LE MAGAZINE DU DÉVELOPPEMENT

Programmez!

Mensuel - Juillet/Août 2008 - N°110

www.programmez.com

Développez FUN!

Créez en 3 clics votre jeu Web

- Une interface à la iPod Touch
- Un joystick Bluetooth en Lego
- XNA : l'usine à jeux de Microsoft

XML

Formats ODF-OOXML : les différences
Générer un DOCX en PHP.
"ProgrammezML":
créez votre langage XML!

TESTS

- Au cœur de PHPUnit
- Tests unitaires en Rails
- Tester une interface graphique

WEB
Grails : un Java à la sauce Ruby on Rails !

GUI
Découvrez la puissance de Qt

SOA
Maîtriser l'architecture SCA

PYTHON
Découvrir et utiliser DB API avec PostgreSQL

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

M 04319 - 110 - F: 5,95 € - RD

WINDEV®

DÉVELOPPEZ 10 FOIS PLUS VITE

GRATUIT
TELECHARGER LA VERSION EXPRESS

INCLUS EN STANDARD DANS WINDEV 12 :

Environnement professionnel de développement (IDE & ALM)

- Tout est en **français**
- Hot Line personnalisée **gratuite***
- Déploiement libre et **gratuit**
- Crée des .EXE sécurisés, des Web Services, des applications .NET, des applications Java...
- 32 bits, **64 bits**
- **Compilation** JIT
- Code multi-plateforme compatible **Mobile** et **Internet**
- Génération d'application **Java** à la demande
- Fonctionne en **TSE** et Citrix
- Générateur automatique d'**IHM**, avec charte graphique automatique. Création d'IHM «Vista» & «Vista Like» par utilisation de gabarits fournis
- Tous les Windows : 98, 2000, NT, XP , Vista, 2008...
- Générateur d'**Etats et Requêtes gratuit**, création de **PDF**, code-barres, étiquettes. Fond de page PDF
- **Automatique** dans chaque application: menu d'export vers Word, Excel, OpenOffice, XML, PDF; Graphiques 3D; Historique de saisie,... Envoi de mail, Macros utilisateur
- **eal-RAD** : Générateur d'applications complètes
- A (Architecture Automatisée d'Application): ez votre propre RAD (Patterns)
- **erFileSQL**, base de données Client/Serveur, Locale et ile sous Windows et Linux (version illimitée, **libre** et **uite**); Gère 4 millions de Téras
- à toutes les Bases de Données tierces: **Oracle, 0, SQL Server**, DB2, **MySQL**, **Access**, xBase, etc...
- **ation** multibases assistée
- tif
- tif à **SAP R/3**, Lotus Notes, LDAP, Outlook,...
- suivi du **planning** d'équipes
- suivi des retours utilisateurs
- Bord de **suivi de projet**
- Modélisation **UML**, Merise et Souple; code uis l'analyse, reverse engineering
- omatique : analyse & programmation
- tilisation de **composants** ; 3-tiers
- Gestion des exigences
- Génération **L5G**, qui élimine 90% du code
- L4G et L3G: C++, C#, Java, VB, Cobol...
- ations Access et VB
- tique (norme X10)
- rie RS 232, parallèle et **USB**
- th,
- **NMP**
- C, FTP, HTTP, Socket, Twain, API, DLL,...
- Editeur de code intelligent, avec test immédiat sans recompilation
- **Tests unitaires** de code et d'IHM automatisés, Editeur visuel de tests de non-régression
- Refactoring
- Outil de **versionning** sophistiqué (gestion des sources)
- Débogueur puissant: threads, composants,... **Débogage à distance**
- **Profiler**, pour optimiser la vitesse du code
- Multilingue automatique: jusqu'à **20 langues**
- Générateur d'aide CHM
- Fonctions Multimédia (image, son, vidéo)
- Générateur d'**Installations en 1 clic,** et mises à jour automatiques (local, à distance, via Internet)
- Autoformation en 1 semaine (**manuel livré**)

Offre réservée aux entreprises, administrations et professionnels, en France Métropolitaine. Le Logiciel et le matériel peuvent être acquis séparément; WINDEV Mobile au tarif catalogue de 1973,40 Euros TTC; le HTC X7510 livré en France métropolitaine au tarif de 1.084,64 Euros TTC et chaque HTC Touch Diamond livré en France métropolitaine au tarif de 721,31 Euros TTC (tarifs au 26/5/2008 modifiables sans préavis, matériel annoncé disponible à partir du 21 juin). Détail de l'opération sur www.pcsoft.fr ou appelez-nous.

ou (au choix)
2 HTC **Touch Diamond** pour 1 Euro de plus

PC SOFT 25 ANNÉE

Fournisseur Officiel de la Préparation Olympique

www.pcsoft.fr

Demandez le dossier gratuit (200 pages + 1 DVD) Inclut la version Express (gratuite) et 112 Témoignages détaillés. Tél: 04.67.032.032 ou info@pcsoft.fr

Logiciel professionnel Document non contractuel. Support technique gratuit: 15 requêtes sur la version en cours de commercialisation.

+ Tutoriel sur
www.programmez.com/tutoriel.php

SOMMAIRE N°110

CD-Rom 110 PROGRAMMEZ !

La nouvelle suite bureautique Open Source ! IBM Lotus Symphony 1.0

Version Windows

XNA Game Studio 3.0 CTP

La toute dernière version du studio de création de jeux de Microsoft. Développez dès aujourd'hui vos jeux pour Windows et Xbox. - Windows

Web 2.0

Expression Blend 2.5 CTP

Vous aimez Silverlight 2.0 ? Blend 2.5 est le compagnon idéal pour le web 2.0 de Microsoft - Windows

Ruby on Rails 2.1

Cette version 2.1 de Rails corrige et améliore la très prometteuse v2.0 de Rails qui s'impose comme LE framework web 2. - Windows

.Net

Sync Framework

La nouvelle pré-version du nouveau framework de synchronisation de données pour les applications .Net - Windows

Velocity

La nouvelle bombe .Net de Microsoft. Créez rapidement un système de cache distribué et des applications à haute disponibilité

PHP

Delphi for PHP

Développez plus vite en PHP grâce à Delphi for PHP de CodeGear. - Windows - 30 jours

PHP Unit

L'outil de tests PHP de référence

MySQL 5.1 RC

Linux & Windows

Open Source Wamp Server 2.0

La dernière version d'environnement serveur open source le plus complet

Tests

Web Performance Suite

Tester rapidement la montée en charge de vos applications web. - Windows - 10 utilisateurs virtuels - 15 jours

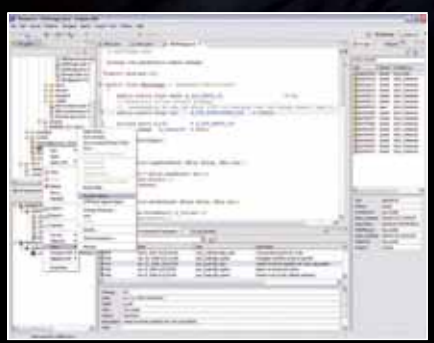
Web Performance Advanced Server Analysis

Votre application web est lente ? Analysez votre serveur pour découvrir l'origine du problème.

