réponse et la montée en charge ne seront plus un problème délicat à aborder.

■ Nicolas **Boonaert**
MVP – Windows Live Platform - Bing Maps for Enterprise
Technical Specialist – Wygwam
Blog : http://blogs.developpeur.org/nicoboo/

■ Sébastien **Warin**
R&I Technical Leader – Wygwam
Blog : http://sebastien.warin.fr

Fig.10

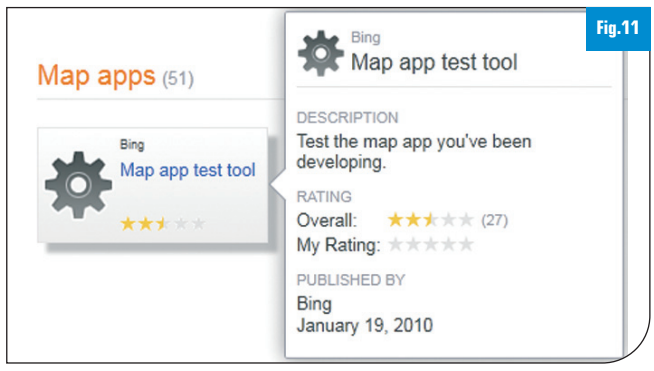


Fig.11

Microsoft Days

À la rencontre de votre potentiel

Développeurs, Partenaires, Décideurs et Professionnels de l'informatique

MICROSOFT VIENT À VOTRE RENCONTRE DANS 7 VILLES EN FRANCE !

Du 29 septembre au 3 novembre 2010

RENDEZ-VOUS DÉCIDEURS INFORMATIQUES

RENCONTRES TECHNIQUES

RENCONTRES PARTENAIRES 360

ÉVÉNEMENT GRATUIT

POUR VOUS INSCRIRE RENDEZ-VOUS SUR
WWW.MICROSOFTDAYS.FR

- **AIX** 29 et 30 septembre
- **PARIS** 5, 6 et 7 octobre
- **LYON** 11 et 12 octobre
- **TOULOUSE** 13 et 14 octobre
- **STRASBOURG** 18 et 19 octobre
- **LILLE** 20 et 21 octobre
- **NANTES** 2 et 3 novembre

ÉDITION SPÉCIALE

Windows Phone

Windows Azure

Microsoft Office 2010
ETONNEZ-VOUS

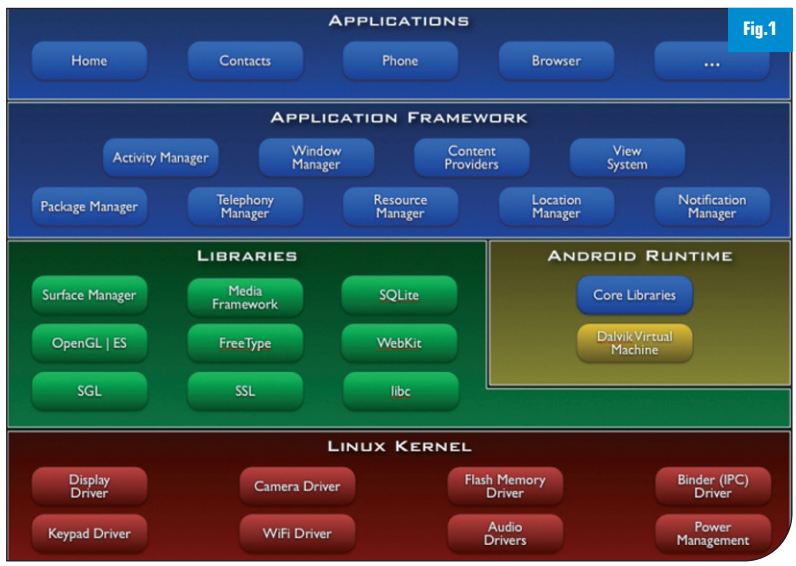
Microsoft Online Services

En partenariat avec Le magazine du développement PROgrammez! www.programmez.com — SOLUTIONS & LOGICIELS

Android Runtime vs Oracle : retour sur une polémique

Le succès connu par l'OS mobile de Google ne laisse pas indifférent les éditeurs concurrents qui aimeraient également tirer parti des retombées engendrées par l'effervescence autour de la plateforme Android. Pour ce faire, la stratégie d'Oracle semble consister à attaquer en justice Google sur le Runtime de la plateforme Android. En effet, ce fameux Runtime violerait un certain nombre de brevets logiciels détenus par Oracle. Plongée au cœur d'une polémique qui ne fait que débuter …

Le rachat de Sun par Oracle, il y a plus d'un an de cela, avait déjà mis en émoi la communauté Open Source, inquiète des futures décisions de la firme de Larry Ellison quant à l'avenir de Java et, plus largement, des technologies issues du portfolio Sun. Si les récentes annonces d'Oracle concernant OpenSolaris n'avaient fait que confirmer les craintes de la communauté, la décision prise par la direction d'Oracle de poursuivre en justice Google pour violation de plusieurs de ses brevets avec Android a fait l'effet d'une bombe dans le courant du mois d'août.



Architecture d'Android

Comment profiter du gâteau ?

Volonté affirmée de récupérer du cash en prenant une part du gâteau Android qui ne cesse de croître, attaque en règle contre l'Open Source tout simplement pour certains, la prise de position d'Oracle contre Google ne laisse personne indifférent au sein de la communauté. Pour beaucoup, Oracle se « tire une balle dans le pied » en attaquant un des acteurs majeurs de la communauté Java qui ne cesse de faire avancer la technologie depuis plusieurs années par ses innovations, tant sur le plan du web avec GWT, que sur le plan du mobile avec Android, et du cloud computing avec Google App Engine.

Cette omniprésence du géant de Mountain View qui n'hésite pas à proposer des technologies s'affranchissant des royalties, mais également des standards édictés par Sun pour la technologie Java sur les différentes plateformes du marché, énerve forcément du côté d'Oracle. Après plusieurs mois de négociations secrètes qui se sont avérées stériles avec Google, Oracle a donc décidé de passer à la vitesse supérieure et de partir en guerre contre Google, avec pour premier champ de bataille le terrain du mobile, et plus particulièrement Android, dont Google gère le développement via l'Open Handset Alliance qu'il a lui-même créée.

Concrètement, la plainte d'Oracle porte sur 7 brevets logiciels que l'éditeur possède désormais suite à son rachat de Sun. Ces mêmes brevets logiciels qui font la controverse aux Etats-Unis et que beaucoup considèrent comme dénués de sens dans le domaine logiciel mais cela est un autre problème. Toujours est-il que selon les dires d'Oracle, la violation de ces brevets concernerait la partie Runtime d'Android et plus particulièrement la machine virtuelle Dalvik qui permet d'exécuter (en le transformant, ndlr) du code écrit en langage Java sur la plateforme. La cour fédérale de Californie qui a reçu cette plainte n'a pour l'instant pas réagi. Du côté de Google, l'étonnement est de mise et l'on se place volontiers derrière l'Open Source en précisant que cette attaque ne concerne pas seulement leur société mais bien l'ensemble de la communauté. Ainsi, et fort logiquement, Google annonce

qu'il continuera à travailler à l'amélioration de la plateforme Android.
Prédire le dénouement de cette affaire est aujourd'hui plus que prématuré, étant donné qu'elle n'en est qu'à ses balbutiements. Les intentions réelles d'Oracle ne sont pas encore bien connues, de même que le système de défense que Google adoptera. Arrangements secrets à coup d'espèces sonnantes et trébuchantes ? Bras de fer prolongé par brevets logiciels interposés ? Décision de la cour en faveur de l'une ou l'autre des parties ? En outre, la bataille qui s'engage entre avocats se fera plus sûrement à coups d'arguments juridiques que d'arguments techniques, voire éthiques. Son issue demeurant ainsi bien plus incertaine. Beaucoup d'analyses ont fleuri ici et là dans les différents médias, détaillant plus en profondeur cette plainte et s'aventurant à livrer des pronostics plus qu'aléatoires sur son issue. Plutôt que de faire de même et de risquer de tomber dans une certaine redondance qui n'apporterait pas grand-chose au débat, nous nous intéressons dans la suite de cet article au cœur de la polémique, à savoir le Runtime Android, sous un angle technique cette fois sans considérations juridiques que nous laisserons aux différents avocats d'Oracle et de Google. La plongée au cœur d'Android nous permettra de mieux appréhender son architecture afin de donner la possibilité à chacun de se faire sa propre opinion technique sur le fondement des accusations d'Oracle.

Une architecture taillée pour l'embarqué

Android est découpé en 4 grandes couches s'appuyant sur un noyau Linux en version 2.6 comme le montre le schéma de la figure 1. Le Runtime Android se situe juste au-dessus du noyau Linux dans cette architecture et peut être lui-même divisé en 2 ensembles de base qui sont la machine virtuelle Dalvik et les bibliothèques Core qui permettent d'offrir un ensemble de fonctionnalités de base pour le développement d'applications Android [Fig.1].

Pour la conception de cette architecture, Google n'a pu partir d'une page blanche et créer sans contraintes les différentes couches qui la composent. En effet, Android a pour vocation d'équiper des plateformes de type smartphone, ce qui induit de facto de respecter un certain nombre de limitations inhérentes à ce type de périphériques. Ainsi, l'architecture proposée par Google a été faite pour s'adapter au mieux aux contraintes suivantes :

- Une vitesse de processeur limitée avec un cadençage minimum à hauteur de 250MHz
- Une mémoire RAM limitée ayant pour minimum un total de 64Mb pour l'ensemble du système qui ne dispose plus que de 20Mb une fois les services haut niveau démarrés
- Un OS sans espace de swap
- Un périphérique propulsé par une simple batterie

Ces différents choix de design s'inscrivent dans un souci constant d'économie tant au niveau énergie, qu'au niveau mémoire vive et mémoire de stockage. Il apparaît clairement aujourd'hui qu'un grand nombre de smartphones du marché dépassent allègrement ces limitations mais le but d'Android étant également de supporter un large éventail d'appareils mobiles, ces dernières doivent être respectées scrupuleusement.
Ces différentes problématiques ont rapidement amené Google à choisir une autre direction que celle consistant à utiliser une implémentation de Java ME pour la partie Runtime de son OS. Ce choix aurait pu paraître logique puisque Google souhaitait avant tout utiliser un langage fédérant un maximum de développeurs pour leur OS mobile et que le langage retenu était Java. Cependant, Java ME souffre d'un certain nombre de limitations majeures avec notamment un système de profiles ne permettant pas facilement de porter une application d'un mobile à un autre puisque rien ne garantit que 2 mobiles utilisent le même profile et donc disposent des mêmes fonctionnalités d'un point de vue hardware. En outre, le JCP et les différentes problématiques d'évolution de Java ME afférentes ont également dû jouer dans la balance côté Google, de même que le fait de devoir payer des royalties à Sun pour l'utilisation d'une implémentation Java ME sur Android.
Google a donc développé une machine virtuelle spécifique pour son OS. Nommée Dalvik par son créateur Dan Bornstein, en hommage à un village de pêcheurs éponyme en Islande dont sont issus ses ancêtres, cette machine virtuelle amène avec elle les avantages qu'offrent généralement ce type de machines, à savoir une fiabilité et une sécurité des programmes maîtrisées. D'autre part, son design a été réalisé sur mesure afin d'être en adéquation avec les contraintes évoquées précédemment. Ainsi, la machine virtuelle Dalvik est faite pour les environnements embarqués avec comme cible première les smartphones mais également, du fait de sa souplesse et de son extensibilité, des systèmes un peu plus puissants tels des notebooks ou des tablettes tactiles.

Un bytecode spécifique Dalvik

Dans un environnement Java standardisé par Sun, le code source écrit en Java est compilé dans un bytecode stocké dans des fichiers .class dont le format est spécifique. Lors de l'exécution du programme, la JVM lit et traite le contenu de ces fichiers .class. La stratégie mise en place par Sun consiste à créer un fichier .class par classes Java effectives. Prenons pour exemple le code source Java suivant contenu dans un fichier Test.java :

```
class Simple {
  private String doSomething(String var){ … }
  private String doOther(){ … }
}
public class Test {
   protected class InnerTest1 {}
   protected class InnerTest2 {}
   public static void main(String[] args) {
      new ActionListener() {
         public void actionPerformed(ActionEvent e) {}
      };
   }
}
```

La compilation de ce code via le compilateur javac produira en sortie les 5 fichiers .class suivants :

- Simple.class correspondant à la classe Simple
- Test.class correspondant à la classe du fichier source éponyme Test
- Test$1.class correspondant à la classe anonyme implémentant l'interface ActionListener
- Test$InnerTest1.class et Test $InnerTest2.class correspondant respectivement aux inner-classes InnerTest1 et InnerTest2

Ces fichiers .class contiennent un certain nombre d'informations dupliquées, ce qui s'avère coûteux en mémoire. Pour une plateforme telle qu'Android où chaque octet gagné compte, il était nécessaire de créer un nouveau format de bytecode permettant d'éliminer ces redondances et ainsi de gagner de l'espace mémoire. Le format Dex est né de ce constat. Sa force principale est de tirer parti du partage de données entre classes afin d'obtenir un bytecode allégé au maximum. Ainsi, un fichier .dex contient plusieurs classes Java et permet de partager des informations communes à ces dernières dans les différentes zones qu'il contient. La figure 2 offre un parallèle intéressant entre l'organisation des fichiers contenant le bytecode Java destinés respectivement à la JVM et à la machine virtuelle Dalvik [Fig.2]. Globalement, on s'aperçoit que les différentes zones présentes dans les 2 formats de fichiers n'ont pas de différences majeures. La principale zone discriminante est la zone « Heterogeneous Constant Pool » qui regroupe des données constantes de différente nature. Chaque fichier.class en contient une qui lui est propre alors qu'un fichier .dex contient 4 grandes zones de constantes communes à l'ensemble des classes qu'il représente. Ainsi, ces zones permettent de stocker les constantes de type string, les noms de champs, de variables, de classes, d'interfaces ainsi que les noms de méthodes.

Fig.2

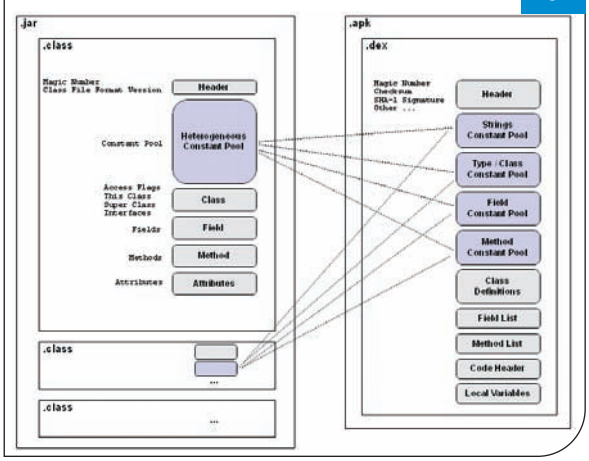
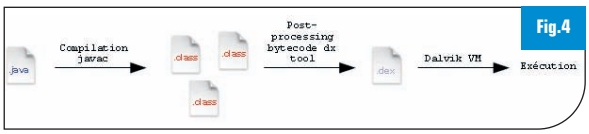


Fig.3

Code	Fichier JAR sans compression (KBytes)	Fichier JAR avec compression (KBytes)	Fichier DEX sans compression (KBytes)
Common Systems API	21 445 (100%)	10 662 (50%)	10 312 (48%)
Web Browser App	470 (100%)	232 (49%)	209 (44%)
Alarm Clock App	119 (100%)	62 (52%)	53 (44%)

Comparaison de la taille des fichiers JAR et DEX

Fig.4



Process de compilation Android

Concrètement, en reprenant l'exemple de la classe *Simple* définie précédemment, la constante de type Ljava/lang/String va se répéter à plusieurs reprises dans le bytecode produit par le compilateur javac de Sun. Cette redondance n'est plus présente dans un fichier .dex puisque la constante de type est stockée une fois et par la suite une référence est placée à chaque endroit où elle est utilisée, ce qui réduit au strict minimum la répétition des constantes. La conséquence de ce choix de design réside dans le fait que les fichiers .dex contiennent logiquement beaucoup plus de pointeurs ou de références que les fichiers .class.

Tout ce travail autour du partage des données entre classes dans le bytecode des fichiers Dex s'avère extrêmement payant au final. Un fichier .class étant constitué à 60% de données stockées dans la zone de constantes, le passage au format .dex permet de diviser par deux en moyenne la taille des fichiers contenant le bytecode comme le montre le tableau de la [Fig.3].

Bien évidemment, le gain immense en termes de mémoire apporté par le format DEX n'est pas sans conséquence. Ainsi, la machine virtuelle Dalvik vient avec un Garbage Collector spécifique dont la stratégie doit respecter les zones mémoires partagées. Chaque application étant lancée dans son propre processus Android et sa propre Dalvik VM, le Garbage Collector reste indépendant entre applications même si des données sont partagées. Pour s'en sortir, le Garbage Collector utilise un système de bits de marquage indiquant si un objet particulier est dans un état atteignable ou non et doit être détruit ou non.

Enfin, un mot sur le process amenant à la création d'un fichier .dex qui sera exécute par la VM Dalvik. Le code source écrit en langage Java est tout d'abord compilé via un compilateur standard tel que javac, ce qui produit un certain nombre de fichiers .class. Ces derniers fichiers sont ensuite transformés et agrégés en un fichier .dex par l'outil dx du SDK Android de Google, ce que résume brièvement le schéma de la [Fig.4].

Zygote ou la duplication rapide de VM

Comme expliqué dans ce qui précède, Android démarre une machine virtuelle Dalvik pour chaque application qu'il lance. De ce fait, la création d'une Dalvik VM doit être performante de même que son temps d'initialisation tout en gardant à l'esprit que la vitesse du processeur reste limitée sur les appareils de type smartphone, cibles d'Android. En outre, le partage de données s'avère nécessaire entre chacune des VM lancées aussi bien pour gagner en espace mémoire qu'en temps d'exécution en évitant un démarrage à « froid » à chaque création de Dalvik VM.

La solution apportée par Google à ces problématiques est un concept répondant au doux nom de Zygote. Il s'agit d'un processus unique qui permet le démarrage rapide de Davik VM. Le processus Zygote prend en compte le fait qu'un grand nombre de données peuvent être partagées entre chaque VM, tant au niveau des classes des API Core qu'au niveau de certaines données accessibles seulement en écriture. Démarré au moment du boot système [Fig.5], Zygote

est lancé par le processus Init, ancêtre de tous les processus lancés dans les systèmes Linux.
Processus natif de d'Android, le Zygote pré-charge et pré-initialise les classes des API Core. Il va ensuite les mettre à disposition des machines virtuelles Dalvik qu'il créera. Le Zygote crée dans la foulée une première VM Dalvik qui servira de machine virtuelle mère à toutes les autres qu'il devra créer. Ainsi, Zygote peut créer ces machines à la demande via le mécanisme de fork de d'Android, ce qui est bien plus efficace en termes de temps qu'une création à « froid » et permet d'accélérer le démarrage de chaque application qui est lancée.
A titre de comparaison, dans la JVM standard de Sun, chaque instance de VM réalise un démarrage complet à chaque lancement et doit donc charger l'ensemble des API Core à chaque fois. Il n'y ainsi pas de partage d'informations entre ces différentes instances. Cependant, Sun possède un brevet logiciel décrivant en partie ce type de comportement, comme le souligne la plainte d'Oracle. Ce système de fork et de copie d'informations pour maximiser le temps de démarrage n'est néanmoins pas l'apanage de Sun puisque les systèmes Unix proposent ce concept depuis maintenant de nombreuses années, ce qui devrait annihiler la portée de ce brevet dans ce contexte.

Une VM à registres

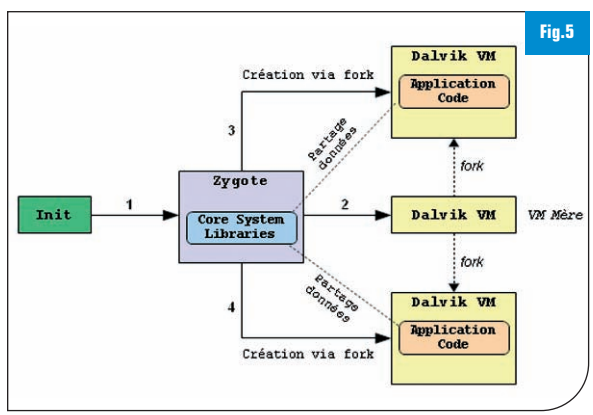
La JVM de Sun est basée sur une architecture à pile ce qui signifie que pendant l'exécution du code, les instructions sont stockées dans une ou plusieurs piles. Il s'agit du type de machine virtuelle le plus répandu à l'heure actuelle dans le monde Java mais dans le contexte ciblé par Android, ce type d'architecture n'est pas vraiment adéquat. Au lieu de cela, la machine virtuelle Dalvik est basée sur une architecture à registres au sein de laquelle les différentes instructions sont stockées dans des registres. Cette architecture permet de réduire le nombre d'instructions traitées (de l'ordre de 30 %) par la VM tout en réduisant également de manière notable le nombre d'unités de code manipulées (de l'ordre de 35 %). En contre partie, le flux d'instructions contient plus de données avec ce type d'architecture puisque ces dernières sont plus larges (à peu près 35 %).
Pour se rendre compte de manière plus précise de ces différences, prenons comme exemple le code source suivant présenté par Google lors de sa conférence Google I/O en 2008 :

```
public static long sumArray(int[] arr) {
    long sum = 0;
    for (int i : arr) {
        sum += i;
    }
    return sum;
}
```

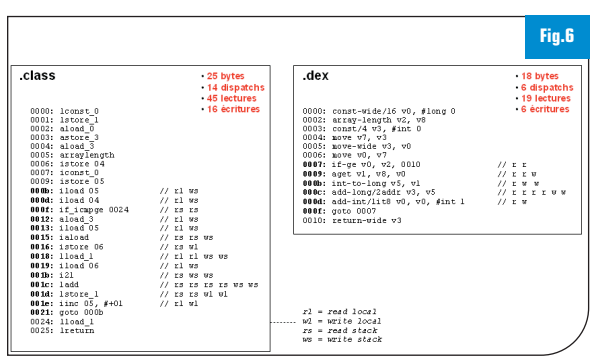
La figure 6 vient présenter le contenu des fichiers .class et .dex produits à partir de ce code source [Fig.6].
On constate effectivement que le principal inconvénient de l'architecture à registres de la VM Dalvik vient de la largeur des instructions qui est plus grande. Ce problème n'est cependant pas très coûteux en temps puisque différentes études ont montré que la charge supplémentaire engendrée par instructions est négligeable (de l'ordre de 1%). Ainsi, la machine virtuelle Dalvik et son architecture à registres sont bien adaptées aux machines mobiles et à leurs processeurs à la puissance limitée. Enfin, il est bon de savoir que ce choix architectural permet au final à Dalvik de présenter un temps d'exécution quasiment deux fois plus rapide que celui des JVM standard avec leurs architectures à pile.

Sécurité

La sécurité sous Android est directement implémentée au niveau de l'OS et garantit que chaque application s'exécute dans un environnement qui lui est propre. Au sein de cette sandbox, il ne peut donc y avoir de risques de collisions entre applications au niveau des données utilisateurs, des accès fichiers ou des accès réseaux par exemple. Chaque application est démarrée au sein de son propre processus et dans sa propre machine virtuelle Dalvik avec un user ID unique, ce qui permet de limiter les possibilités d'une application via l'OS. Ceci offrant également une plus grande stabilité des applications.
Une application doit définir dans un fichier Manifest les différentes fonctionnalités du système cible auxquelles elle souhaite accéder. La vérification de ces différentes permissions se fait à l'installation de l'application par le framework et l'OS se charge de les appliquer au moment de l'exécution. Précédemment dans cet article, nous avons fait état d'un mécanisme de partage de données entre processus. Ce partage s'effectue en toute sécurité puisque comme indiqué un peu plus haut, les API Core partagées ne peuvent être accédées qu'en lecture et non en écriture par les différents processus.
Contrairement à la JVM standard de Sun qui elle gère la sandbox à l'exécution des programmes Java, Dalvik VM n'est pas concerné par la sécurité au runtime. Toutefois, il est intéressant de noter que les API Core concernant la sécurité sont bien présentes au sein du Runtime Android, même si leur comportement n'est pas réellement implémenté avec par exemple



Processus de boot et interactions Zygote



l'absence de fichiers de politiques de sécurité. Ce concept de sandbox serait réglementé via des brevets détenus par Oracle. Cependant, ces derniers s'avèrent très génériques et surtout très larges, ce qui implique que la plupart des machines virtuelles existantes dans les autres langages puissent les violer également et l'on pense ici à la machine virtuelle de Microsoft pour la plateforme .Net notamment. En outre, ce concept de sandbox, bien que déposé par Sun, est sûrement plus ancien et de ce fait il devrait être facile aux avocats de Google de le rendre inapplicable.

API Core

Comme toute machine virtuelle qui se respecte, la VM Dalvik est liée à un ensemble d'API Core qui sont intégrées au Runtime Android et qui servent de base au développement d'applications. Là encore, Google a choisi une voie innovante et atypique pour Android puisqu'au lieu de partir des spécifications Java ME et d'en faire une implémentation, la firme de Mountain View a décidé de partir de Java SE. Afin là encore d'éviter tous problèmes de royalties, l'implémentation retenue n'est pas liée à Sun mais provient de la fondation Apache. Connu sous le nom d'Harmony, ce projet est une implémentation libre de Java SE en version 5.0 mais non certifiée par Sun car n'ayant pas passé les différents tests de compatibilité du TCK. Pour ses API Core, Google a quelque peu épuré la version de base d'Apache Harmony en enlevant tous les packages qui n'apportaient rien dans le contexte d'Android. Exit les bibliothèques graphiques standards Java pour le poste client (AWT et Swing) ainsi que les classes liées à la manipulation d'images (ImageIO) qui n'avaient pas d'intérêt ici. Les classes liées à RMI et aux applets sont également logiquement évincées. En contrepartie, la grande majorité des classes de base de Java sont là, avec bien entendu :

- java.lang
- Java I/O et NIO pour la gestion des entrées / sorties
- Les classes java(x).net pour le réseau
- L'API JDBC pour l'interfaçage avec les bases de données
- L'API Collection de Java avec les Generics
- L'API Reflection
- La nouvelle API Concurrency amenée par Java 5.0 pour la gestion de la concurrence sur la plateforme.

Cette liste est loin d'être exhaustive et l'on s'aperçoit très vite que ce que nous propose Google pour le Runtime Android va bien au-delà de ce que Java ME met à disposition du développeur. Google propose pour ce faire des API Core enrichies par d'autres bibliothèques sous les packages Android.*, qui regroupent des fonctionnalités à la portée assez large. Ainsi, on va de l'intégration de classes pour la gestion de bases de données SQLite en passant par des classes d'IHM adaptées aux systèmes embarqués ainsi que des API plus ou moins spécifiques à l'univers du téléphone mobile (appels téléphoniques, SMS, GPS, ...). La liste ci-dessous présente brièvement les autres bibliothèques annexes ajoutées au Runtime Android :

- Junit, la bibliothèque de tests unitaires de référence dans le monde Java
- Jakarta Commons Codec, une bibliothèque apportant un ensemble de codeurs / décodeurs classiques tels que Base64, Hex, ...
- Jakarta Commons HttpClient, permettant l'interrogation de sites web de manière aisée
- org.bluez pour la gestion du Bluetooth
- org.json afin gérer le format de données JSON
- com.google.android.maps pour interagir avec l'application Google Maps
- com.google.android.xmppService pour manipuler le protocole de communication libre XMPP, utilisé notamment par Google Talk
- javax.microedition.khronos.opengles, une implémentation de la JSR 239 proposant un binding pour OpenGL ES et qui consitue la seule API en provenance du monde Java ME.

Le seul défaut dans la cuirasse de Google en ce qui concerne les API Core pourrait venir d'Apache Harmony. En effet, Sun avait trouvé un accord avec Apache au sujet d'Harmony stipulant que les classes de ce projet ne pouvaient être utilisées que sur le desktop. En utilisant un sous-ensemble d'Harmony dans l'univers embarqué, Android ne respecte pas l'accord qu'Apache avait passé avec Sun et Oracle pourrait peut être en profiter pour appuyer là où ça fait mal et attaquer par ce biais Google.

Conclusion

Cet article nous aura permis d'entrer plus en profondeur dans les entrailles du Runtime Android qui constitue le cœur de l'attaque d'Oracle envers Google et Android. Les différents points techniques abordés montrent qu'il existe bel et bien de réelles différences techniques entre le Java standard de Sun et le Java que Google a mis au point pour son OS. Evidemment, les zones d'accointances sont nombreuses et ce n'est pas Froyo, la dernière version d'Android, qui viendra arranger les choses puisqu'elle apporte des capacités d'interprétation Just In Time (JIT) à la machine virtuelle Dalvik, ce que la JVM fait déjà depuis longtemps ... Ainsi, le débat va perdurer et le procès pourrait s'éterniser ! S'il y a bien des brevets qui semblent être transgressés, l'interprétation complexe induite par les brevets logiciels américains ne permet pas non plus de savoir de quel côté penchera la balance au niveau juridique. Cette bataille entamée pourrait également fort bien accoucher d'une souris via un accord secret entre Oracle et Google pouvant se chiffrer en millions de dollars, bien que cela semble peu probable à en croire les premières réactions des dirigeants du géant de la recherche Internet. *A suivre...*

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

egilia®

LEARNING

LE SPÉCIALISTE DE LA
FORMATION CERTIFIANTE
EN INFORMATIQUE
ET MANAGEMENT

Faire de vos succès
notre réussite

www.egilia.com

CONTACTEZ NOS CONSEILLERS FORMATION

N° National 0 800 800 900

APPEL GRATUIT DEPUIS UN POSTE FIXE

ANVERS . LIEGE . PARIS . LYON . LILLE . AIX-EN-PROVENCE .
STRASBOURG . RENNES . BRUXELLES
TOULOUSE . BORDEAUX . GENEVE . LAUSANNE . ZURICH .

Le build sous tous ses angles

Une application n'a pas d'existence sans avoir été construite. Le monde Java/JEE est doté de très nombreux outils de construction (build) dont le choix provoque souvent la polémique. Un script de build, comme tout code source doit comporter les critères nécessaires de réutilisabilité et de maintenance. Mais pour faciliter son intégration, le build est souvent adossé à un serveur d'intégration continue comme Hudson et différentes stratégies sont mises en place pour faciliter la construction

DANS QUEL CAS CHOISIR LES OUTILS DE BUILD ANT, MAVEN OU GRADLE ?

Le choix d'un outil est d'autant plus difficile que les besoins d'automatisation des projets ont augmenté considérablement ces dernières années, et vont au-delà de la construction traditionnelle des applications

Nous avons choisi de nous limiter aux outils suivants : Ant (pionner) couplé au gestionnaire de dépendances Ivy, Maven (le plus utilisé) et Gradle (le dernier né). Tous ces outils permettent de mettre en place nos scripts de construction, et incluent un mécanisme de gestion des dépendances externes (brique indispensable).

Ant, né dans les années 2000, est basé sur un modèle assez simple, composé de tâches, de propriétés, de cibles, … afin de définir un build. L'outil se concentre uniquement sur le build, c'est pourquoi on lui adosse quasi systématiquement le système Ivy pour la gestion de dépendances. Les outils de Ant et Ivy fournissent tous deux, des éléments de très bas niveau, offrant une très grande flexibilité dans la mise en place des scripts de build. Mais le principal défaut est qu'ils ne fournissent pas de services de plus haut niveau 'out-of the box'. Ce manque d'abstraction rend leur mise en place laborieuse sur un nouveau projet pour configurer les étapes de construction.

Concernant Maven, il est né de la standardisation des applications JEE. Il offre un système de build complet avec une standardisation dans la construction et fournit des conventions de build, un cycle de vie et un gestionnaire de dépendances intégré. Le cycle de vie offre une ossature de construction aux applications, ce qui permet d'identifier très clairement les étapes de build. Les conventions et les dépendances sont fournies grâce à un modèle projet consistant à définir les informations statiques, et Maven fait alors le reste.

Concernant Gradle, il s'agit également d'un système de build complet pour les applications Java/JEE. Il a été conçu pour répondre aux manques des outils de build précédents. Gradle fournit une approche déclarative comme Maven, permettant d'éviter à l'intégrateur de spécifier manuellement les éléments de son build et ainsi de se focaliser sur le 'quoi' et non le 'comment'. Gradle apporte une approche nouvelle en permettant de réutiliser nativement l'ensemble des tâches Ant existantes, le système de gestion de dépendances Ivy et de pouvoir lire et écrire des artefacts de type Maven (lecture et écriture de métadonnée Maven). Il simplifie les builds complexes et s'adapte au plus près des besoins de l'entreprise. Et pour les builds complexes, Maven apparaît souvent comme contraignant. C'est le cas par exemple, quand on veut changer la structure de projet ou le cycle de vie apporté par Maven.

Toutes les contraintes de cet outil sont dues majoritairement à sa nature. Il s'agit d'un framework de build imposant un processus pour la réalisation des scripts de build. Un script Maven doit adhérer à ce processus.

A contrario, Gradle a l'avantage de ne pas souffrir des problèmes rencontrés avec Maven. Plus précisément, Gradle intègre les conventions de Maven mais n'impose pas de processus rigide. Par exemple, Gradle ne fournit pas de standards organisationnels de répertoires, mais il peut offrir à la demande les mêmes standards que Maven. Plus précisément, Gradle est basé sur la notion d'API qui permet de transformer Gradle en un véritable kit de développement des scripts de build. Cela permet de répondre à tous les scénarios des projets, non prédictibles à l'avance. De plus, au delà d'être un kit, Gradle est aussi une 'tool box' en incluant au sein de sa distribution les outils Ant, Ivy et les Maven Ant tasks, qui permettent d'avoir un outil 'tout en un' complètement intégré.

En conclusion, le choix d'un outil va dépendre des besoins de votre application. Si vos applications suivent strictement un modèle standard Java et JEE, les trois outils vous conviendront avec un léger avantage à Maven, conçu pour répondre explicitement à ce besoin, et très simple d'apprentissage (particulièrement apprécié au sein de grosses équipes). En revanche, dès que vous avez ou aurez dans un futur proche des besoins particuliers, des solutions plus souples comme Ant/Ivy et/ou Gradle sont à envisager. Et dans ces dernières conditions, une avance est donnée à Gradle qui propose un modèle très extensible.

L'art d'écrire un script de build réutilisable et maintenable

Un script de build évolue et s'adapte au fur et à mesure des besoins du projet. C'est pourquoi, il est nécessaire de suivre quelques recommandations afin d'avoir des éléments pouvant facilement être réutilisés et maintenus. Voici quelques critères à suivre :

Respecter les conventions ou fournir des conventions manquantes

Les conventions fournies par certains outils apportent une certaine homogénéité et standardisation à travers différents projets d'une même équipe, la maintenance en est alors facilitée. Ainsi, il est préférable de respecter les conventions apportées par les outils comme dans les cas de Maven et Gradle. Ceci est d'autant plus vrai pour Maven où la mise en place d'un build spécifique s'écartant des conventions est difficile à mettre en œuvre. Pour l'outil Ant rien n'est fourni par défaut, il est recommandé de mettre en place ses propres conventions, comme une structure standardisée de ses répertoires du code. Il est conseillé de suivre les conventions des autres outils comme celles de Maven (reprises également par Gradle).

Faire un script portable entre les environnements

Un deuxième critère concerne les éléments de portabilité. Dans la mesure du possible, éviter de spécifier statiquement ('hard coder') des chemins de répertoires comme l'emplacement des sources. Il faut également éviter les éléments spécifiques à un système d'exploitation. Dans le cas où votre script de build met en œuvre des outils externes, essayez de ne pas rendre votre script dépendant statiquement de l'emplacement d'une installation. Ces types de recommandations permettent d'exécuter vos scripts à travers différents environnements. Il est également conseillé d'utiliser des propriétés pour chaque élément variable. Ces propriétés peuvent être stockées directement dans le script ANT ou dans un fichier de propriété séparé afin d'avoir une indépendance entre le script et les données. Et pour ces propriétés, essayer de choisir des noms explicites (par exemple 'dir.reports' est à privilégier au détriment de 'rpts'). Voir liste d'exemples pour Ant.

Réorganisation régulière

Un critère de maintenance souvent oublié par les intégrateurs consiste à réorganiser son script. Cette réorganisation impose de suivre deux règles principales. La première règle est de découper le build en termes de responsabilités. Ce découpage doit être avant tout fonctionnel puis ensuite technique. Cette approche n'est pas évidente à respecter pour certains outils comme Maven qui tend en premier à fournir un découpage technique. La deuxième règle consiste à identifier les éléments communs au sein de vos scripts et projets. Les éléments communs peuvent alors être factorisés sous forme de tâches Ant, Gradle ou de plugins Maven, Gradle (un plugin Gradle est constitué d'un ensemble de tâches réalisant une fonctionnalité).

Fournir de l'aide

Il reste indispensable de fournir une aide à ce que fait le script et à chacune de ses étapes. Nombreux sont les systèmes de build mettant à disposition des fonctionnalités automatiques de documentation à l'invocation d'une commande par un utilisateur. Ce principe est suivi par Maven ou chaque plugin embarque une description globale et une description pour chacune des actions. Dans le cadre de Ant ou Gradle, il vous faut inclure une description pour l'ensemble des tâches (cible Ant et tâche Gradle) que vous créez.

En résumé de cette partie, nous constatons que plus l'outil vous offre de flexibilité, plus les critères de maintenance à prendre en compte sont importants.

Mise en place de son build dans Hudson : quelques recommandations

Après avoir décrit son build (et respecté les critères de maintenance), la prochaine étape consiste à l'exécuter en continu dans un serveur d'intégration continue comme Hudson. Pour faciliter la mise en œuvre, voici quelques recommandations

Utiliser Hudson uniquement comme un orchestrateur

Hudson est capable d'exécuter un script de build ou directement des étapes de la construction d'un projet à travers des plugins spécifiques ou directement via des lignes de commandes. Se pose alors la question : quelles sont les étapes projets à mettre dans Hudson? Ces étapes peuvent êtres catégorisées en étapes obligatoires et en étapes optionnelles. Les étapes obligatoires (génération de code, compilation) définissent le noyau du processus. Par exemple, un artefact ne peut pas se construire sans une compilation préalable. En revanche, l'exécution des tests peut être qualifiée d'étape optionnelle car un livrable projet peut physiquement être construit sans que les tests soient exécutés. Au sein de Hudson, il est recommandé de laisser la gestion des étapes obligatoires à un outil de build qui sera orchestré par Hudson. Et pour les étapes optionnelles, c'est au libre choix de l'intégrateur, en fonction des besoins et de l'existant.

Laisser Hudson jouer le rôle d'outil d'infrastructure

Hudson joue un rôle prédominant dans le choix de la plateforme, des outils utilisés (version, emplacement, ...)

Liste d'exemples pour Ant

CAS 1

Ne pas faire

```
<mkdir dir="D:/data/project/reports"/>
<mkdir dir="./reports"/>
```

Solution

```
<property name="reports.dir" value="${basedir}/
reports"/>
<mkdir dir="${reports.dir}"/>
```

CAS 2

Néanmoins, mettre en place des propriétés n'est pas toujours suffisant.

Ne pas faire

```
<property name="tomcat.home" location="/Users/
gregory/Integ/apache-tomcat-6.0.29"/>
```

Solution

```
<property environment="env" />
<property name="tomcat.home" value="${env.tomcat
_home}"/>
```

La solution consiste à utiliser une variable d'environnement

pour exécuter un processus. De ce fait, il est par exemple conseillé d'éviter de spécifier l'utilisation statiquement des chemins des outils d'infrastructure dans les scripts mis en œuvre.

Avoir une étape de nettoyage indépendante de Hudson

Hudson met à disposition une action de suppression du contenu de son espace de build afin d'éviter les effets de bord lors d'une seconde instance d'un même processus. Mais cette fonctionnalité, bien que pratique, supprime tout le contenu sans aucun critère de sélection. C'est pourquoi la solution à privilégier consiste à mettre en place au niveau de son script de build une étape de suppression des sorties de l'exécution du processus précédent. Et cette étape sera évoquée en amont du reste du processus.

Ne pas mettre en place des variables dans son job Hudson

Hudson met à disposition une possibilité de paramétrer ses projets d'intégration. Cette fonctionnalité consiste à pouvoir demander à l'utilisateur les valeurs d'un ensemble de variables lors d'une exécution manuelle ou à prendre des valeurs par défaut lors d'une exécution automatique. Cette fonctionnalité introduit une dépendance vers un contexte variable et temporaire. Cela entraîne des manques de traçabilité du processus d'intégration. C'est pourquoi, nous recommandons plutôt d'avoir un processus configuré à travers les scripts de build ou statiquement au niveau de la configuration de Hudson.

Ne pas utiliser Hudson comme un système de gestion de dépendances

Hudson propose nativement un système permettant d'enchaîner différents processus (ou différentes étapes d'un processus). Malgré la très grande aisance de cette fonctionnalité au premier abord, celle-ci est extrêmement limitée dans l'expression des contraintes de dépendances. Cette limitation entraîne très rapidement des incapacités de mise en œuvre. Afin d'éviter tout point de blocage, il est souhaitable de segmenter son processus par unités indivisibles (ensemble d'étapes liées pour la réalisation de son processus) et de déclencher un processus Hudson uniquement en réaction à un stimuli correspondant à un changement d'environnement (nouveau code source, nouvel artefact présent dans un dépôt, …).

Limiter le nombre de plugins Hudson utilisés

Il existe aujourd'hui plus de 300 plugins pour enrichir fonctionnellement Hudson. Ce très grand nombre de plugins n'est pas sans conséquences. Le premier défaut concerne l'absence d'homogénéité dans la qualité des plugins, dû en partie à l'absence d'un processus permettant de qualifier une livraison. Le deuxième défaut concerne le fait qu'il manque souvent la version disponible du plugin qu'on souhaite installer, compatible avec sa version du noyau. Il en résulte d'énormes difficultés dans les organisations pour les responsables d'intégration en charge de constituer un ensemble de plugins pour une seule version précise du noyau. C'est pourquoi, il est recommandé de restreindre son infrastructure à un ensemble réduit de plugins indispensables à l'exécution de son processus d'intégration.

Sauvegarder sa configuration Hudson

De par sa nature, Hudson contient des éléments de configuration d'exécution du processus d'intégration. Afin d'assurer une continuité de service en pouvant réagir aux différent problèmes (crash, …), il est nécessaire de sauvegarder le contexte d'un job Hudson et/ou de mettre en place des mécanismes de réplications.

Quand se passer de Hudson et faire un build privé sur le poste du développeur ?

Par défaut, la remontée des modifications du code source d'un développeur déclenche un build Hudson. Le résultat de ce build peut être en échec à cause des incohérences avec le reste des modifications de l'équipe de développement. La branche d'intégration de l'outil de gestion de configuration logicielle (contenant toutes les modifications) est dite « cassée ». L'équipe doit alors mettre tous les moyens en œuvre pour remédier au plus vite à une exécution en échec du processus. Cette remise à niveau du processus peut prendre du temps, d'autant plus que le projet est volumineux et que le nombre de développeurs est important.

C'est pourquoi afin de maximiser le succès du build de l'équipe, il est souhaitable d'utiliser un build privé pour chaque développeur.

Un build de type privé consiste à exécuter le processus d'intégration, comportant les dernières modifications du développeur et la dernière version du code source de la branche d'intégration) et ceci sans impact sur le build de l'équipe. Ce build peut être localisé sur la machine du développeur ou sur une machine déportée au plus près de l'environnement final. Les ingrédients du processus d'intégration vont également dépendre de chaque type de build.

Sur la machine du développeur

Ce type de processus, communément appelé « build privé local » consiste à exécuter un processus d'intégration sur la machine du développeur, à sa demande avant la remontée de son code source dans le gestionnaire de configuration. L'objectif est de renforcer la confiance du développeur dans son commit en réduisant les risques de mettre en échec l'intégration globale de l'application. En termes de contenu, ce build privé dépend du contexte de chaque projet et de chaque développeur mais doit généralement comporter toutes les étapes obligatoires (génération de code, compilation, éventuellement packaging, …). Ensuite, le contenu dépend du temps global d'exécution, pour éviter de faire perdre trop de temps au développeur qui est en attente du résultat. Passé un délai de 3 à 4 minutes, l'utilisation du second type de build privé est alors à privilégier. Les équipes agiles sont très adeptes du build privé local, en partie

du fait que chaque développeur peut prendre la casquette de l'intégrateur et possède les connaissances nécessaires pour le choix des ingrédients de son build.

Sur la machine d'intégration se rapprochant de l'environnement final

Dans cette situation, le processus d'intégration n'est pas exécuté sur la machine du développeur mais sur une machine dédiée. L'objectif du 'build privé distant' est identique au premier type de build privé. En revanche, il est plus adapté quand le processus requiert des outils, ou plus généralement, un contexte d'exécution non disponible (ou ne pouvant être émulé) sur le poste du développeur.

Une fois que le processus est terminé, le développeur peut choisir de communiquer ses modifications ou non (ce choix peut être automatisé).

La mise en place du build privé distant dépend fortement de l'outil de gestion de configuration logicielle utilisé. De par sa nature, la mise en œuvre est facilitée dans le cas de l'utilisation d'un DVCS (Distributed Version Control System). A contrario, dans le cas de l'utilisation d'un système de gestion de configuration centralisé comme Subversion ou CVS, la mise en place de ce type de build est moins évidente. Mais des outils d'intégration continue comme JetBrains TeamCity proposent l'implémentation de ces types de build. En revanche, concernant Hudson, cette fonctionnalité n'est pour le moment pas fournie nativement. Il est alors nécessaire d'avoir des alternatives comme le découpage en branches au niveau de son gestionnaire de configuration logicielle centralisé. Chaque développeur possède sa propre branche de développement et un projet Hudson d'intégration est créé pour chaque branche. Le développeur est alors responsable de son intégration dédiée, isolé des autres développeurs. Une branche d'intégration et un job Hudson d'intégration en écoute sur cette branche sont toujours présents et nécessaires. Néanmoins, cette solution hérite des inconvénients de l'utilisation des branches (comme les problématiques de merges) et peut engendrer une certaine lourdeur de configuration et de maintenance de l'instance Hudson, d'autant plus que l'équipe est importante.