Perforce | Le système rapide de gestion de configuration logicielle

Présentation du module d'extension de Perforce pour Eclipse

Pour travailler avec Perforce dans une interface IDE Eclipse.



Module d'extension de Perforce pour Eclipse

Le module d'extension de Perforce pour Eclipse permet aux développeurs d'accéder facilement au système de GCL Perforce depuis leur interface IDE Eclipse. Il propose les fonctionnalités suivantes :

- Accès rapide à l'historique complet des fichiers
- Prise en charge complète du développement collaboratif, avec possibilité de fusionner les fichiers
- Possibilité de travailler hors ligne lorsque la connexion avec le serveur Perforce est indisponible
- Outil de comparaison des fichiers et prise en charge du suivi des défauts intégrés
- Prise en charge de la fonction de refactoring de l'environnement Eclipse

Le module d'extension de Perforce pour Eclipse prend en charge les systèmes d'exploitation Windows et Linux. Et ce n'est que l'un des nombreux outils intégrés dans le système de GCL Perforce.

eclipse FOUNDATION MEMBER

PERFORCE SOFTWARE

Téléchargez sans conditions une copie gratuite de Perforce sur www.perforce.com. Vous bénéficiez d'une assistance technique gratuite pendant votre période d'évaluation.

L'info permanente

Programmez!
LE MAGAZINE DU DÉVELOPPEMENT

Rédaction : redaction@programmez.com
Directeur de la Rédaction : Jean Kaminsky
Rédacteur en Chef : François Tonic
Ont collaboré : F. Mazué, L. Guillois, F. Dewasmes, J.M. Maman, J. Vidames .
Experts : E. Chenu, J. G. l'Helias, N. Biedermann, J. B. Escoyer, J. Pauli, F. Queudret, E. Hauchecome, F. Dubois, D. Guillaume, M. F. Landréa, P. Gourlain, A. Dupuis, S. Drapeau.
Dessin : Michel Piedoue
Illustration couverture : ©2004 The LEGO Group
Maquette : AJE Conseils
Publicité : Régie publicitaire, K-Now sarl
Pour la publicité uniquement : Tél. : 01 41 77 16 03
coordination@programmez.com
Editeur : Go-02 sarl, 6 rue Bezout - 75014 Paris
Coordination@programmez.com - Dépôt légal : à parution - Commission paritaire : 0707K78366 - ISSN : 1627-0908 - Imprimeur : ETC - 76198 Yvetot
Directeur de la publication : Jean-Claude Vaudecrane
Ce numéro comporte 1 CD Rom

Abonnement : Programmez 22, rue René Boulanger, 75472 Paris Cedex 10 - abonnements.programmez@groupe-gli.com Tél. : 01 55 56 70 55 - Fax : 01 55 56 70 20 - 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 : 45 € - Etudiant : 39 € - CEE et Suisse : 51,83 € Algérie, Maroc, Tunisie : 55,95 € Canada : 64,33 € Tom : 79,61 € - Dom : 62,84 € Autres pays : nous consulter.
PDF : 30 € (Monde Entier) souscription en ligne.

Édito

Jeux, interface et soleil ?

Enfin l'été. Et comme chaque année, on se demande ce que l'on va bien faire, au soleil ou à l'ombre, durant les vacances, à la maison, ou au bureau quand l'ambiance est plus calme. S'occuper du nouveau *buzz word* du moment ? Et celui-ci n'est plus Ajax, Web 2 ou autre Open social, mais tout simplement … l'interface.

A force de dire que l'interface fait son comeback, il fallait bien que cela arrive un jour, pour de vrai. Grâce aux plates-formes riches telles que XUL, Silverlight, Flex, AIR… elle redevient l'élément central. Cela ne résout pas un problème de fond : comment faire une bonne interface ergonomique, fonctionnelle, compréhensible ?

Le designer web a évidemment un rôle à jouer, essentiellement sur les applications web, RIA, voire, RDA. Mais il y a un élément à ne pas négliger non plus. Une application est aujourd'hui de plus en plus destinée à fonctionner sur une diversité de cibles : mobile, desktop, navigateur. À cela peut se rajouter une contrainte de contexte. Par exemple, quel type de données à afficher sur quel type de terminal, selon les critères de l'utilisateur, etc. L'interface se doit d'être flexible à l'extrême, totalement indépendante du code, de l'application. Elle doit être capable de se générer selon le contexte. Des langages descriptifs comme XUL, MXML ou encore XAML joueront un rôle primordial. Car eux seuls peuvent, aujourd'hui, fournir une interface à la volée. Ce n'est pas la tendance actuelle mais la direction est là. Ensuite se posera la question de ne pas utiliser à tort et à travers les effets graphiques et d'assurer la bonne portabilité de son interface quelle que soit la plate-forme. Un pari loin d'être gagné !

Le jeu connaît lui aussi son évolution, voire une mini révolution. La Playstation en son temps fut un choc pour le marché, même si la 3 a encore du mal à s'imposer, la Xbox (puis la Xbox 360) a titillé fortement le marché, mais c'est finalement la Wii qui a tout bonnement bouleversé les habitudes des joueurs en fournissant une interaction toute nouvelle grâce à sa manette. Le joueur interagit réellement avec le jeu. L'expérience est, je trouve, assez fantastique.

L'autre évolution est plus récente, avec le *multitouch,* les capteurs de force et tout ce qui est accéléromètre. Bref vous l'aurez compris, je parle d'iPhone. Les premières maquettes de jeux démontrent tout le potentiel de ces technologies. Et là le développeur a son mot à dire sur la conception, le développement. Certains cherchent à " démocratiser " le développement ludique. Microsoft le fait avec Popfly Game et XNA Studio. Mais comment créer un environnement " universel " pour supporter le plus de systèmes possibles ? Une des réponses se nomme Unity. Il permet de déployer sur le Web, Windows, MacOS X et Wii ! Seul défaut de cette merveille : la création est uniquement disponible sur Mac ! Personne n'est parfait…

Bon été et rendez-vous fin août avec plein de surprises !

François Tonic

PROCHAIN NUMERO *N°111 - Septembre - Parution : 1er septembre 2008*

La révolution du *"Multi Touch"*

Les nouvelles plate-formes se pilotent du bout des doigts : iPhone, HTC, Windows Seven, projets Majook et eLab Bouygues...
Comment ça marche ? Quels outils pour le développeur ?