ADRESSER LES PROBLÉMATIQUES DE TRAÇABILITÉ ET DE REPRODUCTIBILITÉ DU BUILD

La traçabilité du build consiste à identifier clairement les ingrédients ayant participé à la construction des livrables : la matière première (le code source), les outils de construction et l'ensemble des ressources extérieures utilisées. Cette traçabilité permet de connaître avec exactitude le contenu d'un build afin de pouvoir reproduire un livrable avec la plus grande exactitude (nécessaire par exemple pour la correction d'une anomalie sur une ancienne version livrée).

Cette traçabilité ne doit être active que dans le cas où les éléments produits ont une vocation de persistance. Par exemple, les artefacts temporaires utilisés uniquement à la génération des livrables n'ont pas de raison d'être persistés (il s'agit uniquement d'objets volatiles).

On peut distinguer deux niveaux de traçabilité : une au niveau du code et l'autre au niveau de l'environnement de build.

L'identification du code source

Un binaire est une représentation en code machine, issue d'une transformation à partir d'une représentation humaine (le code source). Identifier un livrable va consister à poser une étiquette sur l'ensemble des fichiers du code source utilisés pour le build. Cette étiquette va prendre plusieurs formes en fonction de l'outil de gestion de configuration logicielle qui stocke les sources (un tag dans Subversion, un label dans Clearcase). Dans tous les cas, le mécanisme consiste à associer un identifiant unique à l'ensemble des fichiers source ayant participé au build (avec leur version exacte). Cela signifie que dans le cadre d'un outil de gestion de configuration qui n'est pas muni de révisions, l'étiquette doit être posée avant le build.

Il se présente ensuite le problème de la correspondance entre l'étiquette posée et les artefacts générés. Il existe pour cela plusieurs mécanismes. Une première approche consiste à utiliser un système de gestion d'artefacts associant l'artefact et un ensemble de métadonnées associées. Les métadonnées identifient de manière unique les artefacts et comportent une propriété correspondant à l'étiquette du code source desquels les artefacts sont issus. Si ces derniers ne sont pas gérés par un système extérieur, il est nécessaire que l'artefact porte l'identifiant de l'étiquette. Cela peut être à travers des propriétés embarquées dans l'artefact (propriétés fixées au moment de la construction) ou dans son nom (la production de l'artefact 'myProg-3.2.timestamp.zip' est associée à une étiquette portant le nom 'myProg-3.2.timestamp'). Dans chacune des deux situations, la qualité de l'étiquette est très importante.

Deuxième niveau de traçabilité : l'environnement de build

Au-delà de l'identification du code source ayant participé à la génération des livrables, l'environnement de build est important.

Dans un processus d'intégration continue, cet environnement est paramétré par le serveur d'intégration continue, responsable de l'ensemble des données d'infrastructure comme la version de l'outil de build et son emplacement.

Une mise en œuvre possible consiste à capturer l'environnement (prendre une photo) à chaque build, l'environnement d'infrastructure ayant participé au build. Pour ce faire, sur le marché des outils, le repository manager JFrog Artifactory offre cette fonctionnalité. Artifactory s'interface avec votre serveur d'intégration continue comme Hudson et votre outil de build comme Maven ou Gradle. A chaque build, un artefact est injecté dans le repository management. Sur cet artefact est attaché un objet de métadonnée nommé 'le build info' contenant sous forme de propriétés

l'environnement ayant participé à la création de cet artefact nouvellement stocké.

Cette capture de l'environnement au niveau du build Hudson est indispensable mais elle doit également s'accompagner dans certaines situations d'éléments de contrôles au niveau des outils de build eux-mêmes en raison d'un manque d'identification des dépendances d'infrastructures que fournissent certains outils. C'est le cas par exemple de Maven où les plugins Maven utilisés doivent avoir une version explicitement déclarée, y compris pour les plugins du noyau (par défaut depuis Maven 2.0.9). Pour éviter ces problèmes, il est recommandé l'usage du plugin 'maven-enforcer-plugin' pour contrôler l'uniformité du build sur les éléments de la version de la JDK (souvent négligée et pouvant amener des problèmes d'exécution comme dans l'utilisation des parseurs XML natifs), les versions de plugins…

Gregory **Boissinot**
Expert en intégration continue chez Zenika

Versions de plugins

```
 1 <plugin>
 2   <groupId>org.apache.maven.plugins</groupId>
 3   <artifactId>maven-enforcer-plugin</artifactId>
 4   <version>1.0-beta-1</version>
 5   <executions>
 6     <execution>
 7       <id>enforce</id>
 8       <goals>
 9         <goal>enforce</goal>
10       </goals>
11       <configuration>
12         <rules>
13           <requireMavenVersion>
14             <version>2.0.9</version>
15           </requireMavenVersion>
16           <requireJavaVersion>
17             <version>1.5</version>
18           </requireJavaVersion>
19           <requireOS>
20             <family>unix</family>
21           </requireOS>
22           <requirePluginVersions />
23         </rules>
24       </configuration>
25     </execution>
26   </executions>
27 </plugin>
```

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, 22 rue rené Boulanger, 75472 PARIS - ou par fax : 01 55 56 70 20

1 an : 30€ *au lieu de 36€, prix au numéro* *(Tarif France métropolitaine)* **- Autres destinations :** *CEE et Suisse : 36€ - Algérie, Maroc, Tunisie : 36€ , Canada : 48€ - Dom : 45€ Tom : 60€*

6 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 .....

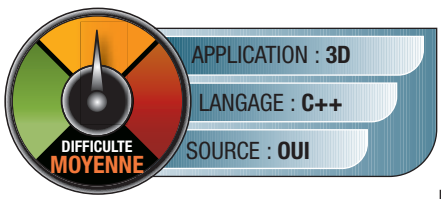
☐ *Je joins mon réglement par chèque à l'ordre de SOLUTIONS LOGICIELS* ☐ *Je souhaite régler à réception de facture*

Editeur : K-Now sarl au cap de 8000€ siège social : 6 rue Bezout, 75014 Paris

SL015

DirectX 11 – Shader Linkage

Nous continuons notre découverte de DirectX11 avec une autre grande nouveauté de ce dernier : Le Shader Linkage. Cette technologie répond à un vrai besoin issu de la communauté des développeurs 3D : assurer à la fois performance et maintenabilité de notre code.



Il est à noter que nous utiliserons à nouveau SlimDX pour le wrapper sur DirectX. Toutefois, la version actuelle en téléchargement est basée sur la version de février 2010 de DirectX. Or cette version de SlimDX ne comporte pas tous les éléments nécessaires pour faire du Shader Linkage. Vous trouverez donc dans la solution livrée avec cet article une version particulière de SlimDX (directement compilée à partir du dernier code source) qui intègre le support complet du Shader Linkage.

LES PROBLÉMATIQUES DE LA PROGRAMMATION AVEC LES SHADERS

La programmation des GPU peut faire penser à la programmation des CPU comme on la concevait il y a quelques dizaines d'années : chaque cycle est extrêmement important et la moindre boucle, le moindre test peut coûter plusieurs images par seconde et donc potentiellement pénaliser le rendu visuel.

Face à ce constat et au besoin toujours plus grand de shaders spécialisés, deux grandes solutions s'offrent au développeur :

- *Faire un shader global capable de tout gérer et paramétrable soit par des tests soit par des directives de compilations :* Cette solution présente l'avantage d'être facilement maintenable mais sa mise en œuvre est soit gourmande en ressources (si on passe par des tests) soit complexe à mettre en œuvre (si on passe par des directives de compilations)
- *Faire des shaders spécialisés :* Cette solution est performante car chaque shader est optimisé pour le résultat à produire. Par contre, on se retrouve vite avec plusieurs centaines de shaders et la maintenance devient alors un enfer.

C'est là qu'intervient le Shader Linkage. Cette technique offerte par DirectX 11 permet d'avoir un seul shader global dans lequel on peut brancher/débrancher des bouts de code de manière performante et avec une mise en œuvre aisée. Il faut imaginer cette solution comme des appels de méthodes que l'on aurait dans un langage plus standard.

PORTABILITÉ ET SUPPORT MATÉRIEL

La principale problématique est que contrairement à DirectCompute, le Shader Linkage ne fonctionne que sur une carte qui supporte pleinement DirectX11. Dans le cadre de notre exemple, nous allons activer le support du mode référence qui permet de faire du DirectX11 en mode totalement software :

```
// Création de notre device (on accepte les cartes dx11 uniquement)
FeatureLevel[] levels = {
    FeatureLevel.Level_11_0
 };

// Définition de notre swap chain
SwapChainDescription desc = new SwapChainDescription
desc.BufferCount = 1;
desc.Usage = Usage.BackBuffer | Usage.RenderTargetOutput;
desc.ModeDescription = new ModeDescription(0, 0, new Rational
(0, 0), Format.R8G8B8A8_UNorm);
desc.SampleDescription = new SampleDescription(1, 0);
desc.OutputHandle = parentForm.Handle;
desc.IsWindowed = true;
desc.SwapEffect = SwapEffect.Discard;

Device.CreateWithSwapChain(null, DriverType.Reference, Device
CreationFlags.Debug, levels, desc, out device11, out swapChain);
```

Comme on peut le voir, la création du device s'appuie sur le FeatureLevel11. Or si votre carte n'est pas compatible avec DirectX11 la création du device échouera. C'est pour cela que nous utiliserons *DriverType.Reference* à la place de *DriverType.Hardware*. Dans un premier temps, regardons le shader complet comme nous l'aurions écrit avec DirectX10 (en faisant le choix de la solution à base de booléens) :

```
float4 staticColor;
bool useStaticColor;

Texture2D diffuseTexture;
SamplerState diffuseSampler
{
    Filter = MIN_MAG_MIP_LINEAR;
    AddressU = Wrap;
    AddressV = Wrap;
};

// Effect
cbuffer globals
{
    matrix worldProjectionMatrix;
}
struct VS_IN
{
    float3 pos : POSITION;
    float2 uv : TEXCOORD0;
};

struct PS_IN
{
    float4 pos : SV_POSITION;
    float2 uv : TEXCOORD0;
};

// Vertex Shader
PS_IN VS( VS_IN input )
{
```

```
    PS_IN output = (PS_IN)0;

    output.pos = mul(float4(input.pos, 1), worldProjectionMatrix);
    output.uv = input.uv;

    return output;
}

// Pixel Shader
float4 PS( PS_IN input ) : SV_Target
{
    if (!useStaticColor)
      return diffuseTexture.Sample(diffuseSampler, input.uv);
    else
      return staticColor;
}

// Technique

technique10 Render10
{
 pass P0
 {
    SetGeometryShader( 0 );
    SetVertexShader( CompileShader( vs_4_0, VS() ) );
    SetPixelShader( CompileShader( ps_4_0, PS() ) );
 }
}
```

Ce shader s'appuie donc sur le booléen useStaticColor pour décider du rendu (ici on oscillera entre du rendu couleur pleine ou l'utilisation d'une texture). Or, les GPU sont beaucoup moins performants que les CPU dans le cadre de la prédiction de branchement et donc ce choix dynamique (comprendre : lors du rendu) impose un coût puisque le shader (et ici précisément le pixel shader) est exécuté des milliers de fois et donc le test a un impact fort.

QUAND L'OBJET VIENT À LA 3D

Pour répondre à ce problème, DirectX11 apporte une solution basée sur le développement objet. En effet, la gestion du Shader Linkage se base sur la déclaration d'interfaces (au sens développeur du terme) qui seront utilisées dans les shaders. Ces interfaces sont par la suite affectées à des instances de classes qui implémentent lesdites interfaces. Dans notre cas nous allons définir une interface qui permettra de récupérer la couleur :

```
// Interface
interface IMaterial
{
    float4 GetColor(float2 texCoords);
};
IMaterial abstractMaterial;
```

Par la suite, l'utilisation de cette interface se fait de manière standard :

```
// Pixel Shader
float4 PS( PS_IN input ) : SV_Target
{
    return abstractMaterial.GetColor(input.uv);
}
```

Il faut également définir des classes qui implémentent cette interface :

```
// Solid color material
class SolidMaterial : IMaterial
{
    float4 GetColor(float2 texCoords);
};

float4 SolidMaterial::GetColor(float2 texCoords)
{
    return staticColor;
}

// Texture material
class TextureMaterial : IMaterial
{
    float4 GetColor(float2 texCoords);
};

float4 TextureMaterial::GetColor(float2 texCoords)
{
    return diffuseTexture.Sample(diffuseSampler, texCoords);
}

// Instances
cbuffer cbInstances
{
    SolidMaterial cSolidMaterial;
    TextureMaterial cTextureMaterial;
}
```

Nous avons deux implémentations : une pour la couleur statique et une pour la texture. Pour notre shader, il faut également modifier la technique pour basculer sur du shader 5.0 :

```
technique11 Render11
{
    pass P0
    {
        SetGeometryShader( 0 );
        SetVertexShader( CompileShader( vs_5_0, VS() ) );
        SetPixelShader( CompileShader( ps_5_0, PS() ) );
    }
}
```

Il ne nous reste donc plus qu'à mettre en place le code qui permettra d'affecter à notre variable *abstractMaterial* (l'interface) l'une des deux instances que sont cSolidMaterial ou *cTextureMaterial*.

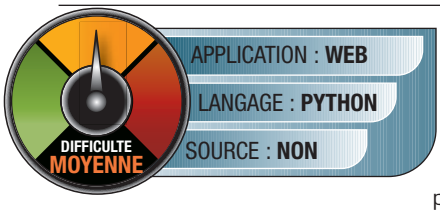
CONCLUSION

Le Shader Linkage est une solution professionnelle qui amène enfin une vraie maturité au développement de shaders. La direction que semble suivre DirectX et HLSL est très encourageante. On se retrouve enfin avec un langage de programmation des cartes graphiques évolué ET performant.

■ David **Catuhe** - *Bewise/Vertice*
blog : http://www.catuhe.com
site : http://www.bewise.fr I http://www.vertice.fr

Du Web vers OpenOffice : exportez en ODT

Vous avez une application web, et vous voulez exporter son contenu pour une lecture hors-ligne, pour impression ou pour archivage ? Le projet XHTML2ODT est fait pour vous !



XHTML2ODT est une bibliothèque permettant de convertir une page web en document ODT (OpenDocument). Même s'il s'agit d'un projet de développement relativement récent issu d'un auteur unique (moi-même), elle s'appuie sur une longue expérience du format ODT et a récemment atteint un état de maturité jugé suffisant pour en faire une promotion aussi large que possible. Tout d'abord, un petit point sur la licence : jusqu'à présent la licence de la bibliothèque était « GPL v2 ou ultérieure ». À l'occasion de la parution de cet article dans ***Programmez!***, elle devient « LGPL v2 ou ultérieure ». Concrètement, cela signifie plus de libertés pour le développeur : il peut intégrer la bibliothèque dans n'importe quelle application, même sous licence propriétaire. La seule contrainte reste donc la suivante : s'il modifie la bibliothèque elle-même (à l'intérieur de son application) et qu'il distribue l'application à un tiers, il doit alors aussi fournir à ce tiers les modifications effectuées sur XHTML2ODT (et uniquement celles-ci). Il n'est donc plus nécessaire de placer l'application entière sous licence GPL, comme c'était le cas avant. J'espère que ce changement de licence facilitera l'adoption de cette bibliothèque, le but final étant de faciliter la création et la manipulation de documents au format ODT.

HISTORIQUE

Le projet XHTML2ODT a débuté en décembre 2009, mais s'appuie sur des développements réalisés autour du format ODT depuis octobre 2007, dans le plugin d'export ODT de Dokuwiki (un wiki assez populaire écrit en PHP). Ce plugin permet d'écrire de la documentation de manière collaborative dans un wiki, puis d'exporter cette documentation pour pouvoir la transmettre à des tiers, après application d'une mise en page. Dokuwiki dispose d'une API assez complète pour les plugins d'export, leur permettant de générer le document ODT directement depuis le format interne.

En migrant mon blog vers Dotclear, j'ai voulu réaliser un plugin d'export ODT pour mes billets, mais Dotclear ne propose pas la même API que Dokuwiki : il stocke en base de données le contenu du billet au format HTML. La seule option était donc de convertir le HTML en ODT directement.

Je me suis inspiré des feuilles XSL d'un autre projet, docbook2odt, qui permet de convertir du format Docbook XML vers ODT. J'ai également décidé d'utiliser un document « modèle » pour le formatage. Le fonctionnement de ce plugin sera analysé dans cet article.

Quelques mois plus tard, j'ai eu l'occasion d'utiliser le gestionnaire de projets Trac ; le besoin d'exporter les pages du wiki en ODT s'est donc fait sentir. Trac stocke lui aussi les pages en HTML brut dans la base de données : la méthode à employer était donc très similaire à celle du plugin Dotclear.

J'ai décidé d'extraire des deux plugins la partie commune et générique de conversion HTML vers ODT, pour en faire un projet indépendant. Le fait d'avoir utilisé des feuilles de style XSL s'est révélé être un bon choix de conception, puisque Dotclear est en PHP alors que Trac est en Python. Le fait que tous les langages (ou presque) disposent d'un processeur XSLT permet de factoriser au maximum la conversion dans les feuilles de style, et de laisser une couche d'adaptation spécifique minimale.

LE FORMAT OPENDOCUMENT

Avant d'entrer dans les détails de la bibliothèque de conversion, il est important d'avoir une idée de la structure d'un document ODT. Le format ODT (OpenDocument Text) est un format normalisé de bureautique (ISO 26300) utilisé originellement dans la suite OpenOffice.org, mais adopté depuis par une dizaine d'autres applications.

Un document ODT est en fait une archive ZIP, contenant quelques fichiers XML. Pour manipuler un document ODT, il suffit donc d'ouvrir le ZIP, de modifier les fichiers XML, et de refermer le ZIP. Un fichier ODT classique contient deux fichiers principaux : content.xml et styles.xml. Les images du document sont placées dans un sous-dossier de l'archive ZIP, et sont référencées par leur nom dans le fichier content.xml.

ARCHITECTURE DE LA BIBLIOTHÈQUE

Entrons maintenant dans le vif du sujet : la bibliothèque XHTML2ODT. La conversion du (X)HTML vers ODT est assez complète puisqu'elle prend aujourd'hui en compte la majorité des balises du XHTML 1.0, mais il y a deux limitations principales :

- le style de la page web n'est pas converti, seul le contenu l'est. Pour gérer le style du document produit, il faut utiliser le document ODT « modèle ». Il n'y a donc pas d'interprétation du CSS, des attributs « style » ou des balises « font ».
- les formulaires HTML ne sont pas traduits en formulaires ODT. Techniquement ce serait possible, mais cela n'aurait pas beaucoup de sens puisqu'on y perd l'interactivité.

Si le document modèle ne contient pas tous les styles utilisés dans la page web, ils seront ajoutés en utilisant des styles basiques dont le rendu est proche de celui du navigateur web. Du point de vue de la qualité du code, on peut noter que le projet contient actuellement 118 tests unitaires (permettant donc d'éviter des régressions futures), et que les documents ODT produits passent les tests de validation de l'OpenDocument Fellowship avec succès. Les feuilles XSL et les scripts fournis sont aussi très largement commentés.Voyons à présent en détail le fonctionnement d'XHTML2ODT. En décompressant l'archive téléchargée, on trouve notamment les fichiers suivants :

- README.txt : une documentation rapide sur le projet, en anglais ;

- xsl : le dossier contenant les feuilles de style XSL ;
- xhtml2odt.py et xhtml2odt.php : deux scripts mettant en œuvre ces feuilles de style, pour servir d'exemple ;
- template.odt : un exemple de fichier « modèle » ODT, à utiliser avec les scripts ;
- tests : le dossier contenant les tests unitaires de la bibliothèque ;
- Makefile : un fichier Makefile pour installer la bibliothèque, les scripts associés, et générer leur documentation ;
- doc-php et doc-py : la documentation des scripts fournis (PHP et Python), à générer en utilisant le Makefile.

Vous pouvez installer la bibliothèque et le script Python en tapant la commande make install. Les tests unitaires se lancent avec la commande make tests, qui nécessite le composant Python « nose » (disponible sur PyPI et très probablement dans votre distribution). La documentation des scripts fournis peut être générée par la commande make doc. La génération est réalisée par PHPdoc pour PHP et par Sphinx pour Python, et sera disponible en HTML dans les dossiers doc-php et doc-py.

UTILISATION BASIQUE

Utilisons maintenant les scripts fournis pour convertir une page web en document ODT. La syntaxe est décrite par l'option « -h ». L'exemple d'utilisation sera présenté avec le script Python, qui est fonctionnellement le plus avancé aujourd'hui, mais presque tout est réalisable de la même façon avec le script PHP.

Le script Python utilise quelques bibliothèques externes qu'il faut donc avoir installées :

- python-tidy (uTidylib) pour s'assurer que le HTML est valide ;
- python-lxml pour la manipulation XML ;
- python-imaging (PIL) pour la manipulation des images.

Une fois ces bibliothèques installées, le script fonctionnera sans erreur, ce que l'on peut valider en affichant les options disponibles grâce à l'argument « -h ». Les principales options sont les suivantes :

- -i FICHIER : lire la page HTML depuis le fichier indiqué
- -o FICHIER : écrire le document ODT dans le fichier indiqué
- -t FICHIER : utiliser le document ODT modèle indiqué
- -u URL : l'URL dont provient la page, pour générer les bons liens hypertexte

Profitez-en pour lire la documentation associée aux autres options, vous en trouverez peut-être une indispensable pour votre cas d'utilisation. Faisons un essai, lancez les commandes suivantes :

```
$ wget -O wp-odt.html http://fr.wikipedia.org/wiki/OpenDocument
$ ./xhtml2odt.py -i wp-odt.html -o wp-odt.odt -t template.odt
-u http://fr.wikipedia.org/wiki/OpenDocument
```

La conversion produit un document wp-odt.odt que vous pouvez ouvrir avec OpenOffice. Le document contient bien l'article de Wikipedia, mais en naviguant vers la fin du document on constate que les liens du menu ont eux-aussi été exportés. Pour éviter cela, on peut demander au script de n'exporter que la partie de la page web qui se trouve dans la balise dont l'ID est « content ». Ajoutez l'option –html-id content à la ligne de commande, et relancez le script. Et voilà, seul le contenu de l'article est exporté !

Essayons avec une autre page, contenant une image cette fois :

```
$ wget -O standblog.html http://standblog.org/blog/post/2010/
06/01/iPad-au-dela-de-l-esthetique
```

```
$ ./xhtml2odt.py -i standblog.html -o standblog.odt -t template
.odt --html-id content -u http://standblog.org/blog/post/2010
/06/01/iPad-au-dela-de-l-esthetique
```

Visualisons maintenant le document avec OpenOffice : l'image est bien intégrée. Attention toutefois, l'objectif de XHTML2ODT n'est pas d'être un interpréteur généraliste de toutes les pages web d'Internet, comme le serait un navigateur web. Beaucoup de pages ne fonctionneront pas parfaitement, mais c'est normal : le but est plutôt de permettre l'écriture d'un plugin d'export ODT pour n'importe quelle application web, cas dans lequel le HTML produit est relativement maîtrisé.