SQL Serveur 2008

Découvrez toutes les nouveautés de la base de données Microsoft, lancée à la rentrée.

Brèves

OpenERP chez Smile

L'intégration Smile a signé un accord avec OpenERP (ancien TinyERP, progiciel de gestion),et un partenariat. Il inclut plus de 200 modules en GPL. Smile est fortement investi dans le marché open source, notamment sur la gestion de contenu.

Microsoft se virtualise dans Suse

Poursuivant leur collaboration, Novell a rejoint le programme Server Virtualization Validation de Microsoft. Cela devra garantir le bon fonctionnement virtuel de Windows Server 2008 dans l'hyperviseur XEN inclus dans Suse.

Une BI gratuite sur Google Docs

L'éditeur Prelytis lance LiveDashboard 4 Team, un service gratuit de tableaux de bord s'intégrant dans Google Docs. Il permet de construire soi-même des tableaux de bord. LiveDashBoard 4Team est une offre de type SaaS (Sofware as a Service). Il est utilisable gratuitement pour les groupes de travail. Site : www.livedashboard4team.org.

Une solution de préproduction multimédia

Celtx revendique le titre de première solution de préproduction multimédia open source. De l'histoire jusqu'à la production, l'outil prend tout en compte. Il supporte de nombreux formats de scripts (écriture, scénario), le versioning et la collaboration sont aussi inclus. Un bel outil qui a déjà conquis plus de 250 000 utilisateurs !
site : www.celtx.com

Agenda

JUILLET
Les 2 & 3 juillet 2008 à Paris, OCTO Technology organise la **1ère édition de l'Université du Système d'Information**. www.universite-du-si.com

Du 1er au 5 juillet 2008 à Mont de Marsan. **9e Rencontres Mondiales du Logiciel Libre.** www.rmll.info

WIDGET

Netvibes libère enfin son UWA

Voilà une bonne nouvelle, Netvibes a libéré sa plate-forme UWA, le widget maison qui se veut le plus universel possible, pouvant s'exécuter sur plusieurs plates-formes widgets. Pour cela, l'éditeur a lancé Netvibes.org qui doit permettre d'informer la communauté, de fournir la documentation, les codes, etc. Plusieurs projets sont d'ores et déjà disponibles comme UWA Javascript Runtime, une librairie PHP Exposition. Ces éléments sont pour le moment en pré-version. On dispose bien entendu des API Netvibes. Le tout est sous licence LGPL. Site : Netvibes.org

DONNÉES

Talend et le Change Data Capture

L'éditeur français a annoncé mi-juin l'intégration dans son outil d'ETL (intégration de données) de fonctions de capture de changement des données, qui permet d'identifier les données modifiées / changées. Intégré à la version 2.4 de Talend Integration Suite, Change Data Capture détecte en temps réel les enregistrements qui ont été modifiés, afin d'envoyer immédiatement des informations actualisées vers d'autres applications et d'accélérer les délais nécessaires pour les charger et les mettre à jour dans le cadre d'un processus d'ETL (Extraction, Transformation et Chargement). Cette fonctionnalité aide les entreprises à réduire leurs coûts en ne chargeant que les données modifiées et en n'utilisant qu'une partie des ressources sur lesquelles s'appuient traditionnellement les processus ETL.
" Les enrichissements apportés à la nouvelle version de Talend Integration Suite visent à accompagner les entreprises dans leurs projets temps réel. Cette nouvelle version accélère l'intégration des données, tout en la rendant plus évolutive et plus économique " déclare Fabrice Bonan, co-Fondateur et Directeur technique chez Talend. *" Talend s'appuie sur les retours de sa communauté d'utilisateurs pour développer les fonctionnalités et les composants les plus demandés. Ainsi, cette nouvelle version de Talend Integration Suite intègre de nombreuses nouvelles fonctions sollicitées ou suggérées par nos utilisateurs. "*

Linux

Une distribution sur mesure

Vous rêvez de concevoir rapidement et simplement votre propre distribution Linux ? Novell, via Suse, a lancé le projet SUSE Studio. Le but est simple : construire, tester et générer une distribution Linux clé en main. On définit en premier le nom de la solution puis on choisit les composants que l'on souhaite y voir : PHP, un serveur d'application, des jeux, une suite bureautique, etc. Par défaut, Studio propose de très nombreux packages (avec un moteur de recherche). On configure les propriétés du système comme l'interface ou encore la langue souhaitée. Un contrôle de l'intégrité du système est disponible afin de vérifier si des composants manquent afin d'éviter tout souci d'exécution. Puis Studio génère le fichier ISO (par la fonction Build) contenant la distribution. Trois exécutions sont possibles : en Live (sur CD, DVD ou clé USB), en installation classique ou via un environnement virtuel (XEN ou VMware). Voilà une heureuse initiative qui devrait faire fureur dès que l'on aura accès à l'outil car pour le moment, seules quelques personnes peuvent tester Studio SuSe.
Avis de la rédaction : Ce type de système à la demande est sans aucun doute un des avenirs du système d'exploitation. Côté Windows, on dispose de la même fonctionnalité avec Windows CE mais pas XP ou Vista. Il sera intéressant de voir comment Studio se comportera dès que la solution sera largement ouverte.
Site : www.studio.suse.com

COLLABORATIF

Jazz bientôt disponible !

IBM peaufine activement sa future technologie / plate-forme de collaboration : Jazz. Cela fait déjà plus de deux ans que l'on en parle. Les premières solutions sortiront dans quelques mois et IBM travaille dessus depuis longtemps. Jazz doit permettre de définir une autre manière de collaborer, de travailler en équipe, qu'elle soit ou non dispersée géographiquement. C'est un réel défi que de proposer une technologie globale. Jazz doit permettre de rendre plus flexible, plus souple l'organisation et de respecter les bonnes pratiques. Elle peut transformer la classique livraison logicielle en une intégration dynamique (des personnes), des processus et des projets, en s'appuyant notamment sur les principes des réseaux sociaux, des bonnes pratiques du web 2. Et IBM annonce la couleur : 20 outils seront estampillés Jazz d'ici fin 2008 ! Le premier, sortant fin juin, sera Rational Team Concert, incorporant des fonctions de réseaux sociaux, de la gestion de présence, de messagerie instantanée. Pour les petites équipes, IBM a prévu une version spécifique : Team Concert Express-C Edition (Linux et Windows), une édition gratuite (www.jazz.net) ! Et ce n'est qu'un début, car IBM a l'ambition de créer un écosystème à la Eclipse pour Jazz, ce qui promet de belles annonces, même chez les concurrents.