ANALYSE DU PLUGIN DOTCLEAR (PHP)

Dotclear est un moteur de blog en PHP assez populaire, qui propose une architecture et une API pour les extensions (plugins).

Dans Dotclear, l'auteur peut rédiger son billet dans deux modes : syntaxe wiki ou HTML. La « syntaxe wiki » permet de mettre en forme son texte grâce à des caractères de formatage spéciaux, qui sont interprétés par le moteur lors de la sauvegarde pour générer du HTML, alors stocké en base de données, puis affiché lors de la consultation du billet. Le mode de rédaction HTML, lui, stocke directement le texte en base de données. Le plugin d'export ODT Dotclear s'enregistre auprès du moteur comme gestionnaire des URL commençant par « /odt/ », et va utiliser le reste de l'URL pour déterminer quel est le billet qu'il faut convertir.

Suivant le type d'article (billet, page, ou page d'accueil), le plugin va sélectionner le fichier modèle ODT approprié. Dans ce fichier ODT, on aura placé des balises utilisées pour la création des modèles de page Dotclear, par exemple « {{tpl:EntryContent}} ». De cette façon, on peut réutiliser une grande partie du code du moteur de Dotclear, et l'utilisateur n'a pas un autre langage de modèle à apprendre. Ces balises peuvent être placées dans le fichier à l'aide d'un traitement de texte classique compatible ODT, par exemple OpenOffice.

Après avoir dézippé le document ODT dans un répertoire temporaire, on donne le fichier content.xml à une version très légèrement modifiée du moteur de templates de Dotclear, qui va remplacer les balises par les champs correspondants du billet depuis la base de données (par exemple le titre, le contenu, la catégorie, les tags, …). Le document produit est donc un mélange d'ODT XML (provenant du modèle) et de HTML (provenant de Dotclear). Le plugin réalise alors l'import des éléments externes, c'est-à-dire des images. Si des balises d'images sont présentes dans le texte, le plugin va télécharger les images manquantes et les insérer dans le document ODT, en remplaçant au passage l'URL absolue par l'URL relative au contenu du document. Si l'URL de l'image était en fait locale au serveur hébergeant le blog Dotclear, le fichier est simplement copié.

Ensuite, le document est fourni à XHTML2ODT, qui va ignorer les balises ODT et convertir les balises HTML en ODT, produisant donc un fichier content.xml exclusivement en ODT XML. Si une image a été téléchargée avec succès, il génèrera un lien interne au document ODT ; dans le cas contraire il génèrera un lien distant, ce qui causera le téléchargement de l'image à l'ouverture du document dans le traitement de texte.

Le plugin fait ensuite une passe sur tout le texte pour détecter quels styles ont été utilisés, et quels styles sont fournis dans le document modèle d'origine. Si de nouveaux styles sont introduits par la conversion de l'article en ODT, il les ajoute à la liste des styles du document en se servant dans les styles fournis par XHTML2ODT. Enfin, les

fichiers contenus dans le répertoire temporaire sont re-zippés, et le document ODT final est envoyé au navigateur avec les en-têtes qui provoqueront un téléchargement plutôt qu'un affichage en ligne. En plus de sa fonction principale d'export, le plugin apporte une intégration relativement avancée au moteur de blog grâce à une page d'administration, un widget, et des boutons d'export. Le plugin d'export ODT de Dotclear est un bon exemple d'utilisation d'XHTML2ODT dans le cadre d'une application web en PHP utilisant des templates. Il ne faut donc pas hésiter à l'utiliser comme exemple pour un développement sur des technologies similaires.

ANALYSE DU PLUGIN TRAC (PYTHON)

Trac est un gestionnaire de projets écrit en Python couvrant les besoins classiques d'un projet de développement logiciel, c'est-à-dire un wiki, une interface au gestionnaire de versions, et un gestionnaire de tickets. Il est très extensible, et son API comprend justement l'export des pages du wiki dans un autre format. La rédaction dans le wiki de Trac se fait elle aussi dans une syntaxe texte simplifiée, que le moteur convertit en HTML avant de stocker en base de données. Le plugin Trac va donc suivre à peu près le même cheminement que le plugin Dotclear. Tout d'abord, le fichier ODT modèle est dézippé dans un répertoire temporaire. À la différence du plugin Dotclear, il ne contient pas de balises de formatage, puisque cette fonctionnalité n'existe pas dans Trac. En revanche, il contient un mot-clé paramétrable (par défaut : TRAC-ODT-INSERT) qui sera remplacé par le contenu de la page wiki. En cas d'absence de ce mot-clé, le contenu de la page est tout simplement ajouté à la fin du document. Le reste des opérations est très similaire à celles du plugin Dotclear :

1. les images distantes sont téléchargées et les images locales sont copiées depuis le répertoire des attachements de Trac ;
2. le contenu de la page wiki est converti en ODT grâce à XHTML2ODT ;
3. les styles manquants sont ajoutés ;
4. enfin, le répertoire temporaire est rezippé pour former le document ODT qui sera envoyé au navigateur.

La grande similarité avec le plugin Dotclear est la preuve de la factorisation maximale des opérations de conversion dans les feuilles de style XSL. Le plugin Trac est donc un bon exemple d'utilisation d'XHTML2ODT dans une application web en Python n'utilisant pas de templates. Là aussi, pour un développement dans des technologies similaires, il ne faut pas hésiter à s'en inspirer.

L'AVENIR D'XHTML2ODT

La bibliothèque servant à convertir d'un format vers un autre, son avenir est naturellement lié à celui des deux formats. Il n'y a a priori pas de révolution du format OpenDocument à attendre dans les prochaines années. Le HTML, en revanche, est en pleine ébullition en ce moment avec l'arrivée du HTML5 et de ses nouvelles balises. XHTML2ODT devra donc s'adapter pour les prendre en compte le plus fidèlement possible. Parmi celles-ci, certaines seront totalement impossibles à intégrer (<canvas>), certaines seront relativement complexes (<audio> et <video>), d'autres ne devront pas être converties (<section>, <article>, …) et le reste nécessitera des adaptations diverses. Le travail d'intégration de ces nouvelles balises a bien commencé dans l'espace de développement, mais se trouve actuellement bloqué par l'utilisation de Tidy, le module de « nettoyage » du HTML que les plugins utilisent en amont de la conversion pour s'assurer que le HTML à traiter est bien valide. Sans Tidy, la conversion XSLT échouerait en essayant de lire du HTML mal formé (au sens XML du terme), et malheureusement ce type de code est très courant sur le web. Aujourd'hui Tidy ne connaît pas les nouvelles balises de HTML5, et cherche donc à les supprimer lors de son exécution. Il existe des moyens de contournement, mais ils ont un impact important sur les plugins utilisant XHTML2ODT. En attendant que Tidy prenne en compte les nouveautés d'HTML5, l'adaptation d'XHTML2ODT continue dans la branche de développement dédiée.
Au-delà des évolutions du HTML, les évolutions possibles pour la bibliothèque elle-même sont les suivantes :

- la prise en compte des quelques éléments du HTML qui ne sont pas encore gérés (cellules de tableau fusionnées, éléments de formulaires, voire même la balise <font>, …) ;
- l'utilisation par de nouveaux plugins d'export ODT, par exemple pour WordPress, Drupal, Joomla, etc. ;
- la correction des bugs éventuels.

Et cette liste n'est évidemment pas exhaustive. Toutes les idées d'améliorations sont envisageables, les propositions peuvent être faites en ouvrant des tickets sur le site du projet.

Pour en savoir plus

Il est intéressant de noter que la fondation OpenDocument Fellowship a proposé un prix de 11 500 dollars à qui réaliserait un export parfait de HTML vers ODT. Je n'ai pas les compétences pour participer à ce concours, mais si quelqu'un veut s'appuyer sur XHTML2ODT pour le réaliser, qu'il n'hésite pas ! C'est en tout cas la preuve que le besoin existe bel et bien, et il n'existe pas à ma connaissance d'autre projet de conversion générique d'HTML vers ODT. J'espère que ce projet vous permettra d'intégrer facilement un export ODT dans votre application web. Si vous êtes amené à l'utiliser, n'hésitez pas à me contacter pour toute demande d'assistance.

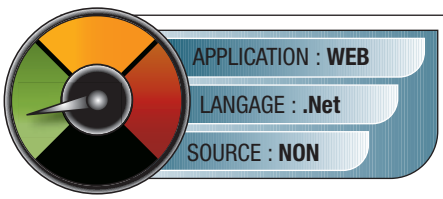
Liens

- Le projet XHTML2ODT : http://xhtml2odt.org
- Le format OpenDocument : http://fr.wikipedia.org/wiki/OpenDocument
- Le plugin Dotclear : http://lab.dotclear.org/wiki/plugin/odt/fr
- Le plugin Trac :http://trac-hacks.org/wiki/OdtExportPlugin
- Le prix de la fondation OpenDocument Fellowship : http://opendocumentfellowship.com/development/projects/html2odf

Aurélien **Bompard**

Sketchflow : réaliser des prototypes rapides !

L'outil SketchFlow a été montré pour la première fois lors du MIX 2009 à Las Vegas. Intégré à l'origine avec Expression Blend 3, il permet de réaliser des prototypes très rapidement. Depuis peu la version 4 de la gamme Expression est disponible, mais SketchFlow est uniquement associé à Blend dans sa version Ultimate.



QU'EST-CE QU'UN PROTOTYPE ?

Un prototype permet de valider la navigation et les fonctionnalités dans le cadre de la définition d'une future application. Considéré comme un complément à des spécifications fonctionnelles, il permet un échange interactif entre le client et l'équipe de développement afin de lever les zones d'ombre qui peuvent subsister dans les détails de la conception de l'application.Un prototype peut se présenter sous plusieurs formes, les plus courantes sont :

- Post-It
- Tableau blanc
- Papier, crayon et gomme
- Word / Excel / Powerpoint / Visio
- Photoshop
- Applications / Maquettes (HTML / Winforms ...)

Leurs limites sont nombreuses, difficile de bien se comprendre sur une navigation lorsque les pages sont simplement listées sur un tableau blanc, ou encore de déterminer l'apparition d'une boîte de dialogue d'authentification avec un papier et un crayon par exemple. Attention, un prototype ne peut pas servir de base pour le développement d'une application, il ne faut pas le confondre avec une première version !

COMMENT S'UTILISE SKETCHFLOW ?

SketchFlow s'utilise aujourd'hui grâce à l'outil de la gamme Expression : Expression Blend 4. Pour cela, un nouveau projet de type SketchFlow doit être créé. Deux choix sont possibles pour sa création : un projet SketchFlow en Silverlight ou un projet SketchFlow en WPF (Windows Presentation Foundation). Le choix peut s'effectuer sur la possibilité d'accéder au player pour le client (le player est le conteneur qui va permettre de visualiser le prototype) ou par rapport à la technologie qui sera utilisée au final pour le développement. Le player en Silverlight est très facile à déployer sur un serveur Web pour un accès à distance plus simple et le player WPF peut, quant à lui, être utilisé en local sur une machine ou déployé grâce à un serveur ClickOnce par exemple, comme une application WPF normale [Fig.1].

SketchFlow est un nouveau type de projet dans Expression Blend et apporte avec lui plusieurs fenêtres supplémentaires comme la SketchFlow Map, SketchFlow Animations et SketchFlow Feedbacks. Voici l'interface de SketchFlow dans Expression Blend 4 : [Fig.2].

La SketchFlow Map permet de voir les différentes pages qui composent le prototype, ainsi que les différents éléments (Component Screen, CustomControls, etc.) qui la composent et qui peuvent être réutilisés dans les autres écrans.

Dans notre exemple ci-dessous, on peut voir les différents écrans en bleu et les Component Screen en vert, comme le Menu que l'on retrouve dans la plupart des écrans. Les flèches bleues représentent la navigation entre le Menu et les pages appelées. Les flèches vertes, quant à elles, indiquent les écrans contenant le Menu. L'ampoule se trouvant sur l'écran Home permet de savoir qu'une remarque (feedback) a été faite pour cet écran, que l'on pourra consulter dans la fenêtre SketchFlow Feedbacks [Fig.3].

Tout comme avec Expression Blend 4 pour une application Silverlight ou WPF, il est possible d'utiliser des états, des transitions, des ani-

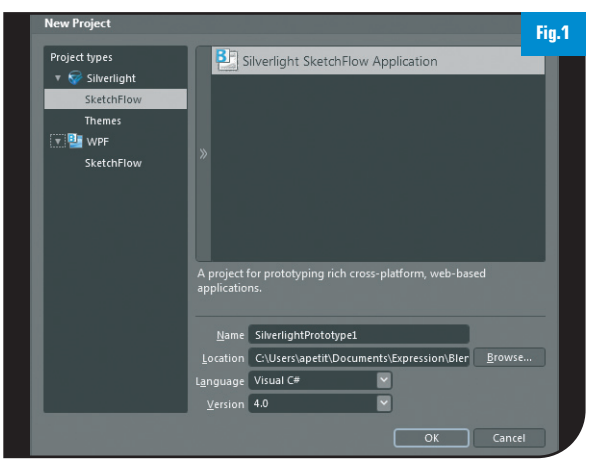


Fig.1

Création d'un projet SketchFlow avec Expression Blend 4

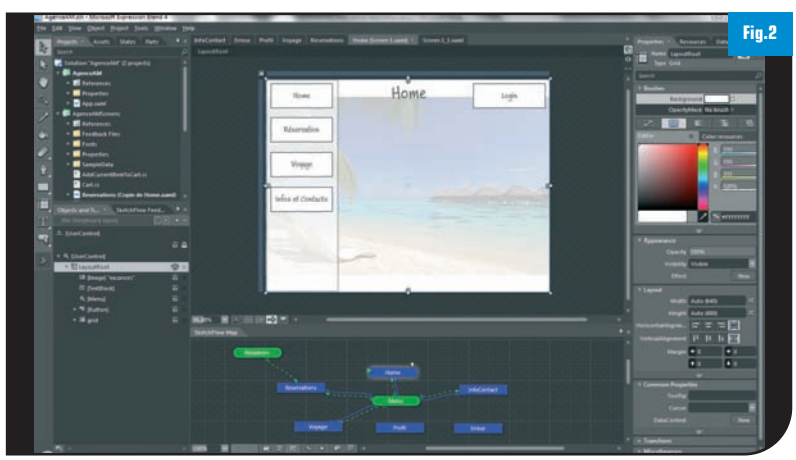


Fig.2

Interface d'Expression Blend 4 avec un projet SketchFlow

mations et des story-boards afin de rendre les pages plus interactives, et simuler simplement les fonctionnalités de l'application, par exemple l'authentification d'un utilisateur [Fig.4].
Afin de simuler au mieux le contenu de l'application, il est possible de créer des jeux de données fictives qui vont permettre de remplir les interfaces et d'avoir un aperçu plus réaliste [Fig.5].
Voici un exemple d'utilisation des jeux de données avec une interface de type maître-détails. Grâce à un simple glisser-déposer depuis la structure de données vers le designer, il est ensuite possible d'organiser ces données comme on le souhaite, les lier entre elles et de laisser apparentes seulement celles qui sont pertinentes dans l'interface [Fig.6].
La figure 7 montre le Player pour une application Silverlight. Cette dernière s'exécute dans un navigateur Internet. L'utilisateur peut naviguer de page en page soit en cliquant sur l'application soit en utilisant la Map. De plus, le client peut annoter l'application, faire des remarques, effectuer des tracés et exporter tout cela sous la forme d'un fichier Feedback (avec une extension .feedback). Celui-ci sera ensuite transmis à l'équipe de développement afin de l'importer dans Expression Blend, de visualiser toutes les annotations et d'apporter toutes les modifications nécessaires au prototype. Dans la fenêtre SketchFlow Map d'Expression Blend, il sera possible de trouver l'emplacement de ces Feedbacks grâce à une ampoule se trouvant sur les écrans concernés [Fig.7].
Un style SketchFlow est appliqué à tous les contrôles afin de toujours garder à l'esprit qu'il s'agit d'un prototype et non de l'application finale. Ce style se veut proche d'un style dessiné à la main afin de renforcer encore plus ce côté papier et crayon du prototype. Voici un condensé des contrôles que l'on a l'habitude de rencontrer dans des applications mais auxquels le style SketchFlow a été appliqué [Fig.8].
Il est à noter qu'une application SketchFlow reste avant tout une application WPF ou Silverlight, et il est aussi possible d'ouvrir cette application dans Visual Studio 2010 et modifier le code C# ou VB afin d'ajouter des comportements supplémentaires au prototype.

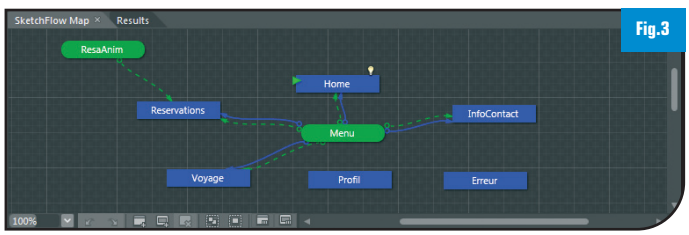


Fig.3

SketchFlow Map

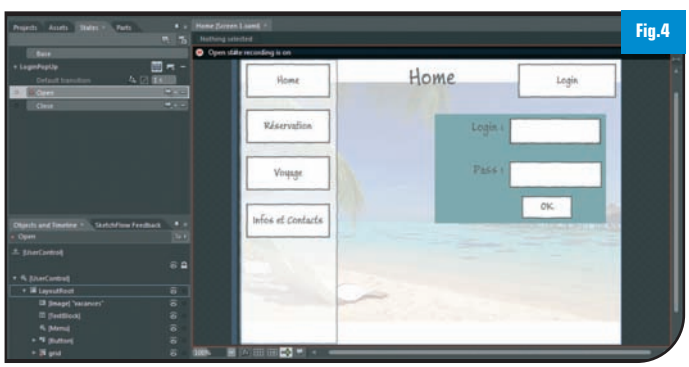


Fig.4

Différents états sur l'interface Home

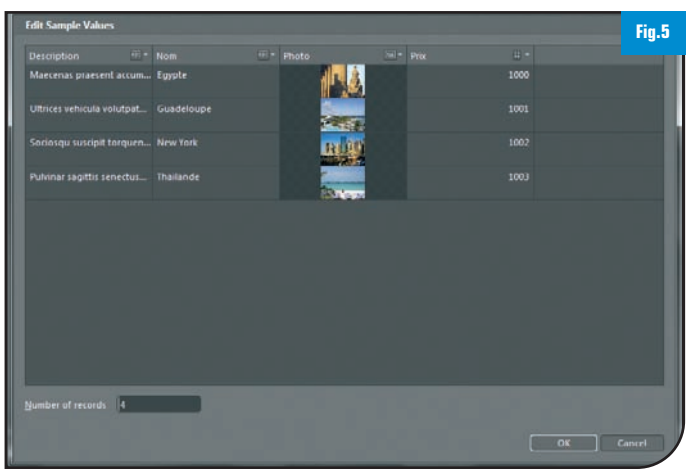


Fig.5

Jeu de données

SketchFlow et TFS

Chaque feedback du fichier .feedback peut être converti en Work Item pour Team Foundation Server (TFS), en faisant un clic-droit sur chaque item et en cliquant sur "Convert Feedback into TFS Work Item" : [Fig.9].
Il faut ensuite sélectionner le type de Work Item que l'on souhaite créer : [Fig.10].
Il est possible de compléter/éditer les informations que va contenir ce nouveau Work Item grâce à l'interface suivante : [Fig.11].
Une fois le Work Item sauvegardé, une boîte de dialogue apparaît

Fig.6

Utilisation de données dans une interface Maître-Détail

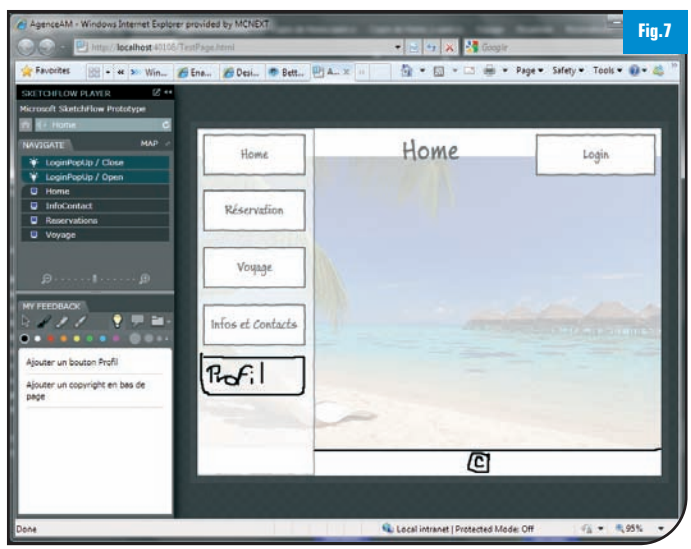


Fig.7

Player Silverlight SketchFlow

contenant le numéro d'identifiant et le lien permettant de visualiser le Work Item nouvellement créé. En regardant la liste des Work Items du projet dans TFS, on peut retrouver ce Work Item issu du Feedback de SketchFlow et l'afficher : [Fig.12].

SketchFlow et SharePoint

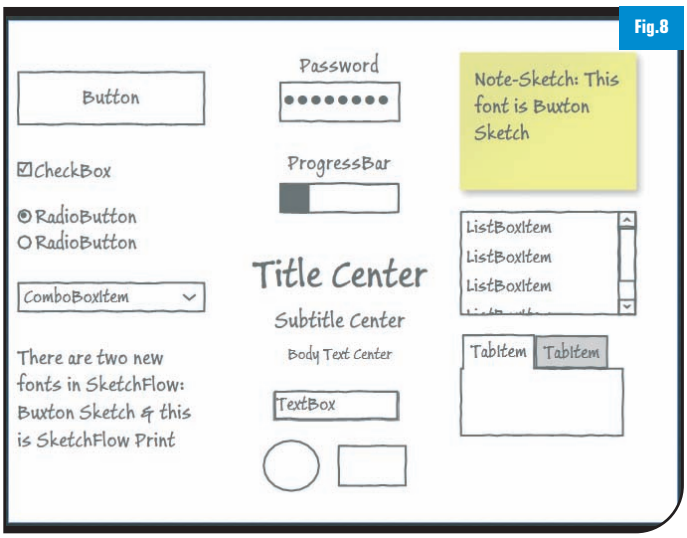
Grâce à l'utilisation des différents outils collaboratifs, il est également possible de publier son application SketchFlow dans un site SharePoint.

Pour cela il suffit de sélectionner « Publish to SharePoint » dans le menu « File » d'Expression Blend 4. Une boîte de dialogue apparaît et demande à quel endroit doit être publié le projet SketchFlow dans le portail SharePoint [Fig.13].

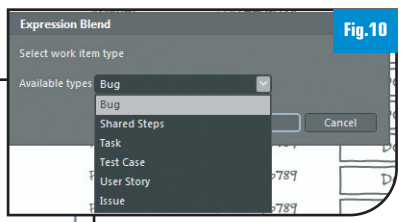
Le projet se déploie et devient accessible à partir du portail SharePoint pour les utilisateurs ayant les droits d'accès. On peut imaginer déployer un projet SketchFlow sur le portail SharePoint du projet mis en place grâce à TFS auquel le client accède par exemple [Fig.14].

SketchFlow est un nouvel outil qui permet de compléter la phase de définition d'un projet lors de la phase des spécifications fonctionnelles. Il est certain que la réalisation d'un prototype avec SketchFlow prend plus de temps qu'un prototype dessiné sur un papier, mais son interactivité, sa facilité de modification et sa capacité à se rapprocher au plus près du besoin du client, permet de définir plus précisément les fonctionnalités et d'éviter les éventuelles modifications des spécifications lors de la phase de réalisation du projet.

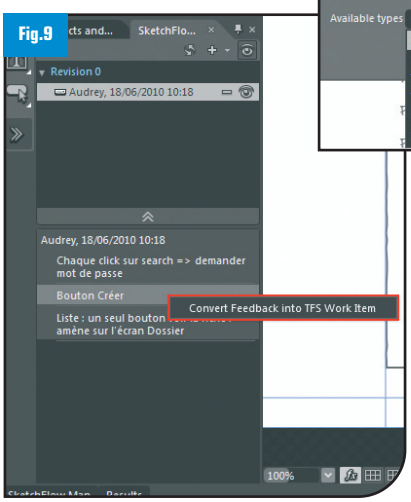
■ Audrey **Petit**
http://blogs.developpeur.org/audrey
MVP Client Application Development
Ingénieur consultante .NET chez MCNEXT



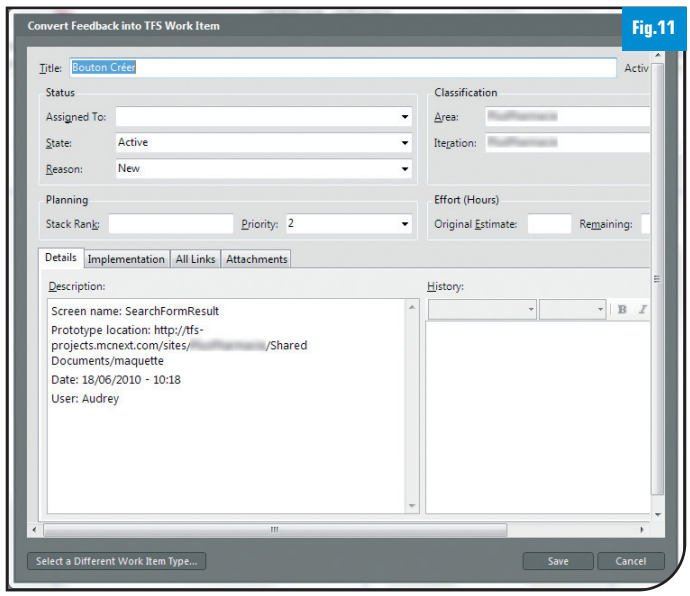
Quelques exemples de contrôles avec le style SketchFlow



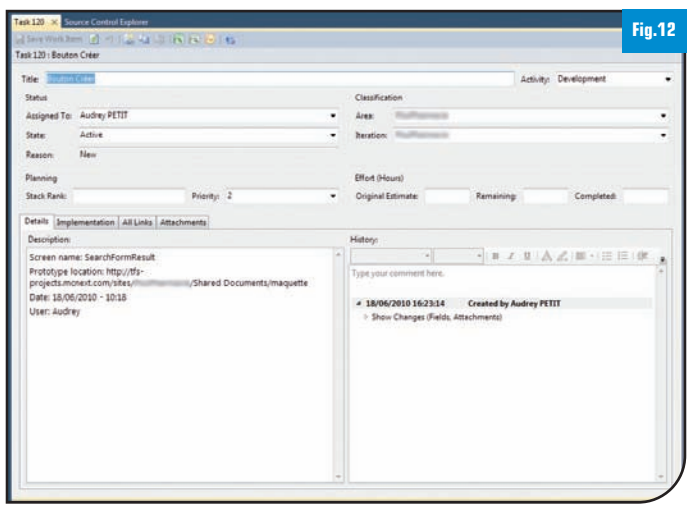
Type du Work Item à créer à partir du feedback sélectionné



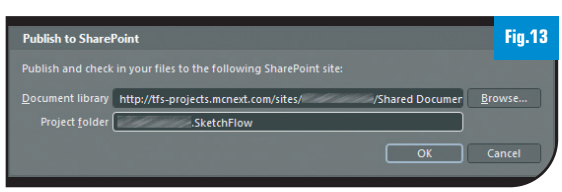
Liste des Feedbacks importés dans Expression Blend 4



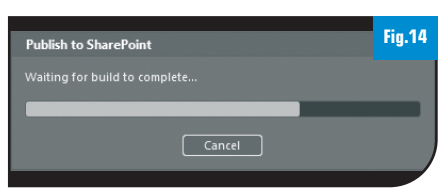
Détails du Work Item à créer à partir d'un feedback



Interface du Work Item créé



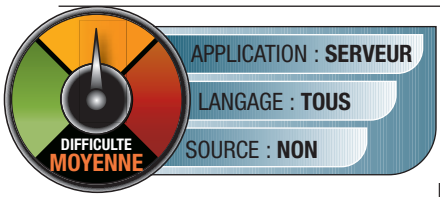
Sélection de la librairie de publication dans SharePoint



Publication de l'application SketchFlow dans SharePoint

Indexation des sources et serveur de symboles avec TeamBuild 2008

TeamBuild 2008 ou, plus précisément, Team Foundation Build 2008 est le serveur de build fourni avec Team Foundation Server (TFS). Son but est d'automatiser et de centraliser la compilation et le déploiement des solutions et projets se trouvant dans le contrôleur de source de TFS.



Avant de voir comment personnaliser son script de build nous verrons un rappel des notions abordées et des outils nécessaires. Ensuite nous verrons l'intégration avec TeamBuild et comment configurer Visual Studio 2008 et 2010 pour se servir de ce que nous allons mettre en place.

DÉFINITIONS

Fichiers de symboles

Un fichier de symboles est un fichier contenant un ensemble d'informations de débogage pour un binaire (exécutable ou assembly). Ces fichiers peuvent contenir les informations suivantes :

- Les adresses des fonctions.
- Les adresses et le nom des variables globales.
- Le nom des variables locales et des paramètres.
- La position dans la pile des variables locales et des paramètres.
- Le nom des fichiers sources et le numéro de ligne dans ces fichiers pour chaque instruction.
- …

Ces fichiers permettent lors de débogage ou de crash de fournir des informations utiles pour le développeur.

Les fichiers de symboles générés par Visual Studio ont l'extension « pdb » (program database). Par défaut Visual Studio génère pour chaque projet un fichier de symboles complet pour la configuration « Debug » qui contiendra les symboles publics et privés et un fichier plus léger pour la configuration « Release » ne contenant que les symboles publics. Afin de modifier la génération de ses fichiers il faut aller dans les propriétés du projet sur l'onglet « Build » et cliquer sur le bouton « Advanced » [Fig.1].

Plus d'informations sur les fichiers de symboles sont disponibles sur la MSDN : http://msdn.microsoft.com/en-us/library/cc266472.aspx

Indexation des sources

Comme indiqué précédemment, les fichiers de symboles contiennent le nom des fichiers sources associés au binaires. Le chemin complet des fichiers sources contenu dans les fichiers de symboles correspond à l'endroit où se trouvaient les fichiers sources lors de la compilation du binaires [Fig.2].

Cela permet à Visual Studio de proposer, lors du débogage, de voir le fichier source et de faire de l'exécution en pas à pas. Lorsque l'on débogue sur la même machine, il n'y a pas de problème. Par contre lorsque l'on débogue depuis une autre machine, par exemple si l'on utilise un binaire fourni par une autre équipe, cela peut poser des problèmes car Visual Studio ne trouvera pas forcément les fichiers sources là où il les attend. De plus, dans le cas d'une utilisation de TFS se pose aussi le problème de version du fichier.

L'indexation des sources est un mécanisme permettant d'ajouter dans les fichiers de symboles un autre chemin vers les fichiers sources et, dans le cas de TFS, d'ajouter en plus la version du fichier en se basant sur le numéro de changset. Cela permettra à Visual Studio de pouvoir récupérer les sources directement depuis TFS.

Microsoft fournit un outil permettant de faire ces modifications et nous verrons comment l'installer et le configurer.

Serveur de symboles

Un serveur de symboles n'est rien d'autre qu'un répertoire dans lequel les fichiers de symboles sont stockés dans un répertoire bien précis afin que Visual Studio puisse retrouver la bonne version du fichier lors d'un débogage. L'ajout d'un fichier se fera donc par l'intermédiaire d'un outil spécifique proposé par Microsoft comme pour l'indexation des sources.

INSTALLATION ET CONFIGURATION DES COMPOSANTS

Avant de pouvoir mettre en place l'indexation et le serveur de symboles il faut installer les outils nécessaires. Ces outils font partie des outils de débogage fournis par Microsoft et doivent être installés sur

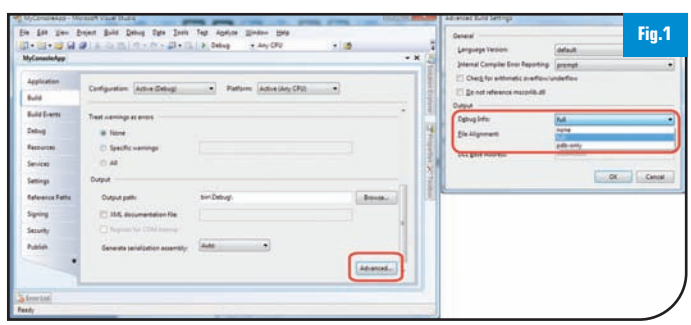


Fig.1

Configuration des symboles

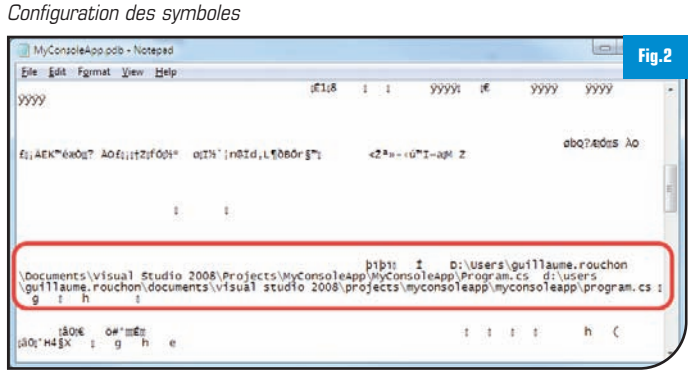


Fig.2

Emplacement des sources

le serveur de build. Si vous n'avez pas encore installé ce serveur, vous pouvez vous référer à la documentation d'installation de TFS : http://www.microsoft.com/downloads/details.aspx?FamilyID=FF12844F-398C-4FE9-8B0D-9E84181D9923&displaylang=en

Debugging tools for Windows

L'ensemble des outils nécessaires se trouvent dans les « Debugging tools for Windows » fournis par Microsoft. La première chose à faire sera donc de les télécharger à l'adresse suivante :
http://www.microsoft.com/whdc/devtools/debugging/installx86.mspx#a
Une fois téléchargé, lancez l'installeur et, dans la liste des composants, sélectionnez « Tools » et « Source Server » [Fig.3].
Les « Tools » contiennent les outils et scripts nécessaires à l'indexation des sources et le « Source Server » contient les outils nécessaires pour le serveur de source.
Une fois les outils installés il faut configurer l'indexation des sources afin de prendre en compte le serveur TFS et le fait que les binaires seront compilés par TeamBuild. Allez dans le répertoire « srcsrv » se trouvant dans le répertoire d'installation (par défaut « C:\Program Files\Debugging Tools for Windows (x86) ») puis :

- Modifier le fichier « srcsrv.ini » afin de configurer « MYSERVER » avec l'URL du serveur TFS (attention au slash final qui doit être présent). [Fig.4]
- Par défaut le script d'indexation va chercher les fichiers de symboles dans le répertoire ou se trouvent les sources. Dans le cas de TeamBuild, ces fichiers sont dans deux répertoires distincts (« Sources » et « Binaries »), il faut donc modifier le fichier « tfsindex.cmd » afin d'ajouter en paramètre le répertoire ou se trouvent les symboles et de passer cette valeur au script Perl appelé. Pour cela il faut ajouter le paramètre « -SYMBOLS=%1 ». [Fig.5]

Le script permettant l'indexation des sources est en Perl, il faut donc aussi installer ActivePerl sur le serveur de build.

ActivePerl

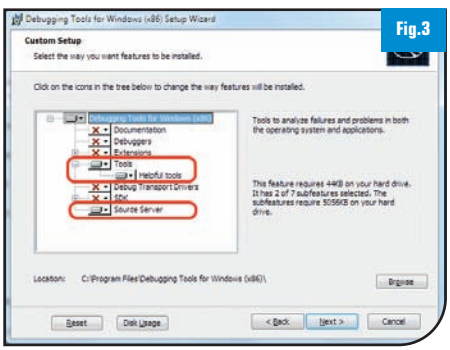
Téléchargez la dernière version de ActivePerl à l'adresse suivante :
http://www.activestate.com/activeperl/
Une fois téléchargée, lancez l'installeur et, dans la liste des composants, sélectionnez uniquement le runtime [Fig.6]. N'oubliez pas de sélectionner l'option pour mettre à jour la variable d'environnement « Path » afin d'ajouter Perl [Fig.7]. Il ne reste plus qu'à configurer le serveur de build afin de pouvoir intégrer l'indexation des sources et le serveur de symboles dans les scripts TeamBuild.

Configuration du serveur de build

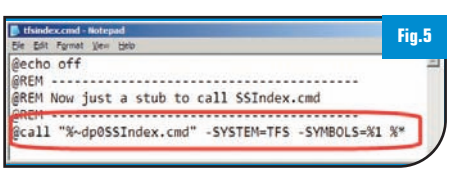
La première des choses à faire est d'ajouter une nouvelle variable d'environnement contenant le répertoire d'installation des outils de débogage installé précédemment. Cela simplifiera par la suite l'écriture des scripts :

- Allez dans les propriétés du poste de travail dans l'onglet « Advanced ».
- Cliquez sur « Environment Variables ».
- Dans la zone « System variables » cliquez sur « New » et ajouter une variable avec le nom « DBGTOOLSROOT » et pour valeur le répertoire dans lequel vous avez installé les outils de débogage (sans oublier d'ajouter un back slash à la fin) [Fig.8].

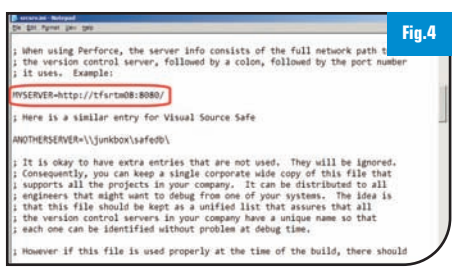
Afin que le service TeamBuild puisse prendre en compte cette nouvelle variable, il faudra le redémarrer.
Il faut maintenant créer un répertoire qui servira de répertoire de stockage pour le serveur de symboles et créer un partage réseau sur ce répertoire. Par souci de simplicité, nous allons créer ce répertoire sur le serveur de build mais vous pouvez le créer sur n'importe quel serveur :
Créez un répertoire « Symbols » (ou tout autre nom).
Vérifiez que le compte sous lequel tourne le service de Build à bien tous les droits sur le répertoire et que les développeurs ont le droit de lecture.
Ajoutez un partage réseau sur ce répertoire et vérifiez que le compte du service de build à bien tous les droits sur le partage et que les développeurs ont le droit de lecture.



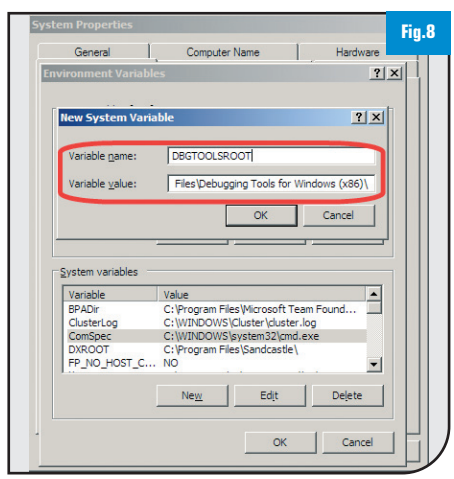
Installation des Debugging Tools



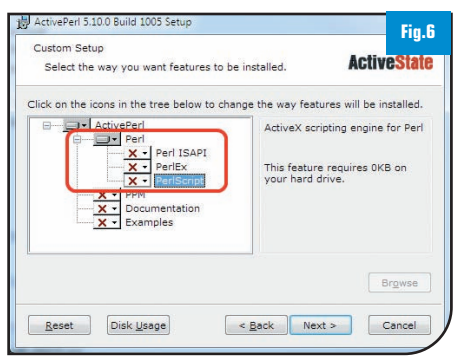
Fichier tfsindex.cmd



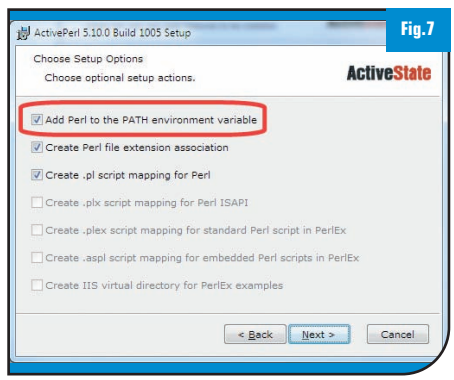
Fichier srcsrv.ini



Variable DBGTOOLSROOT



Installation d'Active Perl 1/2



Installation d'Active Perl 2/2

Pour résumer, le répertoire qui stockera les symboles doit avoir les mêmes droits que le répertoire stockant le résultat des build.

CONFIGURATION DU SCRIPT TEAMBUILD

Maintenant que le serveur est configuré, nous allons voir comment modifier un script de build (fichier TFSBuild.proj) afin d'ajouter l'indexation des sources et à sa publication sur le serveur de symboles. Nous verrons ensuite comment faire un composant réutilisable afin d'éviter de nombreux copier/coller entre différents scripts.

Indexation des fichiers de symboles

Pour indexer les symboles, il suffit d'appeler la commande « tfsindex.cmd ». Pour cela, nous utiliserons la tâche MSBuild « Exec » en la configurant de manière adéquate :

```
<Target Name=»IndexSources»>
  <Message Text=»Indexing sources at $(SolutionRoot)»/>
  <Exec
   Command=»&quot;$(DBGTOOLSROOT)srcsrv\tfsindex.com&quot;  &
quot;$(BinariesRoot)\Release\&quot;»
   WorkingDirectory=»$(SolutionRoot)»/>
</Target>
```

Nous avons créé une nouvelle target qui appellera la commande voulue en lui passant en paramètre le chemin ou se trouvent les fichiers de symboles et en spécifiant comme répertoire de travail le répertoire ou se trouvent les sources. Afin de ne pas avoir de problème si les chemins passés ont des espaces nous avons entouré ces valeurs avec des guillemets au format XML « " ».
Il ne reste plus qu'à ajouter l'appel à notre target. Les fichiers de symboles étant générés à la compilation nous allons modifier la target d'extension « AfterCompile » afin de lui ajouter une dépendance :

```
<Target Name=»AfterCompile» DependsOnTargets=»IndexSources»/>
```

Serveur de symboles

Comme pour l'indexation de sources, nous allons utiliser la tâche MSBuild « Exec » afin d'appeler la commande « symstore » pour publier nos fichier de symboles :

```
<Target Name=»PublishSymbols»>
  <Message Text=»Publishing symbols from $(BinariesRoot)»/>
  <Exec
   Command=»&quot;$(DBGTOOLSROOT)symstore&quot; add /f &quot;
$(BinariesRoot)\Release\*.pdb&quot; /s &quot;\\tfsrtm08\Symbols
&quot; /t &quot;$(BuildDefinitionName)&quot; /v &quot;$(Build
Number)&quot;»/>
</Target>
```

La commande « symstore » prend un ensemble de paramètre décrits dans le tableau ci-dessous :

Paramètre	Description
add /f &quot ;$	Indique que l'on veut ajouter des fichiers au serveur.
(BinariesRoot)\Release*.pdb"	Spécifie les fichiers à ajouter. Nous ajoutons tous les fichiers se trouvant dans le répertoire Release des binaires compilés.
/s &quot ;\\tfsrtm08\Symbols"	Spécifie le chemin du serveur de symboles. Il s'agit du répertoire partagé créé plus haut.
/t &quot ;$ (BuildDefinitionName)&quot ;	Spécifie le nom du produit. Dans l'exemple nous fournissons le nom du build.
/v &quot ;$(BuildNumber)&quot ;	Spécifie la version du produit. Dans l'exemple nous utilisons le numéro de build.

Comme pour l'indexation il faut ajouter le code pour appeler notre target. Cet appel devant être fait après l'indexation, le plus simple est d'ajouter notre target à la liste des dépendances de la target « AfterCompile » :

```
<Target  Name=»AfterCompile»  DependsOnTargets=»IndexSources;
PublishSymbols»/>
```

Composant réutilisable (.target)

Le code présenté ci-dessus marchera dans tous les cas et vous pouvez le personnaliser pour, par exemple :
- Ne pas prendre tous les fichiers de symboles.
- Gérer les fichiers de plusieurs configurations.
- Modifier le nom du produit et/ou la version.

Le problème est que si vous avez beaucoup de script TeamBuild, cela devient vite fastidieux à gérer dans chaque script. Heureusement il est possible avec MSBuild de déplacer des bouts de script dans des fichiers séparés afin de les partager (ces fichiers ont généralement comme extension .target). Nous allons donc voir comment écrire un tel fichier et lui ajouter un ensemble de propriétés afin de le rendre configurable facilement.

Paramétrage

Nous allons ajouter des paramètres pour :
- Exécuter ou non l'indexation des sources.
- Exécuter ou non l'ajout au serveur de symboles.
- Exécuter ou non la suppression des symboles après la publication.
- Spécifier les répertoires contenant les fichiers de symboles.
- Pour chaque répertoire le nom du projet et la version (pour le serveur de symboles).
- Le répertoire du serveur de symboles.