SGBD

4D livre son nouveau serveur

Sorti en septembre dernier, 4D v11 SQL a bouleversé le marché 4D en introduisant un moteur SQL dans le noyau. Jusqu'à présent, seule la version monoposte était disponible. Depuis quelques jours, la version serveur l'est également. Il s'agit en réalité de la 11.2. Cette édition reprend les nouveautés introduites dans 4D v11 SQL avec de nouvelles améliorations, comme le rassemblement de plusieurs commandes dans une unique commande pour rationaliser le langage de développement (cela doit se faire en toute transparence pour le code ancien). L'autre grosse nouveauté est la possibilité d'exécuter du code sur le serveur (sans être une procédure stockée), exploitant ainsi au mieux la capacité du serveur, le tout en mode synchrone et sans code supplémentaire (une option à cocher). La partie administration a été largement refondue pour être plus cohérente (on dispose ainsi de 7 onglets pour administrer son serveur). D'autre part, l'outil inclut désormais une zone web que l'on peut mettre dans un formulaire, cela permet d'afficher tout contenu web et de l'attaquer en Javascript. Cette fonction était à l'origine présente uniquement dans le pack web 2.0 de l'éditeur. D'autre part, l'outil 4D peut maintenant se connecter à un serveur 4D sans passer par un 4D Client (qui disparaît). Windows Server 2008 sera par contre supporté avec la prochaine version (v11.3 prévue cet automne). Enfin, le 4D Web 2.0 Pack sera mis à jour avec la sortie de l'iPhone 2.0 (et de son nouveau SDK). Prochainement, les développeurs Flex pourront se connecter à un serveur 4D. Site : www.4d.fr

Brèves

eyeOS mis à jour

Le système d'exploitation accessible via son navigateur web est aujourd'hui disponible en v1.6.0.1. Cette version ne propose aucune modification de sécurité, mais des fixations de bugs et la fonction import – export dans les eyeFeeds ou encore une sélection de langages dans la fenêtre de log. Site : http://eyeos.org/fr/

Des nouvelles d'Android

Aucune nouvelle version du SDK n'a été livrée depuis mars dernier. Il s'agissait surtout d'une version de maintenance et de correction de bugs. Cependant, régulièrement, de nouvelles applications apparaissent sur le site officiel, comme une horloge 3D. La sortie des premiers téléphones Android est toujours attendue pour fin 2008, début 2009 au plus tard.

Leirios devient SmartTesting

Editeur français venant, à l'origine, du monde de l'embarqué, Leirios a fait peu à peu évoluer son offre de tests vers les solutions de tests IT, tests d'application web. Le changement de nom qui s'est opéré début juin, avec changement complet de la charte graphique, du logo, etc., s'est fait dans la volonté d'adresser l'international. Le nom Leirios pouvait alors être un problème à l'étranger. Le nom Smartesting a été pris, le terme Smart Testing était utilisé par l'éditeur pour désigner l'intelligence des tests, son expertise, et les populations visées. Le but de l'éditeur est de s'intégrer avec les autres outils déjà en place dans l'entreprise et plus spécifiquement dans les tests fonctionnels. L'objectif de l'éditeur est de construire un véritable écosystème. Et pour 2009, l'éditeur souhaite monter dans les tests en visant les experts métiers qui ne sont pas techniques.

OPEN SOURCE

Anaska rejoint le groupe Alter Way

Mi-juin dernier, l'annonce est tombée. Anaska, leader français de la formation open source, rejoint Alter Way et complète ainsi l'offre open source du groupe qui comporte notamment les sociétés Nexen et Ingeniweb. Créé en 2002, Anaska fut à plusieurs reprises approché, comme nous l'a rappelé Cyril Pierre de Geyer. Après une nouvelle tentative d'Alter Way, début 2008. Les dirigeants finirent par accepter. Le pari initial d'Anaska était de devenir leader de la formation open source, chose faite depuis 2 ans. Il s'agissait de pouvoir trouver de nouvelles motivations et de nouveaux marchés. Il y a une véritable complémentarité avec Alter Way : formation, hébergement, développement, consulting, etc. Chaque entité du groupe garde son autonomie. " Nous avons saisi l'opportunité d'inscrire nos activités dans un projet ambitieux et un environnement caractérisé par une forte expertise technique. A titre personnel, nous avons également été motivés par la perspective de nous impliquer dans la gouvernance du groupe au sein d'une équipe dirigeante de haut niveau ", précise Romain Bourdon, l'associé de Cyril Pierre de Geyer (Anaska).

LANGAGE

Ruby on Rails 2.1 : disponible

Quelques mois après la disponibilité de la v2 de Rails, la communauté lance la v2.1. Cette évolution fixe les bugs de la v2 et ajoute quelques nouveautés. Comme l'écrit le blog officiel, depuis 6 mois, il n'y a pas moins de 1400 développeurs contribuant à créer 1600 patchs ! Les nouveautés sont : la Time Zones (gestion des fuseaux horaires), un système de cache amélioration, un nouveau mécanisme sur la définition et dépendances Gem. On dispose aussi d'une nouvelle fonction étendue de la fonction de recherche dans ActiveRecord : named scope. Cette version marque aussi l'utilisation d'un nouvel environnement de dépôt, de tracking, de bug : lighthouse.
Site : http://www.rubyonrails.org/

MOBILE

iPhone v2 arrive

Mi-juillet, la 2e génération du iPhone arrive. Avec une nouvelle version du système qui va apporter

bien des améliorations au niveau système mobile, des fonctions et de Safari. Les applications risquent de déferler par centaines très rapidement. Il y a deux éléments clés dans cet iPhone 2 : la partie entreprise (notamment avec le support d'Exchange de Microsoft), le SDK pour les développeurs. Le push de données sera donc possible comme avec un BlackBerry, ce qui devrait renforcer la présence dans l'entreprise. Cisco apporte son expertise dans la sécurité (disponibilité d'un VPN). On notera aussi l'arrivée (enfin) du GPS, de la 3G pour la téléphonie, d'un magasin virtuel pour les applications (App Store). Pour rappel, comme le système est un MacOS X sur mobile, on dispose des mêmes fonctions que sur un Mac : même noyau, même interface, même outils de développements. Cocoa Touch est un dérivé de Cocoa, adapté à la mobilité. Cocoa est un ensemble de librairies, de frameworks systèmes. Cocoa Touch inclut ainsi les événements, les contrôles pour le multi touch de l'écran, gère l'accéléromètre du téléphone (pour l'orientation, le retour de force), la caméra, la localisation, etc. Le tout disponible dans Xcode pour le codage et Interface Builder pour l'interface.

SYSTÈME

MacOS X : quand le Léopard se met au blanc

Durant la conférence mondiale des développeurs de la pomme, Apple a dévoilé la prochaine version d'OS X : Snow Leopard. Il sera disponible vers juin 2009. Aucune nouvelle fonction importante n'est attendue. Il s'agirait d'une version de réécriture d'une grande partie du système et d'optimisation de l'ensemble. Le but est de redonner une modularité, une meilleure robustesse. Bref, de stabiliser le système, de réduire la taille d'OS X et les besoins en mémoire vive. On s'attend à un support optimal du 64-bit, du multicore, d'OpenCL (pratique pour utiliser le processeur graphique à la place du processeur). Il est possible que le PowerPC soit définitivement arrêté dans le support système.

Infragistics®
Powering The Presentation Layer

Infragistics NetAdvantage®

Un potentiel sans limite pour Windows Forms, ASP.NET et WPF

Les meilleurs composants pour interfaces utilisateur disponibles
Tous les composants essentiels pour la conception et le style d'applications corporatives irrésistibles

Une expérience utilisateur hors du commun
Concevez des IHM exceptionnelles et ergonomiques pour multi-plateformes

Un cycle de développement accéléré
Réduisez vos délais de développement et assurer une livraison rapide avec des composants prêts à l'emploi

Une utilisation facilitée
Concevez des IHM reproduisant fidèlement le look & feel des dernières technologies Microsoft® avec un minimum de code

Retour sur investissement
Focalisez vos attentions sur l'aspect métier et permettez à la boite à outils Netadvantage d'assurer la cohérence, le style et l'utilisation simplifiée de l'interface de vos applications pour Windows Forms, WPF et WEB 2.0

Support complet pour l'IDE de Visual Studio®
Conçu pour compatibilité avec les plateformes de développement de Microsoft y compris Visual Studio 2008

Pour de plus amples informations:

infragistics.com

N° Vert 0800 667 307

Infragistics®
Powering The Presentation Layer
sales-europe@infragistics.com

WINDOWS FORMS ASP.NET WPF JSF

grids scheduling charting toolbars navigation menus listbars trees tabs explorer bars editors & more

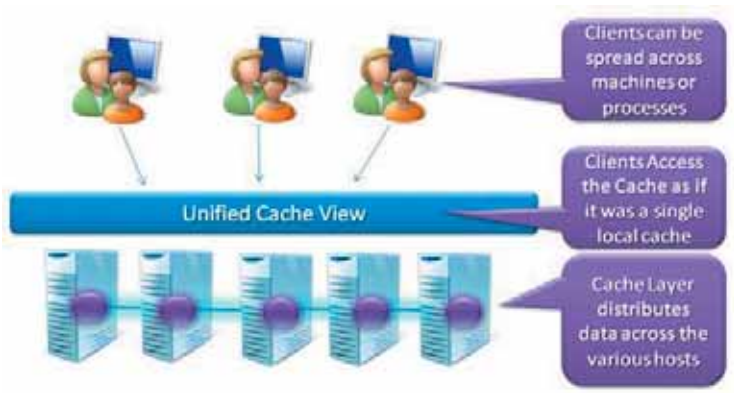
SYSTÈME

Microsoft renouvelle son embarqué

Microsoft mise beaucoup sur le marché de l'embarqué. Pour preuve, l'éditeur a décidé de faire oublier la dénomination Windows XP Embedded pour Windows Embedded Standard. La nouvelle version, estampillée 2009, est disponible en CTP (pré-version), la version finale est attendue pour le 4e trimestre 2008. Cette édition remet à niveau le système embarqué par rapport à Windows XP, notamment en intégration XP 3 et les dernières nouveautés comme .Net 3.5, IE 7 ou encore Network Access Protection. La version actuelle était en effet en retard. Un point sensible a été beaucoup amélioré : le Target Designer. Désormais, il sera possible de créer sa distribution et de la déployer en environ 20 minutes, selon l'éditeur. Concernant le développement, Windows Embedded Standard, il supporte bien entendu Visual Studio 2008. Si les fonctions multi-touch concernent avant tout le marché grand public, l'éditeur mise beaucoup sur le Machine to Machine, avec de plus en plus d'appareils, de terminaux interconnectés. À noter qu'avec cette prochaine version, Microsoft relance pour 10 ans le support du système alors que sur PC, XP arrive à son terme petit à petit. Quant à un Windows Embedded basé sur Vista, pour le moment, Microsoft reste discret même si cela est prévisible à terme, mais l'industrie n'est sans doute pas prête à cela.

FRAMEWORK

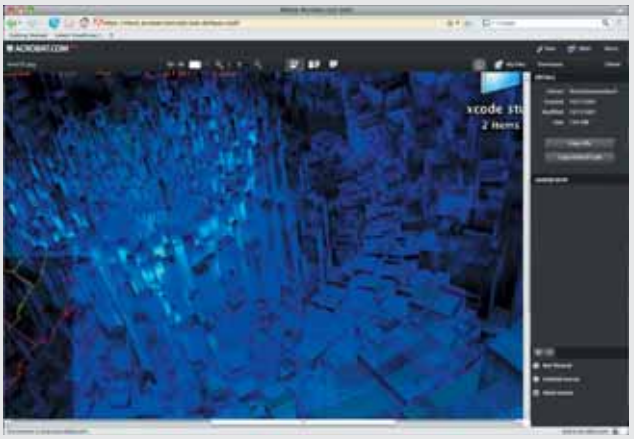
Velocity : l'autre manière de faire du cache



Microsoft a dévoilé il y a peu le projet Velocity. Il doit permettre de créer une plate-forme pour faciliter le cache distribué en mémoire pour les applications, facilitant ainsi la montée en charge, la disponibilité et les performances. Velocity doit ainsi utiliser la mémoire de plusieurs ordinateurs dans un cache unique (vu ainsi par les applications). Cela s'utilise dans les applications .Net. Ce cache peut être sérialisé en objets CLR. À noter : la possibilité d'utiliser des centaines d'ordinateurs pour faire le cache, des configurations communes, un rajout dynamique, le load balancing automatique, l'intégration avec le cache de ASP.Net, l'utilisation avec des multilangages client (PHP, C++, etc.). Avec ce nouveau projet, Microsoft renforce sa panoplie de frameworks pour le développeur après le Framework Sync (annoncé en version finale pour le 3e trimestre 2008) pour la synchronisation. Disponible en CTP. Pas de date pour la sortie finale.

FORMAT

Acrobat 9 est là !



Adobe a livré fin juin la 9e version d'Acrobat. La principale nouveauté est la présence d'un portfolio pour gérer, organiser et naviguer dans les documents. On dispose d'une indexation et surtout d'une recherche plein texte permettant une grande souplesse de recherche. On peut stocker, diffuser, consulter les documents. Cela ressemble à ce que l'on peut trouver dans un outil de GED. Il fonctionne partout où Acrobat fonctionne. Acrobat 9 arrive avec un nouveau module baptisé Adobe Presenter qui s'intègre dans PowerPoint et permet d'en récupérer et capturer les présentations, sans perte de qualité ou de fonctions. Bref Acrobat 9 étend ses capacités de dématérialisation des documents.