Pour chacun de ces paramètres, une valeur par défaut devra être fournie. Nous allons donc ajouter à notre fichier « .target » les propriétés et items suivantes (les sauts de lignes sont présents pour rendre lisible le code) :

```
<Project xmlns=»http://schemas.microsoft.com/developer/msbuild/2003» >
  <PropertyGroup>
    <SkipIndexSources Condition=» ‘$(SkipIndexSources)’==’’ «>
      false
    </SkipIndexSources>
    <SkipPublishSymbols Condition=» ‘$(SkipPublishSymbols)’==’’ «>
      false
    </SkipPublishSymbols>
    <SkipCleanSymbols Condition=» ‘$(SkipCleanSymbols)’==’’ «>
      false
    </SkipCleanSymbols>
    <SymbolServerPath Condition=» ‘$(SymbolServerPath)’==’’ «>
      $(DropLocation)\Symbols
    </SymbolServerPath>
  </PropertyGroup>

  <Target Name=»GetSymbolFiles»>

    <ItemGroup>
      <SymbolFiles Condition=» ‘@(SymbolFiles)’==’’ « Include=»
$(BinariesRoot)\**\*.pdb»>
        <Product>$(BuildDefinitionName)</Product>
```

```
        <Version>$(BuildNumber)</Version>
      </SymbolFiles>
    </ItemGroup>

  </Target>
</Project>
```

Par défaut, toutes les étapes du script seront exécutées et le répertoire pour le serveur de symboles est le répertoire « Symbols » sous le répertoire de drop de binaires. Le création de l'item « SymbolFiles » est dans une target afin qu'il soit créé dynamiquement après la compilation de la solution (sinon il n'y aurait eu aucun fichier).

Il ne reste plus qu'à ajouter des targets pour les différentes phases :
Indexation des sources.
• Publication des symboles.
• Suppression des symboles dans les répertoires sources.
• Indexation des fichiers de symboles
Une target aura pour rôle d'indexer les sources. Cette target ne devra s'exécuter que si la propriété « SkipIndexSources » est différente de « true » et qu'il y a des fichiers de symboles :

```
<Target
  Name=»CoreIndexSources»
  Condition=» '$(SkipIndexSources)'!='true' and '$(IsDesktop
Build)'!='true' and '@(SymbolFiles)'!='' «>

  <Message Text=»Indexing sources in directory '%(SymbolFiles.
RootDir)%(SymbolFiles.Directory)'.»/>

  <Exec
    Command=»&quot;$(DBGTOOLSROOT)srcsrv\tfsindex.cmd&quot; &
quot;%(SymbolFiles.RootDir)%(SymbolFiles.Directory)\&quot;»
    WorkingDirectory=»$(SolutionRoot)»/>

</Target>
```

Publication des symboles

La target pour la publication des symboles ne devra s'exécuter que si « SkipPublishSymbols » est différent de « true » et qu'il y a des fichiers de symboles :

```
<Target
  Name=»CorePublishSymbols»
  Condition=» '$(SkipPublishSymbols)'!='true' and '$(IsDesktop
Build)'!='true' and '@(SymbolFiles)'!='' «>

  <Message Text=»Publishing symbols in directory '%(SymbolFiles.
RootDir)%(SymbolFiles.Directory)'.»/>

  <Exec
    Command=»&quot;$(DBGTOOLSROOT)symstore&quot; add /f &quot;%
(SymbolFiles.RootDir)%(SymbolFiles.Directory)*.pdb&quot; /s &quot;$
(SymbolServerPath)&quot; /t &quot;%(SymbolFiles.Product)&quot; /v
&quot;%(SymbolFiles.Version)
&quot;»/>

</Target>
```

Suppression des symboles

Afin de ne pas fournir des fichiers de symboles avec les binaires, une target de suppression est ajoutée et comme pour les autres pourra ne pas être exécutée :

```
<Target
  Name=»CoreCleanSymbols»
  Condition=» '$(SkipCleanSymbols)'!='true' and '$(IsDesktop
Build)'!='true' and '@(SymbolFiles)'!='' «>

  <Delete Files=»@(SymbolFiles)»/>

</Target>
```

Assembler le tout

Pour finir une target principale permettra d'appeler dans l'ordre les targets définies précédemment :

```
<PropertyGroup>
  <IndexSourceAndPublishSymbolsDependsOn>
    GetSymbolFiles;
    CoreIndexSources;
    CorePublishSymbols;
    CoreCleanSymbols
  </IndexSourceAndPublishSymbolsDependsOn>
</PropertyGroup>

<Target
      Name=»IndexSourceAndPublishSymbols»
      DependsOnTargets=»$(IndexSourceAndPublishSymbolsDependsOn)»/>
```

Afin de permettre d'étendre la chaîne de traitement, on utilise une propriété afin de définir la liste des dépendances de la target principale « IndexSourceAndPublishSymbols ».
Le fichier « Qetza.TeamBuild.IndexSourcesAndPublishSymbols.targets » joint à cet article contient l'ensemble du script décrit ci-dessus avec en plus :
• Des tâches pour ajouter des étapes dans le log du build (BuildStep).
• Une target de validation des paramètres.
• Un ensemble de point d'extension (BeforeIndexSources, AfterIndexSources, …).

Utilisation du script réutilisable

Maintenant que nous avons un script réutilisable, il faut mettre à jour notre script de build afin de l'utiliser. Dans un premier temps il faut copier ce fichier « .target » sur le serveur de build. Le plus simple est de le mettre dans un sous répertoire de « C:\Program Files\MSBuild ». Dans votre script de build il faut ajouter une ligne afin d'importer notre fichier :

```
<Import Project=»$(MSBuildExtensionsPath)\Custom\IndexSources
AndPublishSymbols.targets»/>
```

Il faut ensuite appeler notre target principale « IndexSourceAndPublishSymbols » via une dépendance sur la target « AfterCompile » :

```
<Target Name=»AfterCompile» DependsOnTargets=»IndexSourceAnd
PublishSymbols» />
```

Enfin n'oubliez pas de personnaliser, si besoin, les propriétés et si vous voulez personnaliser les fichiers de symboles pris en compte il faudra surcharger la target « GetSymbolFiles » afin de créer un item « SymboleFiles » contenant les fichiers voulus (attention à ne pas oublier les métadonnées Product et Version).
Une fois le script mis en place, il ne reste plus qu'à configurer les postes clients afin d'utiliser ce que nous venons de mettre en place.

CONFIGURATION DE VISUAL STUDIO 2008 ET 2010

La configuration de Visual Studio se fait en deux étapes :

- Définir le chemin ou se trouvent les symboles.
- Activer le serveur de source afin que Visual Studio interroge TFS pour récupérer la bonne version des sources.

Pour le chemin des fichiers de symboles, dans les options de Visual Studio (Menu « Tools/Options ») il faut aller sur le nœud « Debugging\Symbols » et ajouter le répertoire partagé contenant les symboles [Fig.9].
Visual Studio ira maintenant chercher les fichiers « .pdb » dans le répertoire « \\TFSRTM08\Symbols » en plus du répertoire local.
Les fichiers étant réindexés pour utiliser TFS, il faut activer l'option dans Visual Studio pour qu'il interroge le serveur TFS. Cela se passe toujours dans les options, dans le nœud « Debugging\General » ou il faut activer le serveur de sources [Fig.10].
Visual Studio est maintenant configuré pour utiliser tout ce que nous avons mis en place. Il reste une dernière amélioration à apporter. En effet dans l'état actuel, à chaque appel à TFS pour récupérer un fichier, Visual Studio demandera une confirmation au développeur. Afin de ne plus afficher cette confirmation, il faut éditer ou créer le fichier « C:\Program Files\Microsoft Visual Studio 9.0\Common7\IDE\srcsrv.ini » et y ajouter les lignes suivantes :

```
[trusted commands]
tf.exe view
```

Cela indique à Visual Studio qu'il ne faut pas demander confirmation pour toutes les commandes commençant par « tf.exe view ».

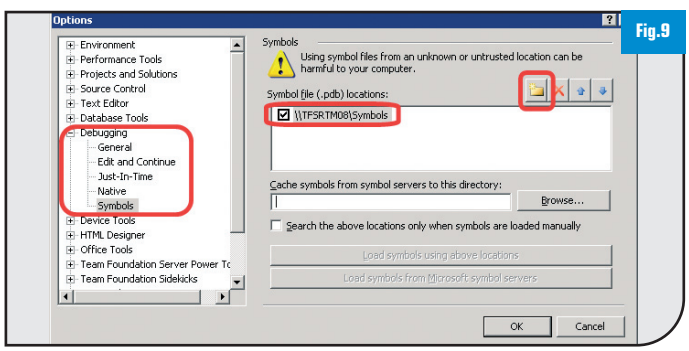
Le futur : TFS 2010

Le moteur de TeamBuild 2010 a été entièrement revu, il se base désormais sur des workflows écrits avec Workflow Foundation 4. Le workflow proposé par défaut intègre maintenant en standard l'indexation des sources et la publication des symboles [Fig.11]. Comme vous pouvez le voir, des activités spécifiques ont été écrites, ce qui permet de s'affranchir de l'installation des « Debugging tools for Windows » et d'ActivePerl. Le moteur de build se basant sur Workflow Foundation, les workflows sont facilement configurables via des arguments ce qui simplifie leur réutilisation, notamment dans le cas qui nous intéresse [Fig.12]. Il est bien entendu possible d'étendre le workflow de base ainsi que ses arguments pour par exemple spécifier le ou les répertoires ou chercher les fichiers de symboles à utiliser (par défaut tous les fichiers « .pdb » du répertoire « Binaries » sont pris en compte). Mais cela sera pour un autre article

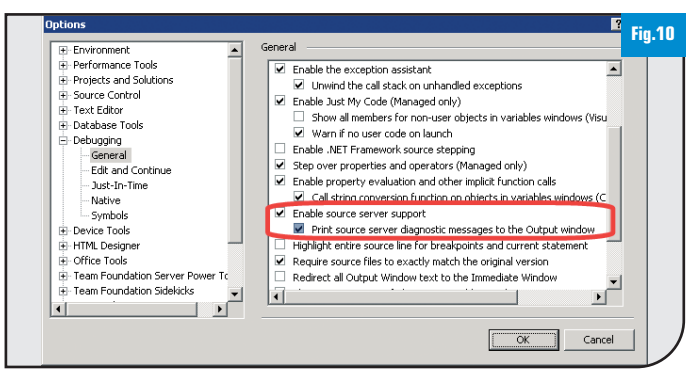
CONCLUSION

Lors de développement avec plusieurs équipes, le débogage peut vite devenir fastidieux lorsque l'on utilise des binaires fournis par les autres équipes du fait de la nécessité d'avoir à la fois les fichiers de symboles et les sources correspondantes au bon endroit sur sa machine. L'indexation des sources permet de s'affranchir de la contrainte sur les sources et le serveur de symboles de celle sur la présence en local des fichiers de symboles. L'intégration de ces deux processus au script de build permet de rationnaliser cette démarche et de garder une cohérence pour un coût minime. Coût encore plus minime avec la version 2010 de TeamBuild et TFS.

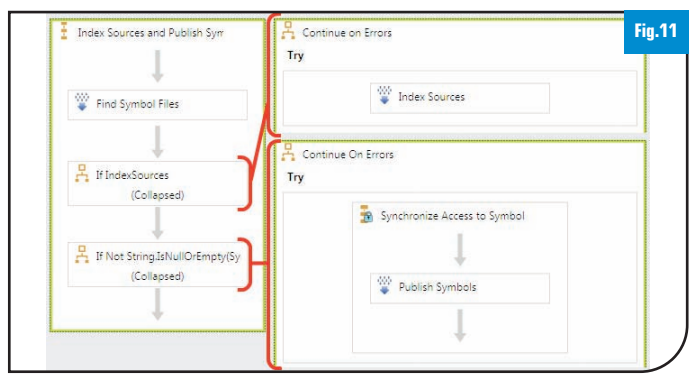
■ Guillaume **Rouchon**
Architecte .Net et expert ALM pour les Sogeti .Net Rangers
Blog : http://blog.qetza.net



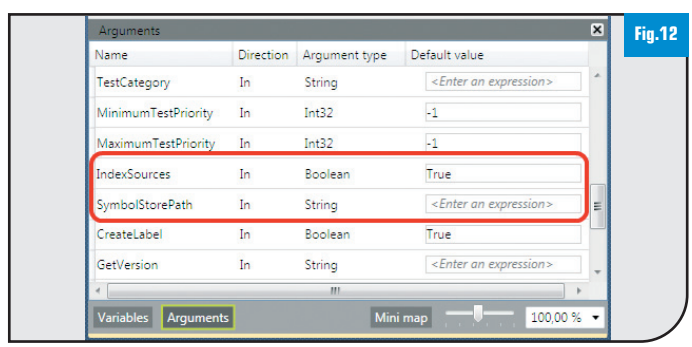
Configuration du répertoire des symboles



Configuration du serveur de source



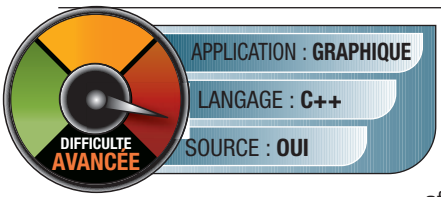
TeamBuild 2010 1/2



TeamBuild 2010 2/2

Programmation parallèle avec OpenCL 1re partie

OpenCL est un environnement de programmation parallèle qui permet d'écrire du code portable exploitant la puissance des processeurs multi-cœurs et des processeurs graphiques. Découverte et maniement de base.



La tendance est de plus en plus à l'omniprésence de la programmation parallèle, et c'est une tendance qui devrait durer longtemps. En effet la montée en puissance des processeurs, qu'ils soient classiques (CPU) ou graphiques (GPU) ne se trouve plus tellement dans l'augmentation des cadences d'horloges. Un jour viendra sans doute où les processeurs seront construits en diamant et seront capables de supporter des températures extrêmes, et donc des cadences d'horloges infernales. En attendant que ces composants quittent les laboratoires de recherche et s'installent dans les ordinateurs de bureau, les processeurs augmentent leur puissance en augmentant leur nombre de cœurs. Et comme d'un autre côté les applications sont de plus en plus gourmandes en puissance, l'exploitation des cœurs d'une puce et donc la programmation parallèle devient un passage obligé. On peut se poser alors la question : la programmation parallèle, oui, mais comment et pourquoi faire ? Si l'on recherche l'écriture facile, des frameworks comme CCR ou Axum (Programmez! 120 et 126 respectivement) qui s'appuient sur .Net pourront convenir. Si l'on recherche la performance la meilleure pour des calculs lourds [Fig.1], à base de code natif et permettant l'exploitation des CPU et des GPU, tout en écrivant un code portable à partir d'un standard, alors on se tournera vers OpenCL

1 PRÉSENTATION D'OPENCL

OpenCL est un standard défini par le Khronos Group. Khronos Group s'occupe aussi d'OpenGL, la librairie graphique 3D bien connue. OpenCL est une API définie par ce même Khronos Group. Cette API de bas niveau vise à tirer parti des possibilités de programmation parallèle offertes par les CPU et GPU modernes. Très logiquement une interopérabilité OpenCL/OpenGL est prévue. OpenCL présente des analogies avec CUDA de NVIDIA. CUDA vous a été présenté dans Programmez! 117 et 118 et c'est peut-être une bonne occasion de relire ces articles. Cependant, de l'humble avis de votre serviteur, une fois bien comprise, OpenCL se révèle plus facile et agréable à manier que CUDA. Notamment le compilateur OpenCL n'est pas un outil externe, mais il est intégré à l'implémentation et le code OpenCL est compilé à la volée à l'exécution.

2 TERMINOLOGIE ET MODÈLE D'EXÉCUTION

Les tutoriels OpenCL sont pour l'instant très peu nombreux et nous sommes contraints de travailler avec les spécifications d'OpenCL que l'on trouvera à http://www.khronos.org/ ou dans les SDK OpenCL. Clarifions très succinctement ces spécifications. En OpenCL on travaille avec des plateformes qui sont des collections de devices gérés par une implémentation et permettant l'exécution de kernels sur les

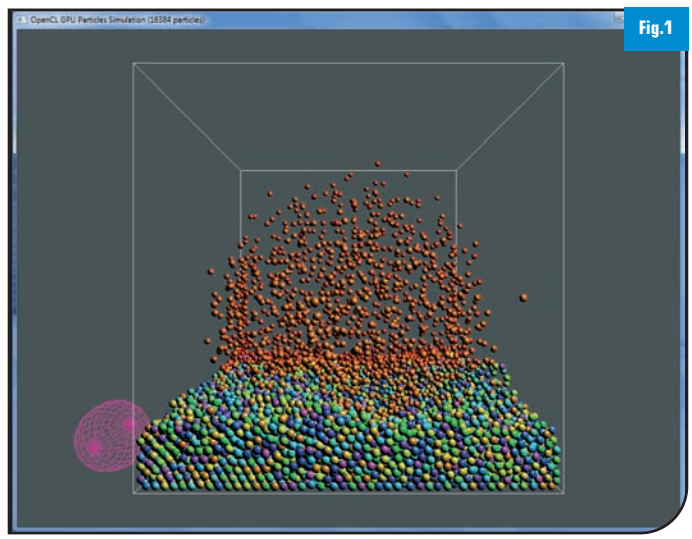


Fig.1

Ne manquez pas la démo 'particles' qui vient avec le SDK.

devices. Un device est une collection d'unités de calculs, un CPU multi-cœur est un device OpenCL, une GPU programmable également. Les kernels sont des fonctions compilées pour être exécutées sur des devices. Une instance de kernel qui s'exécute est une unité de travail, ou work item. Les unités de travail sont groupées dans un NDRange qui est un tableau à une, deux ou trois dimensions. Par commodité, il est possible de diviser un NDRange en groupes de travail, ou work groups, qui sont eux aussi des tableaux à une, deux ou trois dimensions. Pour obtenir l'exécution des kernels, le programmeur travaille avec un contexte, qui est un environnement qui gère un ou plusieurs devices. Le programmeur place les kernels à exécuter dans une file de commandes dite *command queue.* Cette command queue se charge de provoquer l'exécution des kernels, ainsi que du transfert bidirectionnel des données. L'exécution des kernels est asynchrone par définition. Le transfert des données peut lui aussi être asynchrone. Nous allons maintenant nous initier au maniement de tout cela.

3 LES OUTILS

Pour l'écriture de cet article, nous avons travaillé sous Windows sur des machines équipées de GPU NVIDIA et nous avons donc utilisé le SDK OpenCL for CUDA proposé par NVIDIA. Ce SDK, qui requiert une GPU GeForce 8 au minimum, est téléchargeable gratuitement et ne requiert aucune connaissance en CUDA. Par contre, veillez à installer un pilote dernier cri pour votre GPU. Le pilote qui va bien est le pilote de développement qui est proposé sur la page de téléchargement du SDK. Nous avons travaillé avec Visual Studio 2008, mais un simple compilateur C antédiluvien conviendrait parfaitement. Le SDK vient avec toute une panoplie de fonction «helpers» que nous n'avons pas utilisées. Nous avons au contraire préféré nous concentrer sur la compréhension et le maniement des API. Une fois familiarisé avec ces deux points, le lecteur pourra trouver

un avantage à l'utilisation de ces helpers. Le SDK OpenCL de NVIDIA a ceci de particulier qu'il ne prend en charge que les GPU. Il s'agit donc d'une implémentation OpenCL incomplète, mais cela ne gêne nullement l'apprentissage. Ceux qui disposent de machines équipées de processeurs AMD et ATI trouveront un SDK complet sur le site du fondeur. OpenCL étant un standard, cet article reste valable, et les sources des exemples aussi. Les lecteurs qui voudraient faire de l'OpenCL avec leurs processeurs Intel peuvent installer le SDK AMD. Cela fonctionne, pour les CPU uniquement, paraît-il…

4 CONNAÎTRE LES CAPACITÉS DE LA PLATEFORME

Avec ce genre de framework, le programme de base, le «hello world!» si l'on peut dire, consiste en un code qui permet de connaître les possibilités de la plateforme. Avec OpenCL, qui permet d'écrire du code portable, donc susceptible d'être compilé et exécuté sur des machines très différentes, cet aspect devra être traité avec soin, d'abord pour ne pas demander à la plateforme des choses qui lui sont impossibles, ensuite pour optimiser au mieux l'utilisation des possibilités qu'elle nous offre. L'optimisation en OpenCL est d'ailleurs un vaste sujet. Le lecteur trouvera un guide complet sur l'optimisation en OpenCL dans le SDK. Voici un extrait du programme de démonstration DemoQueries (au complet sur notre site) qui montre le principe d'une interrogation de la plateforme sur ses capacités. On commence au niveau le plus haut, la plateforme, puis on descend dans les devices, puis dans les capacités de ceux-ci. A chaque fois l'API remplit un tableau d'identifiants, puis on demande les détails relatifs à chacun des identifiants. La quantité d'informations pouvant être obtenues est énorme. Nous n'en extrayons que quelques-unes [Fig.2] et nous renvoyons le lecteur aux spécifications pour les autres :

```
#include <iostream>

using namespace std;

#include <CL/opencl.h>


int main(int argc, const char** argv)
{
 cl_int result;

 // Obtenir les identifiants de plates-formes
 const cl_uint num_entries = 10;
 cl_platform_id platforms[num_entries];
```

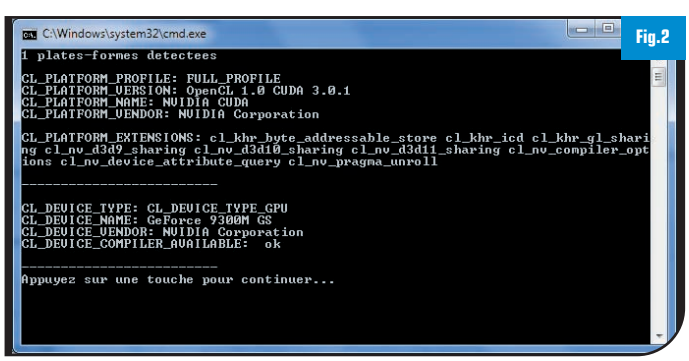


Fig.2

Nous interrogeons notre plate-forme sur ses capacités.

```
cl_uint num_platforms;
result = clGetPlatformIDs(num_entries, platforms, &num_platforms);
if(result != CL_SUCCESS)
{
    cerr << "Erreur clGetPlatformIDs" << endl;
    return 1;
}
cout << num_platforms << " plates-formes detectees" << endl << endl;

// Obtenir les infos sur les plates-formes
const size_t size_buffer = 255;
char buffer[size_buffer+1];
size_t param_value_size_ret;
for(cl_uint i=0; i< num_platforms; i++)
{
  clGetPlatformInfo(platforms[i], CL_PLATFORM_PROFILE,
    size_buffer, buffer, &param_value_size_ret);
  cout << "CL_PLATFORM_PROFILE: " << buffer << endl;

// etc, etc;
```

Dans ce code, pour simplifier, nous ne regardons que les devices de la première plate-forme trouvée. Ce code est aussi l'occasion de voir comment compiler une application OpenCL. Tout d'abord votre compilateur doit pouvoir accéder aux en-têtes d'OpenCL qui se trouvent dans le sous-répertoire 'OpenCL/common/inc' du SDK [Fig.3].
Ensuite, sous Windows, l'éditeur de liens doit accéder aux fichiers.lib qui se trouvent dans le sous-répertoire OpenCL/common/lin/win32 (ou x64 selon le cas) de SDK [Fig.4] et le fichier opencl.lib doit être ajouté à la liste des librairies, comme illustré. [Fig.5].
Enfin, lors de l'exécution, le système d'exploitation doit trouver la librairie partagée, opencl.dll ou opencl.so. Dans le cas de Windows, c'est immédiat car le SDK place le fichier dans c:\windows\system32. Sous un système de type Unix, on veillera, par exemple, à faire pointer opencl.so par la variable d'environnement LD_LIBRARY_PATH.