Pratique et puissant

Autre nouveauté importante, l'intégration poussée de la vidéo dans les documents PDF. A partir d'un glisser-déposer d'une vidéo, l'outil convertit automatiquement en format Flash Vidéo et l'intègre dans le document, et surtout on peut annoter la vidéo pour modifier des passages ! La collaboration et le partage sont des fonctions particulièrement importantes de la v9. Adobe veut simplifier le partage des documents entre plusieurs utilisateurs. On peut ainsi travailler sur un même document, faire des annotations, modifier le document et chaque utilisateur verra immédiatement le résultat. C'est pratique et puissant.

La partie collaboration passe notamment par le tout nouveau service en ligne (de type SaaS) : Adobe.com. Ce service se compose de 4 fonctions : création de PDF, partage (share), stockage en ligne de fichiers (My Files, jusqu'à 5 Go), Buzzword (traitement de texte, partage de document) et enfin ConnectNow (salle de réunion virtuelle jusqu'à 3 personnes). Gratuit et actuellement en version bêta, Acrobat.com se veut comme son espace virtuel de travail pour la collaboration, le partage, la création et le stockage de documents PDF. D'une interface dynamique et plutôt réussie, Adobe.com est une extension naturelle d'Acrobat et facilite le travail depuis son portable, son domicile. Un lien direct est d'ailleurs disponible entre Acrobat 9 et Acrobat.com. À noter que l'on disposera de nouvelles librairies de développement autour des fonctions d'Acrobat.com.

LEONARDi

...LYRIA REJOINT LE GROUPE W4 ! L'UNION DU BPM ET DE L'IHM MD... A SUIVRE...

DRIVING BUSINESS PROCESS EXCELLENCE W4

L'ihm

EN TOUTE SIMPLICITÉ !

Votre application de gestion multilingue avec plus de 100 vues de 20 types différents, en DHTML/Ajax, en Swing ou en plugin Eclipse, connectée à un SGBD et un bus JMS.

« il vous faut combien de temps pour la réaliser ? »

Si votre réponse est moins d'une semaine, inutile de vous rendre sur notre site, ni de télécharger la version gratuite de LEONARDI, sinon il est temps de passer à la vitesse "*Model-Driven*"...

NOUVELLE VERSION!

LEONARDI V4.0 OPEN SOURCE

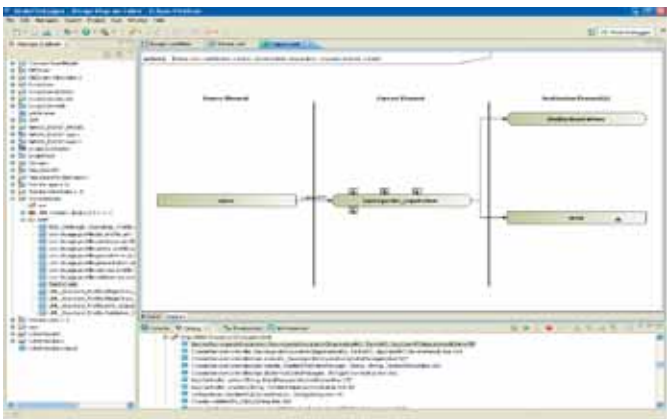
LYRIA

Pour en savoir plus sur la solution Leonardi, rendez-vous sur notre site www.lyria.com ou envoyez-nous un courriel à info@lyria.com

BLU AGE Edition 2009 : *réconcilier développeurs et modèles*

L'éditeur français Netfective se prépare à lancer l'édition 2009 de son outil de génération BLU AGE.

Les premiers éléments seront disponibles courant juillet 2008 et l'ensemble de cette nouvelle version en octobre. Cette édition 2009 apporte une réponse à l'un des freins majeurs à l'adoption des approches MDA (Model Driven Architecture) et MDD (Model Driven Development) : comment s'assurer de la conformité des modèles ? Car finalement, il est difficile, voire impossible dans une approche MDD " classique " de reproduire le principe du codage, compilation et test, particulièrement chers aux développeurs. Il faut donc avoir cette approche dans les modèles. Si le compilateur vérifie la conformité du code aux règles de composition d'instructions, alors il faut que le modèle soit vérifiable par un compilateur de modèles…



Tout sur Eclipse

BLU AGE 2009 complète sa panoplie fonctionnelle en étant totalement Eclipse (v3.3 et à présent 3.4). L'ergonomie a été revue en profondeur pour faciliter la prise en main et le travail sur les modèles. L'une des fonctions majeures de cette édition est la possibilité de valider, de déboguer les modèles en faisant un " simple " exécuter comme on le ferait dans Eclipse, NetBeans, Visual Studio. Cela permet de vérifier le comportement du modèle. Le débogueur supporte l'introspection et on peut modifier à la volée les valeurs. De plus, la modification du modèle peut être faite en direct dans l'environnement Eclipse. La validation passe aussi par l'utilisation des UML Best Practice. Il est important de pouvoir structurer la création des modèles et éviter que chacun fasse " sa sauce ". Le contrôle étant réalisé en amont par l'outil, on peut faire une vérification consistante, c'est-à-dire avoir des diagrammes et une interface consistants. Au bout du compte, on obtient un modèle fiable et donc une application conforme aux attentes. N'oublions pas que BLU AGE génère l'application de bout en bout sans rajouter une ligne de code, et ce à partir de trois input pouvant être réalisés en parallèle : les modèles, la description des IHM en xHTML, et le paramétrage des " cartouches " de génération (pour les différents langages et frameworks supportés).

La nouvelle version propose deux modules importants : le build et le deliver, se présentant sous la forme de plug-in Eclipse. Le build permet essentiellement l'élaboration des modèles, alors que le deliver focalise sur la notion de chaîne de production applicative avec des fonctions collaboratives. C'est aussi dans le deliver que l'on choisit les cartouches de génération, en fonction des technologies cibles souhaitées (libre à vous de créer vos propres cartouches). Les deux modules peuvent fonctionner indépendamment. Et l'éditeur parle aujourd'hui de IME pour Integrated Modeling Environment, le pendant de l'IDE pour la génération d'applications.

L'autre point intéressant de cette version est la possibilité de faire du reverse modeling. C'est- à-dire que l'on part du code d'une application existante pour créer un modèle, qui servira de base à la refonte de l'application. C'est le principe du code to model. On peut aussi utiliser autre chose que de l'UML pour le modèle, BLU AGE étant ouvert. Pour rappel, BLU AGE permet de générer les applications web pour .NET et Java. Une version Flex est en cours de développement, aucune date n'est annoncée pour le moment.

Site : http://www.bluage.com

■ **Jean Vidames**

Le problème de la validation

L'autre challenge est plus ambitieux : c'est la validation. En quoi mon programme répond au cahier des charges, à mes besoins fonctionnels ? Les calculs effectués par ce programme doivent en effet être l'image informatique fidèle et conforme des fonctionnalités exprimées et requises. Il " suffit " ainsi de tester, même si parfois l'énergie dépensée pour tester du code démontre souvent des erreurs de conception, et de façon consubstantielle, aboutit à des explosions de budget. Les modèles visent à atténuer de façon significative les erreurs de conception. C'est leur nature profonde. Un tel postulat ne peut véritablement devenir une réalité que si les modèles sont aussi, comme le code, testables, cela avec toute l'intuition dont les programmeurs font preuve lorsqu'ils pratiquent le test : les points d'arrêt, l'exécution pas-à-pas, l'introspection des objets et des variables, etc.

Pourquoi peindre avec les doigts ?

Visualisation Java pour clients riches et Ajax

ILOG JViews 8.1, la dernière version de la suite d'outils graphiques Java d'ILOG, couvre l'ensemble des fonctionnalités de visualisation avancée.

ILOG JViews 8.1 offre :

- Des composants graphiques puissants : diagrammes, courbes, tableaux de bord, cartographie, diagrammes de Gantt
- Des services évolués : agencement automatique de graphes, affichage performant pour des jeux de données volumineux
- Plusieurs techniques de déploiement : clients riches, applications Web interactives Ajax, Eclipse/RCP et portails
- Une expertise prouvée dans les industries les plus exigeantes : Informatique, télécoms, transport, énergie et défense.

Testez un de nos produits Java dès aujourd'hui http://jviews.ilog.com

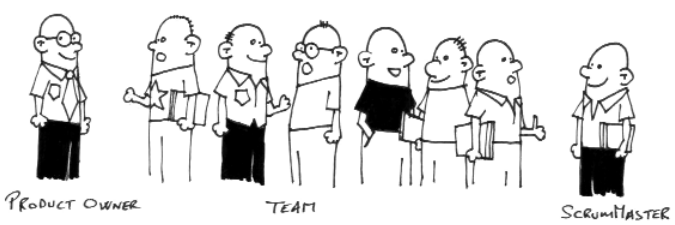
ILOG
Changing the rules of business™

Le petit SCRUM illustré

Scrum est une méthode Agile de gestion de projet mise au point par Ken Schwaber et Jeff Sutherland. Son principe est de responsabiliser et d'accorder la liberté d'organisation à une équipe pluridisciplinaire qui s'engage à construire un produit en incréments utilisables lors d'itérations de durée fixe.

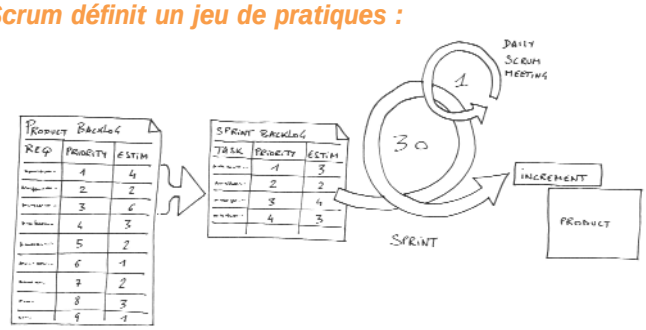
La méthode est légère, puisque basée sur un petit jeu de pratiques et quelques rôles. Même si la méthode est initialement destinée au développement de logiciels, elle peut s'appliquer à n'importe quel contexte où un groupe de personnes ont besoin de travailler ensemble pour atteindre un but commun.

Scrum définit des rôles :

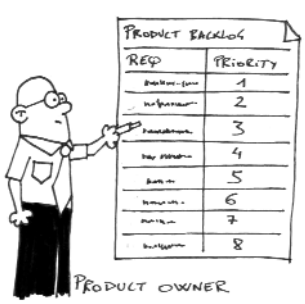


L'**Equipe** est composée de tous les profils nécessaires à la réussite du projet. Elle est responsable d'implémenter les fonctionnalités du produit, de résoudre les problèmes techniques et de l'organisation.
Le **Product Owner** est le responsable du produit à développer. Il représente le client et les utilisateurs. Il définit les fonctionnalités du produit et l'ordre dans lesquelles elles seront développées. Il est disponible pour répondre aux questions de l'Equipe.
Le **ScrumMaster** est responsable de l'application de la méthode Scrum. Il doit protéger l'Equipe des perturbations externes et doit supprimer tous les obstacles non-techniques qui gênent l'Equipe.

Scrum définit un jeu de pratiques :

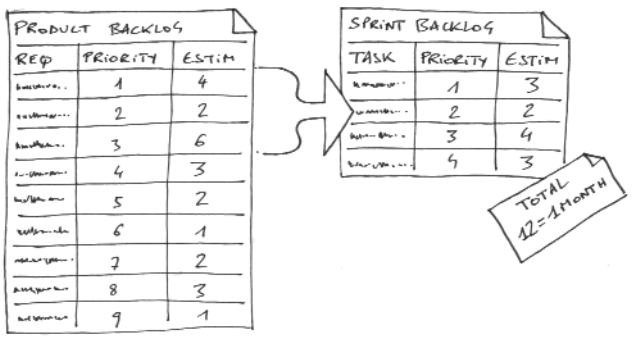


Le Product Owner est responsable de l'entretien du **Product Backlog**. Il s'agit de la liste des fonctionnalités du produit à développer. Le Product Owner attribue une priorité à chaque exigence de la liste.

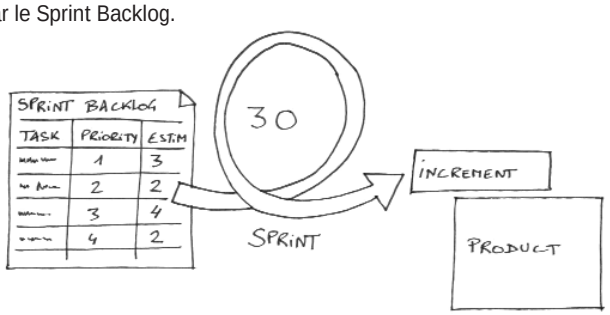


Lors de la réunion de planification de la prochaine itération, ou **Sprint Planning Meeting,** l'Equipe estime le contenu du Product Backlog.