5 ADDITIONNER DEUX NOMBRES

Nous allons maintenant écrire notre premier 'vrai' programme OpenCL. L'objectif de celui-ci est très modeste: additionner deux nombres entiers. Mais comme cette addition sera effectuée de manière asynchrone dans la GPU nous en serons tout fiers ;-) Notre code (DemoAddition sur notre site) a surtout pour but de nous familiariser avec l'API et son maniement. Pour cette raison, il ne s'occupe pas (ou très peu) de gérer les erreurs, ceci dans un souci de légèreté et de clarté. Une application OpenCL suit (à quelques nuances près) toujours la même démarche :

- Obtenir une (ou plusieurs) plateforme.
- Obtenir un (ou plusieurs) device.
- Obtenir un contexte, qui est une abstraction permettant de travailler avec la plateforme et le device.
- Créer une file de commande (command queue) dans ce contexte.
- Créer des buffers et y copier les données.
- Créer un programme. Cette opération consiste en la compilation à la volée d'un ou plusieurs kernels.
- Instancier les kernels dans le programme.
- Définir les arguments à passer aux kernels.

- Mettre en queue un NDRange. Cette opération lance le calcul parallèle.
- Lire les résultats en les copiant dans un buffer.

Voici donc le code partiel ici, complet sur notre site :

```
#include <iostream>
using namespace std;
#include <CL/opencl.h>

const char* program_source[] =
{
 "__kernel void IntAdd(__global const int* a, __global const int*
 b, __global int* c)",
 "{",
   "int iGID = get_global_id(0);",
   "c[iGID] = a[iGID] + b[iGID];",
 "}",
};


int main(int argc, char **argv)
{
 const cl_uint num_entries = 10;
 cl_platform_id platforms[num_entries];
 cl_uint num_platforms;
 clGetPlatformIDs(num_entries, platforms, &num_platforms);

 cl_device_id devices[num_entries];
 cl_uint num_devices;
 clGetDeviceIDs(platforms[0], CL_DEVICE_TYPE_GPU,
     num_entries, devices, &num_devices);

 cl_context_properties properties[] = {
   CL_CONTEXT_PLATFORM,
   (cl_context_properties)platforms[0], // la première plate-forme
   0
 };

 cl_int error;
 cl_context context = clCreateContext(properties,
   num_devices, devices, NULL, NULL, &error);
```

```
if(error != CL_SUCCESS) {
  cerr << "Erreur clCreateContext" << endl;
  return 1;
}

cl_command_queue command_queue = clCreateCommandQueue (context,
  devices[0], NULL, &error);

if(error != CL_SUCCESS) {
  cerr << "Erreur clCreateCommandQueue" << endl;
  clReleaseContext(context);
  return 1;
}

cl_int a = 1; cl_int b = 2; cl_int c = 0;
// Buffer pour la donnée a
cl_mem buffera = clCreateBuffer (context,
  CL_MEM_READ_ONLY | CL_MEM_COPY_HOST_PTR,
  sizeof(cl_int), &a, &error);

// puis création des autres buffers. Cf source complet

cl_program program = clCreateProgramWithSource(
  context, sizeof(program_source)/sizeof(char*),
  program_source, NULL, &error);

cl_int result = clBuildProgram(program, 0, NULL, NULL, NULL, NULL);
cl_kernel kernel = clCreateKernel(program, "IntAdd", NULL);

result = clSetKernelArg(kernel, 0, sizeof(cl_mem), &buffera);
result = clSetKernelArg(kernel, 1, sizeof(cl_mem), &bufferb);
result = clSetKernelArg(kernel, 2, sizeof(cl_mem), &bufferc);

const size_t global_work_size[] = { 1 };

result = clEnqueueNDRangeKernel (command_queue,
  kernel, 1, NULL, global_work_size, NULL, 0, NULL, NULL);
result = clEnqueueReadBuffer(command_queue,
  bufferc, CL_TRUE, 0, sizeof(cl_int), &c, 0, NULL, NULL);
cout << "Resultat: " << c << endl;
```

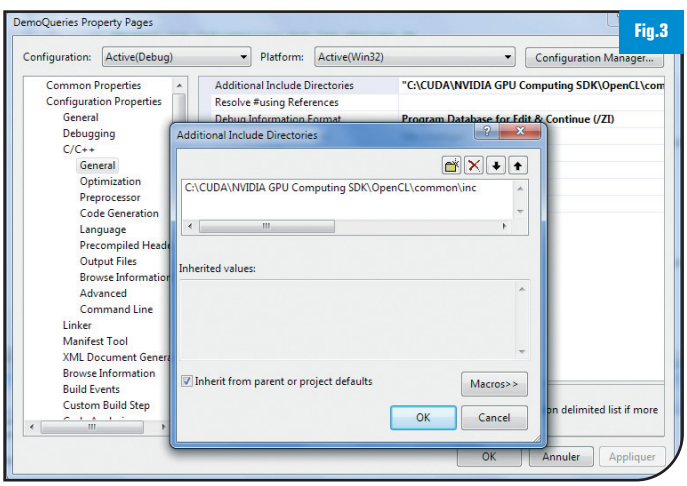


Fig.3

Configuration de Visual Studio pour qu'il accède aux en-têtes d'OpenCL.

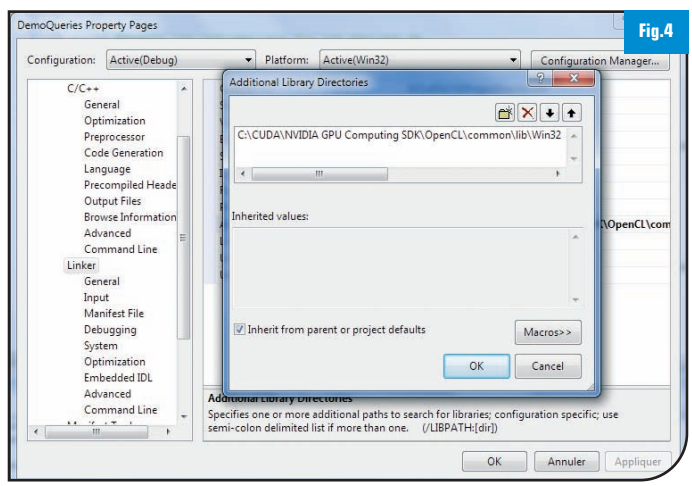


Fig.4

Visual Studio doit également trouver les fichiers .lib

```
  clReleaseKernel(kernel);
  clReleaseProgram(program);
  clReleaseMemObject(bufferc);
  clReleaseMemObject(bufferb);
  clReleaseMemObject(buffera);
  clReleaseCommandQueue(command_queue);
  clReleaseContext(context);
}
```

Commentons les points importants. Dans les rares exemples que l'on peut trouver ici ou là, le contexte est créé avec l'API clCreateContextFromType. Votre serviteur a eu grand mal à comprendre comme utiliser l'API clCreateContext et est heureux de vous montrer la solution dans cet exemple :-) Ensuite nous copions les données dans les tampons (buffers) lors de la création de ceux-ci. La copie est dont réalisée de manière synchrone. Il est possible d'effectuer cette copie de manière asynchrone, ainsi que nous le verrons dans l'exemple suivant. Nous prenons le soin de copier les données dans une zone de mémoire en lecture seule, puisque notre kernel ne doit pas les modifier. Utiliser de la mémoire en lecture seule peut faire gagner en performance dans le cas de gros volumes de données. Le code de notre kernel est :

```
__kernel void IntAdd(__global const int* a, __global const int* b,
  __global int* c)
{
    int iGID = get_global_id(0);
    c[iGID] = a[iGID] + b[iGID];
}
```

Nous utilisons get_global_id pour connaître, à l'exécution, le rang de l'instance du kernel exécuté dans le NDRange. Nous renvoyons le lecteur aux spécifications pour savoir comment procéder dans le cas de groupes de travail. Dans notre exemple le code de ce kernel est placé dans un tableau de chaînes de caractères, et c'est ce tableau qui est passé au compilateur. Bien entendu, il serait possible de placer le code du kernel dans un fichier et de charger celui-ci pour en passer le contenu au compilateur. Nous invitons le lecteur à bien étudier l'appel à clEnqueueNDRangeKernel. Dans cet appel, nous créons un NDRange de dimension 1. Nous passons donc conjointement un tableau de dimension 1 pour définir le nombre total d'instances de notre kernel, et donc de threads. L'unique valeur de ce tableau est 1. Nous lançons donc un seul thread. Nous récupérons le résultat des courses avec la fonction clEnqueueReadBuffer qui est bloquante. Le système garantit que tous les threads (dans notre cas un seul) sont éxécutés quand cette fonction rend la main. Et nous constatons finalement que même dans un GPU 1 + 2 = 3 :-)

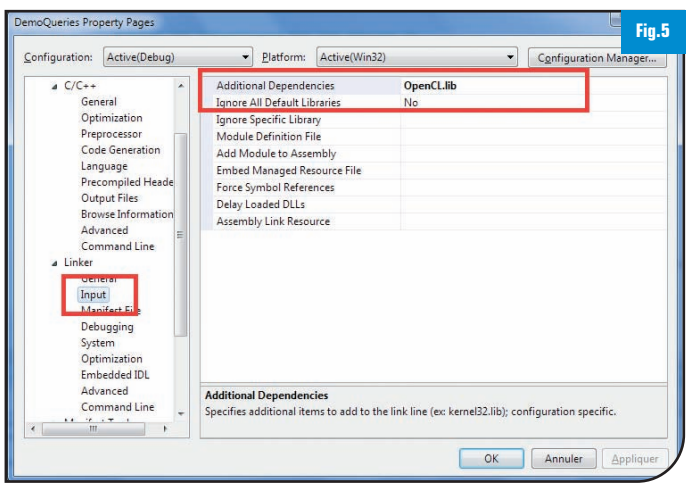


Fig.5

Une application OpenCL doit être liée au fichier opencl.lib sous Windows.

6 NOTIONS DE SYNCHRONISATION

L'art de la programamtion parallèle réside dans la synchronisation des threads. Pour les cas simples, OpenCL s'occupe de tout. Notre dernier exemple (DemoAdditionsAsyn sur notre site) additionne deux groupes de trois entiers, qui pourraient, par exemple, être les coordonnées de deux vecteurs 3D. Notre kernel est strictement le même que précédemment. Par contre cette fois nous ne copions pas les données lors de la création des buffers :

```
// Buffer pour les données a
cl_mem buffera = clCreateBuffer (context,
  CL_MEM_READ_ONLY,
  sizeof(cl_int)*dimension,
  NULL,
  &error);
```

Mais nous demandons que cette copie soit faite de manière asynchrone en plaçant une commande dans la file :

```
// Mettre le buffer a dans la queue et
// copier les données de façon asynchrone
result = clEnqueueWriteBuffer (command_queue,
  buffera,
  CL_FALSE,
  0,
  sizeof(cl_int)*dimension,
  &a,
  0, NULL, NULL);
```

Cet appel rend immédiatement la main. Nous procédons bien sûr de même pour le second buffer. Puis nous lançons 3 threads, chacun s'occupant seulement d'additionner deux coordonnées:

```
const size_t global_work_size[] = {
  dimension, // dimension vaut 3
};

result = clEnqueueNDRangeKernel (command_queue,
        kernel,
        1,
        NULL,
        global_work_size,
        NULL, 0, NULL, NULL);
```

Enfin, les résultats sont lus avec la fonction clEnqueueReadBuffer, comme pour l'exemple précédent. Comment sait-on que l'exécution des trois threads est terminée ? C'est le runtime qui s'en charge, et dans le cas de GPU NVIDIA la synchronisation est effectuée au niveau du hardware ainsi que nos l'avons vu dans les articles CUDA susmentionnés. Nous voyons donc qu'OpenCL nous évite de travailler explicitement avec des barrières dans les cas simples. Mais parfois les choses sont plus complexes… pour une prochaine fois :-)

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

DÉVELOPPEZ VOTRE SAVOIR-FAIRE

Economisez jusqu'à 50%

Programmez ! est le magazine du développement Langage et code, développement web, carrières et métier : Programmez !, c'est votre outil de veille technologique.

Pour votre développement personnel et professionnel, abonnez-vous à Programmez !
www.programmez.com

1 -25%

Abonnement 1 an

49€ au lieu de 65,45 € tarif au numéro - *Tarif France métropolitaine*

2 +0,8€ par mois

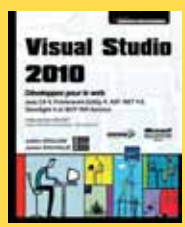
Abonnement Intégral : + archives

1 an au magazine + archives sur Internet et PDF

59€ *Tarif France métropolitaine*

3 jusqu'à -50%

Abonnement 2 ans + 1 livre numérique ENI



- **79€** au lieu de 130,90 (valeur de 22 numéros) *Tarif France métropolitaine* + un livre d'une valeur de 23,9 € à 31,9 €, soit un total de 154,8 € à 162,8 €
- **89€** 2 ans au magazine + archives sur Internet et PDF + 1 livre numérique ENI

OUI, *je m'abonne* — *Vous pouvez vous abonner en ligne et trouver tous les tarifs* www.programmez.com

- ☐ **Abonnement 1 an au magazine : 49 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*
- ☐ **Abonnement Intégral : 1 an au magazine + archives : 59 €** *Tarif France métropolitaine*
- ☐ **Abonnement 2 ans au magazine + livre numérique ENI : 79 €** *Tarif France métropolitaine*
- ☐ **Abonnement 2 ans au magazine + livre numérique ENI + archives : 89 €** *Tarif France métropolitaine*

Livres à Choisir : ☐ Visual Studio 2010 ☐ PHP5.3 ☐ Bing Maps ☐ MySQL 5, Administration et optimisation
☐ Java et Spring, Concevoir, construire et développer une application Java/J2EE avec Spring. *Détails sur www.programmez.com/abonnement.php*

☐ M. ☐ Mme ☐ Mlle Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

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

E-mail : @

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

A remplir et retourner sous enveloppe affranchie à : Programmez ! - Service Abonnements - 22 rue René Boulanger - 75472 Paris Cedex 10.
abonnements.programmez@groupe-gli.com

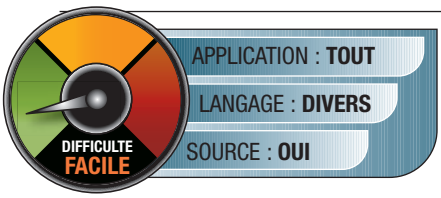
Offre limitée, valable jusqu'au 30 octobre 2010

Le renvoi du présent bulletin implique pour le souscripteur l'acceptation pleine et entière de toutes les conditions de vente de cette offre. Conformément à la loi Informatique et Libertés du 05/01/78, vous disposez d'un droit d'accès et de rectification aux données vous concernant. Par notre intermédiaire, vous pouvez être amené à recevoir des propositions d'autres sociétés ou associations. Si vous ne le souhaitez pas, il vous suffit de nous écrire en nous précisant toutes vos coordonnées.

PROG 134

Développer des applications RIA natives cross plateformes avec Titanium

Les applications riches, ou RIA sont très à la mode en ce moment, et les frameworks nombreux: Flex, Sliverlight, Pivot, JavaFX, etc. Découvrons aujourd'hui Titanium.



Est-ce un simple effet de mode ou quelque chose qui est destiné à durer ? L'avenir nous le dira. Mais le fait est que les RIA (Rich Internet Application) ou applications Web riches ont le vent en poupe, et l'abondance des frameworks commence à créer l'embarras du choix. Beaucoup de ces frameworks présentent de fortes, pour ne pas dire de très fortes similitudes. Grosso modo, nous trouvons un langage de script pour la logique de l'application, des fichiers XML pour la description des interfaces, et un runtime pour exécuter les applications. Souvent le runtime est disponible pour plusieurs plateformes, ce qui permet au développeur de cibler celle-ci avec un seul et même code. C'est dans ce contexte uniforme et peut-être un peu monotone, que se présente Titanium, le framework RIA de la société Appcelerator. Titanium présente un certain nombre d'originalités, qui sont autant d'avantages, et qui sont parfois des inconvénients. Prendre en main Titanium n'est pas très aisé, car il est mal documenté dans sa version gratuite et Open Source. Toutefois, une fois le cap de la prise en main passé, il se révèle amusant, léger et facile à manipuler. Titanium mérite qu'on le découvre, ce que nous faisons à présent.

1 VUE D'ENSEMBLE DE TITANIUM

Titanium cible de nombreuses plateformes mobiles ou fixes. iPhone et Android du côté des mobiles, ainsi qu'un support bêta de l'iPad qui est apparu au moment de la rédaction de cet article, et Windows, Linux, et Mac OS X du côté des ordinateurs de bureau. Un vaste programme, avec en plus, cerise sur le gâteau, des applications générées dans le code natif de la plateforme cible. Pour parvenir à ce résultat, Titanium se démarque sensiblement de la concurrence. Là où celle-ci utilise abondamment XML pour la description des interfaces, Titanium ne l'utilise pas, ou disons quasiment pas, pour être rigoureux. A la place, Titanium, privilégie les langages de programmation. Javascript s'y taille la part du lion, ce qui ne surprendra personne dans un contexte Web côté client. Mais Titanium se singularise par la possibilité qu'il offre de coder aussi en Ruby, en Python et en PHP. Autre singularité, il est possible d'utiliser ces langages alternatifs simultanément, et conjointement à Javascript. Un pont est réalisé entre ces différents langages par un objet de communication baptisé Kroll. Cependant il conviendra de faire attention à certaines subtilités dues aux différences existant entre ces langages. Notamment le passage des arguments de fonctions par valeur ou par référence peut induire des bugs difficiles à identifier. En plus des langages de programmation proprement dits, Titanium s'appuie sur HTML et CSS, contexte web oblige. Le moteur de rendu HTML est Webkit, le célèbre moteur libre. Tous les langages peuvent accéder au DOM d'une page HTML et le modifier. Tout ceci semble très intéressant, mais il y a aussi des points faibles, du moins de l'avis de votre serviteur, comme par exemple l'absence d'une librairie de Widgets. Du côté des applications de bureau, les API ne proposent rien d'autre que la construction d'un menu. Il semble que les concepteurs de Titanium estiment que les développeurs se rabattront sur des librairies de widgets tierces, comme il en existe en Javascript.

2 DES APPLICATIONS GÉNÉRÉES DANS LES NUAGES

Nous avons parlé plus haut d'applications natives et d'absence de runtime. Or Javascript, Ruby, Python et PHP nécessitent normalement un runtime. Contrairement à la plupart des SDK qui fournissent ou utilisent des compilateurs, l'outil de développement 'Titanium Developer' est surtout une sorte d'intermédiaire entre le code que vous avez écrit avec des outils tiers et un générateur d'applications exposé par l'infrastructure Cloud de Appcelerator. Vous devez donc être en ligne pour travailler avec Titanium, ce qui apporte l'avantage de la légèreté du Titanium Developper, mais l'inconvénient de ne pas pouvoir travailler lorsque l'infrastructure Cloud d'Appcelerator est hors service, ce qui est arrivé plusieurs fois à votre serviteur lors de la préparation de cet article. Avant la génération d'une application native qui est la dernière étape de la création d'une application, le Titanium Developer, manifestement écrit lui-même avec Titanium, permet d'exécuter votre code dans une sorte d'interpréteur, et permet aussi de le déboguer. Ce débogueur s'avère d'ailleurs très utile dans la pratique et son utilisation compense en partie la faiblesse de la documentation. Pour ces deux phases vous devez également être en ligne. Lorsqu'on en arrive à la génération du code natif, celle-ci se fait sur le Cloud d'Appcelerator, puis le

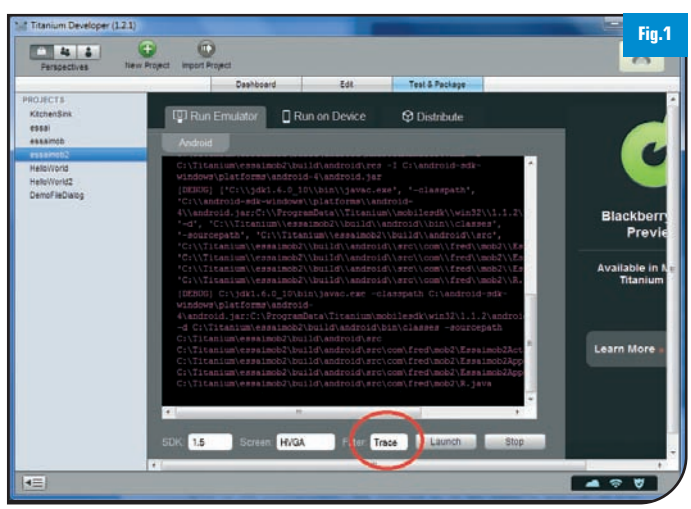


Fig.1

Passez le Titanium Developer en mode trace, pour vérifier qu'un déploiement sur Android se déroule normalement.

Titanium Developer vous fournit un lien sur l'archive vous permettant de la télécharger. Ce lien est valable tant que vous n'avez pas effacé votre application dans le Titanium Developer. Que fait à ce moment là l'entreprise Appcelerator ? Conserve-t-elle vos sources, vos binaires quelque part ? L'histoire ne le dit pas.

3 MANIPULATIONS DE BASE

L'installation de titanium est une simple formalité. Si vous voulez développer pour iPhone ou Android, vous devez bien entendu installer en plus le SDK correspondant. Une remarque en ce qui concerne Android. Je n'ai pu obtenir de résultats satisfaisants qu'avec la version 1.5 du SDK Android. En outre, si Titanium Developer lance l'émulateur Android et déploie votre application dessus, ce déploiement est extrêmement long, si bien que vous vous demanderez immanquablement si l'opération a échoué. Pour lever le doute, passez le Titanium Developer en mode Trace en cliquant sur le bouton 'Filter', comme illustré [Fig.1]. Les messages émis vous rassureront alors quant au bon déroulement des opérations. Nous nous intéressons maintenant exclusivement aux applications pour ordinateurs de bureau. L'illustration [Fig.2] montre la première phase, au cours de laquelle le développeur renseigne une boîte de dialogue. Une phase a priori anodine mais qui peut vous faire vous arracher les cheveux plus tard. Titanium Developer comporte un bug, au moins dans la version 1.0 utilisée pour la rédaction de cet article, qui fait que si le répertoire de travail de votre application est choisi sur un disque différent de celui où est installé Titanium, vous ne pourrez pas accéder au débogueur, à l'inspecteur de DOM, ni aux autres fonctionnalités indispensables pour la mise au point d'une application. Notons encore que c'est au cours de cette première phase que le choix des langages qui seront utilisés doit être effectué, en cochant la case correspondante. Il n'y a pas de case pour Javascript qui est systématiquement sélectionné.

4 HELLO WORLD !

Infrastructure Cloud ou pas, la tradition est la tradition. Créons donc une application du bureau basique, sans demander l'utilisation de langages particuliers. Titanium Developer va créer une arborescence de fichiers à la racine de laquelle vous trouverez un fichier tiapp.xml, qui est un descriptif de votre application. C'est aussi le seul document XML avec lequel vous allez travailler, et encore, de très loin. Ce fichier est bien documenté, nous ne nous y attardons pas, hormis pour une ligne :

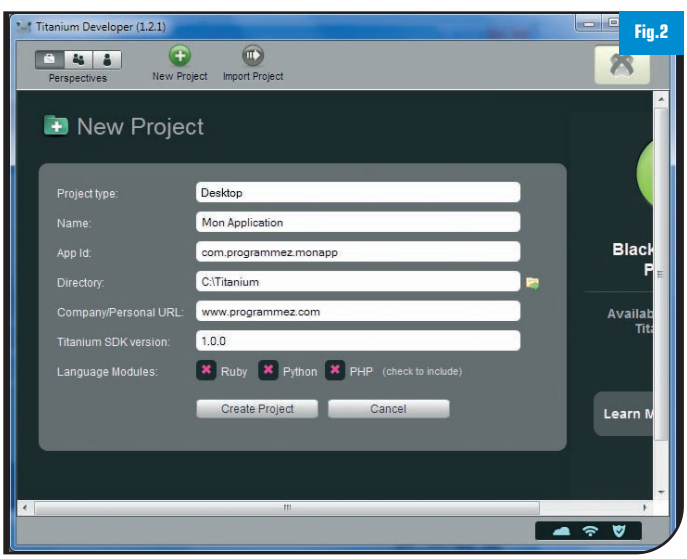


Fig.2

Première phase de la création d'une application Titanium.

```
<url>app://index.html</url>
```

Index.html sera bien évidemment la page d'accueil de notre application. Ce fichier se trouve dans le sous-répertoire 'Resources' de l'arborescence. Nous comprenons qu'une URL de protocole app:// permettra d'accéder par programmation à n'importe quel ficher dans ce sous- répertoire. Voici le fichier index.html de notre Hello World! :

```
<html>
<head>
</head>
<script type=»text/javascript» src=»myscript.js»></script>
<body style=»background-color:#1c1c1c;margin:0» onload=»mytrace();»>
<div style=»border-top:1px solid #404040»>
<div style=»color:#fff;;padding:10px»>Hello World Programmez!
</div>
</div>
</body>
</html>
```

Une page Web tout à fait classique qui fait référence à un fichier Javascript myscript.js. Ce fichier doit lui aussi être déposé dans le sous-répertoire 'Resources'. Nous voyons que Titanium sait charger ce fichier. Toutefois :

```
<script type=»text/javascript» src=»app://myscript.js»></script>
```

était également correct.

5 SAVOIR DÉBOGUER UNE APPLICATION TITANIUM

Regardons maintenant notre code Javascript :

```
function mytrace()
{
  Titanium.API.setLogLevel(1);
  Titanium.API.log(6, «Programmez!»);
  Titanium.API.log(6, «Abonnez vous ! :-)»);
}
```

Ce code montre un moyen que je vous recommande pour déboguer vos applications Titanium: le mécanisme de log. La première ligne du code définit le niveau de sortie des messages au moyen d'un numéro, documenté de façon erronée dans la documentation en ligne. Une fois ce numéro défini, on peut écrire dans un log, qui apparaît dans une console lorsque l'application en cours de développement est lancée par le Titanium Developer [Fig.3]. Titanium propose également un débogueur. Pour le lancer, vous devez d'abord lancer l'application, puis cliquer avec le bouton droit de la souris dans la fenêtre de votre application. Un menu contextuel vous permettra d'ouvrir le débogueur [Fig.4]. Celui-ci vous permet de modifier les éléments du DOM, et bien sûr de tracer pas à pas du code Javascript. Quand une application ne démarre pas correctement, ouvrir le débogueur sous l'onglet 'Scripts' permet en général d'identi-

fier immédiatement le problème. Supposons par exemple que dans index.html nous changions :

```
<body style=»background-color:#1c1c1c;margin:0» onload=»mytrace();»>

En:

<body style=»background-color:#1c1c1c;margin:0» onload=»matrace();»>
```

Le débogueur nous signale d'emblée que la fonction *matrace* est inexistante. [Fig.5]

5 HELLOWORLD 2.0

Voyons maintenant comment construire une application riche un peu moins pauvre que la précédente. Notre Hello World 2.0 présente plusieurs boutons. [Fig.6] Une action sur le premier modifie le DOM d'index.html, une action sur le second charge une autre page dans la fenêtre et une action sur le troisième ouvre une autre fenêtre qui charge la page www.programmez.com/. [Fig.7] Enfin, dans la fenêtre principale, nous installons un menu. L'exemple Helloword2, disponible sur notre site réalise tout cela.

Sur le fond, ce code est tout simple, mais il mérite pourtant quelques remarques en ce qui concerne le corps de la fonction goto_programmez. Cette fonction crée et affiche la fenêtre qui va charger la page www.programmez.com/. Pour créer la fenêtre, nous commençons par passer une structure de données au format JSON qui définit les propriétés de la dite fenêtre. Les noms des propriétés, non documentés, se déduisent des accesseurs de la classe UserWindow. Ainsi à partir des accesseurs getWidth et setWidth nous déduisons l'existence de la propriété width. Le nom des propriétés est toujours en lettres minuscules. Ainsi, les accesseurs setURL et getURL correspondent bien à une propriété url et non URL. Une fois la fenêtre créée, on voudra légitimement la faire apparaître sur l'écran, et on constatera que les méthodes setVisible ou show ne fonctionnent pas. Il faut invoquer la méthode open. C'est contre-intuitif, mais c'est ainsi :-)

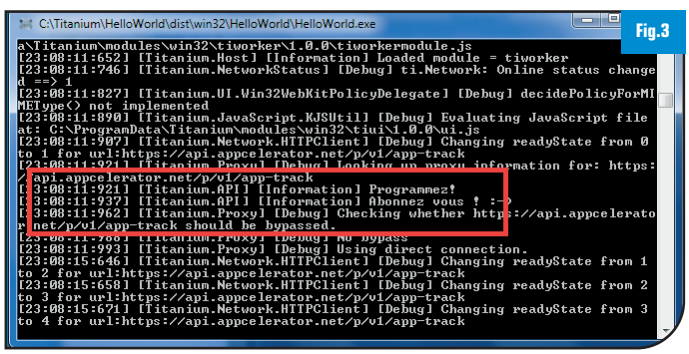


Fig.3

Le mécanisme de log permet l'affichage d'information tout au long de l'exécution d'une application.

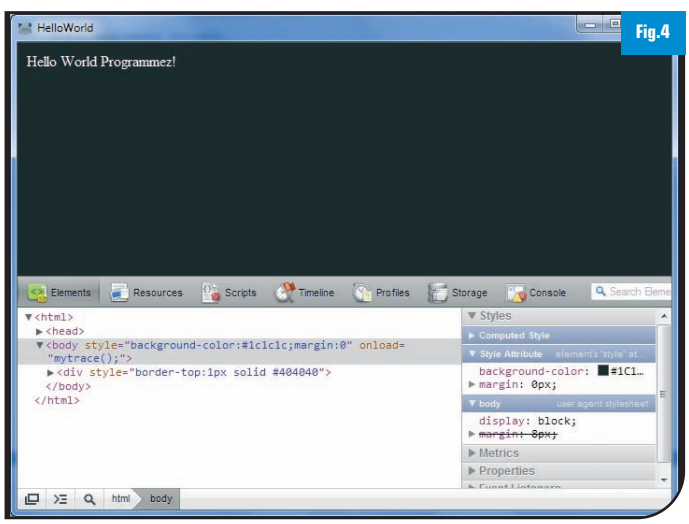


Fig.4

Le débogueur du Titanium Developer en action.

6 LES BOÎTES DE DIALOGUES

Une autre chose contre-intuitive est la documentation en ce qui concerne les boîtes de dialogue. Nous parlons de l'API des applications de bureau, car les choses vont différemment du côté des applications mobiles. La documentation mentionne l'existence d'une classe Dialog, et de méthodes telles que showDialog, qui donnent à penser qu'il est possible d'ouvrir des fenêtres sous la forme de boîtes de dialogues modales. D'autant plus que du côté mobile, une fenêtre devient modale lorsqu'on invoque la méthode open ainsi :

```
wnd.open({modal :  true});
```

Mais du côté des applications de bureau rien ne semble fonctionner. Une évolution de l'API et un correctif remédieront peut-être à cela plus tard. En attendant les seules boîtes de dialogues modales qu'il soit possible d'ouvrir, sont des boîtes de dialogues systèmes, pour la sélection de fichier ou de couleurs par exemple. Les noms de ces fonctions ne sont pas donnés par la documentation, mais il est possible de les trouver dans la console du débogueur. Ainsi nous écrivons dans cette console :

```
Titanium.UI
```

Puis nous pressons à plusieurs reprises la touche TAB afin de faire défiler les noms de fonctions existants [Fig.8]. Reste alors à savoir comment invoquer ces fonctions. Ces renseignements peuvent se trouver dans les F.A.Q à l'adresse http://developer.appcelerator.com. Voici maintenant, tiré de notre dernier exemple DemoFileDialog, le code qui ouvre une boîte de dialogue de sélection de fichiers :

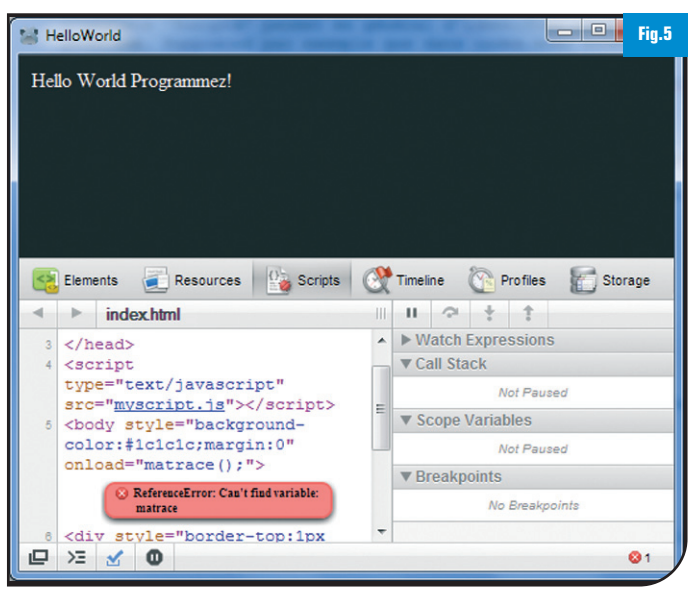


Fig.5

Le débogueur permet de localiser un problème au démarrage d'une application.

```
<html>
<head>
</head>

<script type=»text/javascript» >

callback = function(filenames)
{
    var file = filenames[0];
    alert(file);
}

function openfiledialog()
{
 var options = {
     title: «Programmez! Ouvrir...»,
     types: ['*'],
     typesDescription: «Tous fichiers»,
     path: Titanium.Filesystem.getUserDirectory()
 }

 Titanium.UI.openFileChooserDialog(callback, options);
}


</script>

<body style=»background-color:#1c1c1c;margin:0»>
<div style=»border-top:1px solid #404040»>
<div style=»color:#fff;;padding:10px»>
 Programmez!
</div>

<input type=»submit» id=»button» value=»Cliquez ici» onclick=
»openfiledialog();» />

</body>
</html>
```

Deux points sont à remarquer. Tout d'abord le filtre sous la propriété types. Ici, avec le caractère générique * nous acceptons tous les fichiers. Nous aurions pu, par exemple, sélectionner des fichiers World seulement en donnant une liste d'extensions de noms de fichiers :

```
['doc', 'docx']
```

Enfin, on notera la présence d'une fonction de rappel qui est invoquée lorsque l'utilisateur ferme le dialogue. C'est cette fonction de rappel qui permet d'accéder au(x) fichier(s) sélectionné(s).

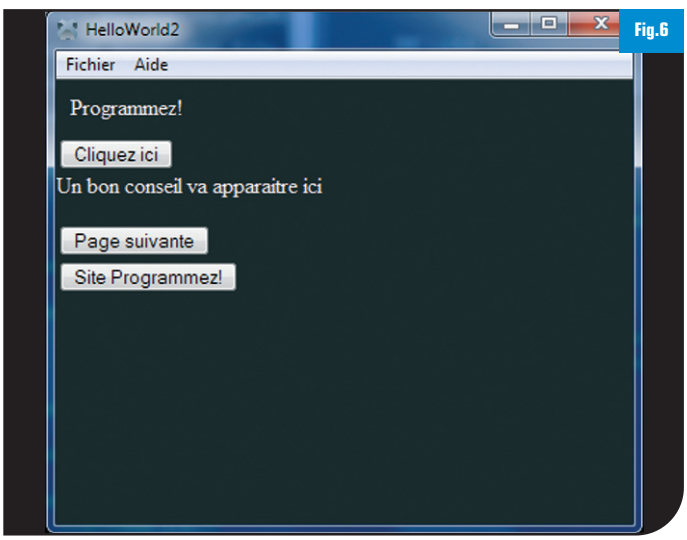


Fig.6

Hello World 2.0

7 CONCLUSION

Du côté des applications de bureau, Titanium semble encore manquer un peu de maturité, et la documentation est à améliorer. Cependant l'opportunité qu'il offre d'obtenir, à partir d'un seul code source, des applications natives prêtes à installer pour trois plateformes, Windows, Linux et Mac OS X, fait de Titanium un framework original et intéressant.

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

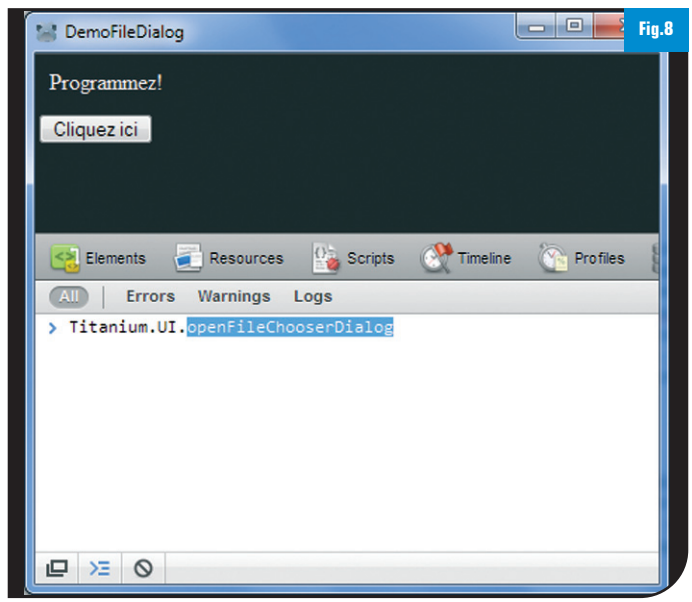


Fig.8

La fonction d'achèvement de code de la console du débogueur permet de découvrir des fonctions non documentées.

Fig.7

La page d'accueil du site de programmez, chargée par notre Hello World 2.0

WEB

Javascript

*Difficulté : ***
Editeur : MicroApplications
Auteur : Oliver Hondermarck
Prix : 15 €

Javascript est un des piliers du web 2 et du web tout court, mais souvent jugé complexe et lourd. Coder en javascript n'a jamais été une sinécure. L'auteur se propose de vous faire découvrir JS, comment on manipule des dates, des tableaux, comment écrire un cookie, gérer les interactions. L'ouvrage est assez complet et brasse l'ensemble des fonctions et commandes du langage. L'auteur mise aussi sur la pratique. Des exemples concrets et des exercices permettront de pratiquer tout de suite javascript !

LIVRE DU MOIS

Tableaux de bord de la sécurité réseau 3e édition

*Difficulté : **** - Editeur : Eyrolles*
Auteur : divers - Prix : 42€

La sécurité reste un sujet sensible et important pour l'entreprise, le développeur. Comment mettre en place une politique de sécurité réseau et utiliser les bons outils de contrôle et de pilotage pour le directeur informatique, RSSI ? Après avoir répertorié les attaques auxquelles peut être confronté un réseau d'entreprise, l'auteur décrit les différentes étapes de la mise en place d'une politique de sécurité : analyse des risques et expressions des besoins de sécurité, définition de la politique de sécurité réseau (recueil de règles), choix et déploiement des solutions techniques, mise en place de procédures et d'outils de contrôle. Parmi les nouveautés de cette troisième édition : la sécurité des systèmes (pare-feu. virtualisation, etc.), la sécurité des services réseau (IPv6, protection par topologie réseau pseudo-wire, VPLS, VPN MPLS/BGP, etc.), la sécurité de la zone d'administration (isolation en profondeur, création de niche par domaine de confiance), etc. Une nouvelle partie est dédiée à la supervision de la sécurité, un outil crucial dans la sécurité moderne. Un must à lire et à relire !

CLOUD

Intégrer Google Apps dans le SI

*Difficulté : ***
Editeur : Dunod
Auteur : divers
Prix : 28€

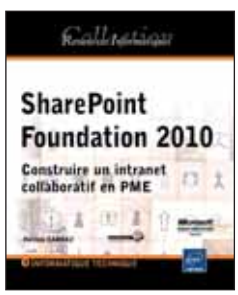
Le Saas est désormais une commodité pour le système d'information. Google est un des éditeurs les plus actifs sur le domaine avec son offre Google Apps : messagerie, agenda, traitement de texte, tableur… pour un prix deux à cinq fois moindre que les offres classiques. Même si Microsoft et d'autres répondent avec des offres similaires. Cet ouvrage fait le tour des questions relatives au passage à la solution Google Apps pour des entreprises moyennes ou grandes. Il permet au lecteur d'élaborer une trajectoire d'adoption sur mesure et lui fournit des retours d'expérience.

WEB

Sites web avec WordPress 3.0

*Difficulté : ***
Editeur : Dunod
Auteur : Simon Kern
Prix : 24,90 €

Wordpress est l'un des outils de blog et de création de sites les plus utilisés et une alternative très complète à dotclear. Longtemps perçu comme simple éditeur de blogs, WordPress est devenu un système complet et puissant de gestion de contenu (ou« CMS ») permettant de concevoir et de gérer facilement un site Internet. L'objectif du livre est d'être avant tout pratique et de rentrer rapidement dans les entrailles de l'outil : Installation de WordPress et de ses ressources incontournables. Fonctionnalités de base et ajout de nouvelles. Gestion d'un site mono- ou multi-utilisateur. Création et utilisation de templates avec les template tags et les boucles WordPress. Maintenance et sécurisation d'un site. Adaptation de WordPress à l'Internet mobile, etc.



PORTAIL

Sharepoint foundation 2010

*Difficulté : ****
Editeur : Eni
Auteur : Patrick Carraz
Prix : 29,90 €

Ce livre sur SharePoint Foundation 2010 s'adresse aux Administrateurs Système et aux responsables informatiques de petites et moyennes entreprises. Il présente une méthodologie de mise en œuvre d'un outil de travail collaboratif dans ce type d'entreprise et expose les réflexions sur son utilité au quotidien. Le livre permet la compréhension approfondie du produit SharePoint Foundation 2010, de la préparation du projet au déploiement et à l'administration, sans oublier l'accompagnement des utilisateurs. Il s'adresse à toute personne sans expérience particulière sur SharePoint, souhaitant bénéficier d'un tour d'horizon complet de la brique de base de la gamme et il ne nécessite aucune compétence de développeur : les fonctions mises en œuvre dans les exemples ne contiennent aucun code et ne nécessitent pas de programmation.

LANGAGE

Programmation Principes et pratique avec C++

*Difficulté : *****
Editeur : Pearson
Auteur : Bjarne Stroustrup
Prix : 55 €

Écrit par Bjarne Stroustrup, créateur du langage C++, ce livre traite de l'ensemble des concepts et techniques de programmation, qu'il s'agisse de la programmation orientée objet ou de la programmation générique. Le livre bénéficie des talents de pédagogue et des années d'expérience de l'auteur : l'exposé des notions théoriques est systématiquement accompagné d'applications concrètes avec C++, l'un des langages les plus couramment utilisés dans le domaine du développement de logiciels. Théorie et pratique permettent d'acquérir la maîtrise nécessaire à la rédaction de programmes robustes et efficaces, et l'apprentissage par étape assure une progression continue. Un must !

Kevin, responsable du pôle **formation d'ISIMEDIA :**

"La nature ne m'a pas fait blonde à forte poitrine et je n'ai pas eu la chance de grandir sous le soleil d'Hawaï. Mais je développe tous les jours avec WINDEV et WEBDEV, et j'aime communiquer mon savoir-faire"

ISIMEDIA. Le spécialiste WINDEV / WEBDEV & WINDEV Mobile

Formations WINDEV / WEBDEV
Paris • Lyon • Montpellier • Nantes

Savez-vous qu'il est désormais possible de vous former à WINDEV et WEBDEV avec des professionnels du développement ?

Nos formations sont réalisées par **des collaborateurs d'ISIMEDIA** : ce sont avant tout des professionnels du développement qui vous feront bénéficier d'**un vrai retour d'expérience**. Leur expérience « terrain » est déterminante : WINDEV et WEBDEV sont en effet des outils riches qui disposent de nombreuses fonctionnalités destinées à raccourcir les temps de développement. Or, l'expérience montre que certaines de ces fonctionnalités nécessitent d'être employées avec précaution dès lors que l'on souhaite réaliser une application d'envergure, performante et maintenable. Nos formateurs sont donc des interlocuteurs privilégiés qui sauront mieux que des formateurs professionnels, vous conseiller pour réaliser des applications de qualité.

Formations inter-entreprises (cours standards)

INTITULÉ	DURÉE	TARIF HT
Prise en main	2 jours	890 € HT
Perfectionnement	3 jours	1 350 € HT
Prise en main + Perfectionnement	5 jours	1 990 € HT
Expert	3 jours	1 350 € HT

Plans de cours et calendriers disponibles sur notre site Web www.isimedia.com

INTITULÉ	DURÉE	TARIF HT
Prise en main	2 jours	890 € HT
Perfectionnement	3 jours	1 350 € HT
Prise en main + Perfectionnement	5 jours	1 990 € HT

Formations intra-entreprises

Le contenu est entièrement personnalisable : du plan de cours standard au transfert de compétences préalable à la réalisation d'un nouveau projet, nous pouvons réaliser des zooms approfondis sur certaines fonctionnalités de WINDEV et WEBDEV. Ces formations, dont la durée varie entre 2 et 5 jours, font l'objet d'un devis spécifique.

Les + d'une formation inter-entreprise dispensée par ISIMEDIA :

1. Une formation réalisée par un professionnel du développement WINDEV / WEBDEV
2. Une totale indépendance vis-à-vis de l'éditeur PC SOFT
3. Une machine par participant (équipée d'une clé WINDEV ou WEBDEV)
4. Une mise en application de chacun des thèmes abordés sur un projet concret
5. Les supports de cours remis sur clé USB
6. Des sessions organisées sur Paris, Lyon, Montpellier et Nantes
7. Des sessions limitées à 8 participants

Besoin d'informations?

Contactez-nous directement : Tél : 04 67 55 81 55 • E-mail : formation@isimedia.com

Rejoignez la communauté de développeurs Nokia,

et créez des applications
pour les millions de Smartphones Nokia
à travers le monde !

Découvrez les nouveaux SDK Nokia

Et distribuez vos applications sur Ovi Store

auprès de millions d'utilisateurs Nokia.

Rendez-vous sur
www.forum.nokia.com/develop

NOKIA
Connecting People*

OVI NOKIA