En se basant sur ces pondérations, le Product Owner identifie alors le contour à développer lors de la prochaine itération parmi les fonctionnalités les plus prioritaires du Product Backlog. Le contour opérationnel réalisable lors de l'itération est transformé par l'équipe en tâches estimées dans le **Sprint Backlog.**



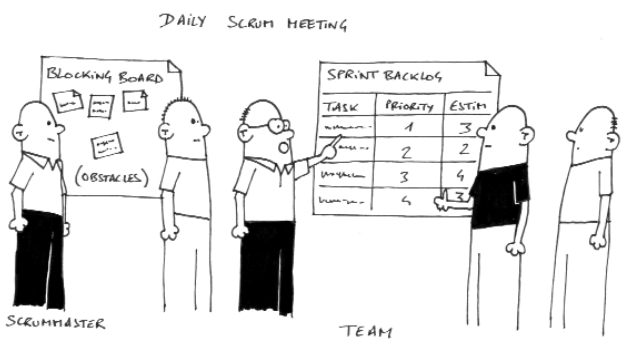
Le **Sprint** est une itération de trente jours calendaires pendant laquelle l'Equipe s'engage à implémenter l'incrément utilisable de produit défini par le Sprint Backlog.



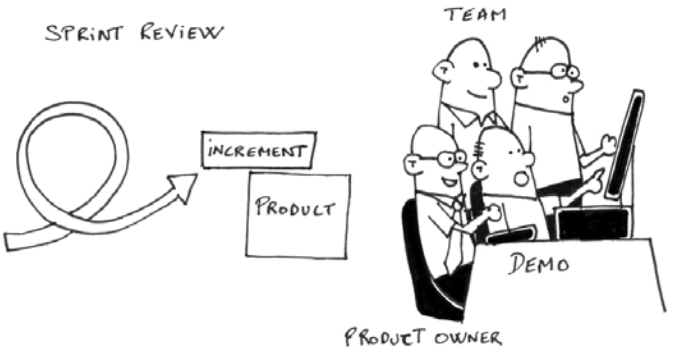
Le **Daily Scrum Meeting** est une courte réunion quotidienne à laquelle participent l'Equipe et le ScrumMaster. A tour de rôle, chaque membre de l'Equipe répond à trois questions:

- *" Qu'ai-je accompli depuis la dernière réunion? "*
- *" Qu'est-ce que je m'engage à accomplir aujourd'hui? "*
- *" Quels obstacles m'empêchent de tenir mes engagements? "*

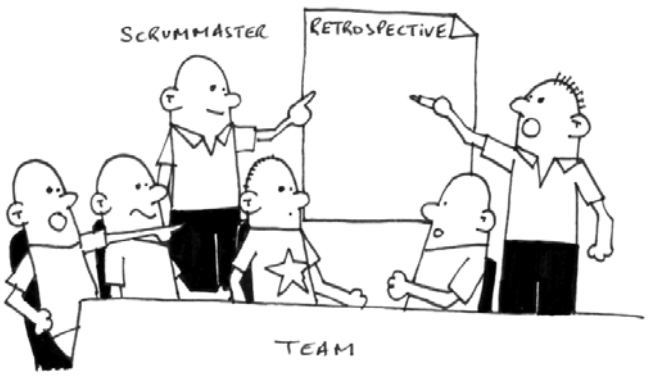
Les engagements pris pour la journée doivent concerner les tâches les plus prioritaires du Sprint Backlog.



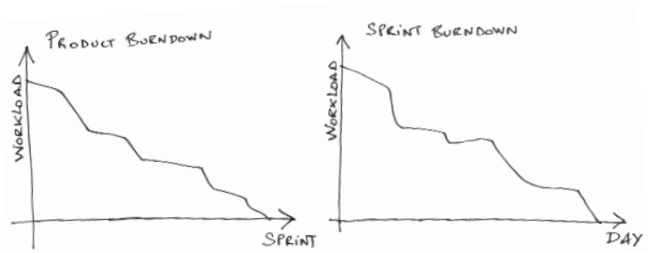
A la fin d'un Sprint, le Product Owner, le ScrumMaster et l'Equipe mènent la réunion de fin de Sprint, ou **Sprint Review Meeting.** L'Equipe présente au Product Owner l'incrément utilisable de produit réalisé lors de l'itération. Celui-ci doit alors faire part à l'Equipe de ses remarques.



Ensuite, lors du **Sprint Retrospective Meeting,** l'Equipe cherche à s'organiser pour améliorer son efficacité pour la prochaine itération.



L'avancement est mesuré à l'aide du **Product Burndown Graph** et du **Sprint Burndown Graph.** L'estimation du " restant à faire " est tracée en ordonnée pour la complétion du produit et du Sprint (respectivement). Le temps est représenté en abscisse en Sprints et jours (respectivement).



Pour aller plus loin :

- Agile Software Development With Scrum, Ken Scwhaber et Mike Beedle, 2002, ISBN 0-13-067634-9
- Agile Project management With Scrum, Ken Schwaber, 2004, ISBN 0-7356-1993-X
- Formations ScrumMaster certifiantes centralisées par ScrumAlliance http://www.scrumalliance.org/
- Communauté de praticiens francophones: http://fr.groups.yahoo.com/group/xp-france/

Emmanuel CHENU
emmanuel.chenu@gmail.com
http://emmanuelchenu.blogspot.com/

Eden Games gère ses jeux avec Perforce SCM

La gestion du code source, des versions, de build, des branches, est cruciale pour tout développeur. Eden Games, éditeur français de jeux créé en 1998, était précédemment sur SourceSafe, le gestionnaire de Microsoft. Mais en 2004, après plusieurs années d'utilisation, l'outil montrait ses limites. Une évaluation des outils du marché se fit alors en interne, avec des contraintes de multi projets, multi branches, d'exécution de scripts, ainsi que de disponibilité et de montée en charge. Le choix se porta sur l'outil de Perforce. Il est utilisé aujourd'hui par les 80 développeurs. Les codes sources des jeux sont stockés sur un unique dépôt (sauvegarde quotidienne). Trois gros projets sont gérés sur cette architecture : Alone in the Dark, Test Drive Unlimited et un outil interface d'intégration. Le tout avec des scripts d'administration, de maintenance. *" Notre exigence de qualité supposait que nous utilisions un logiciel de gestion de configuration qui nous donne une parfaite maîtrise des versions de jeux que nous livrons. Perforce répond tout à fait à ce besoin, notamment avec ses fonctions de ChangeList et de branchement "*, déclare **Frédéric Jay**, Responsable de l'équipe de R&D d'EDEN Software. Bref, l'infrastructure de développement sur laquelle repose la mise au point des jeux est primordiale pour Eden.

Jean Vidames

PAR FRANÇOIS TONIC

Lectures d'été

Collection " le guide complet "

- ***Difficulté : ** à *****
- ***Editeur : Micro Application***
- ***Auteur : divers***
- ***Prix : 20 €***

Dans cette collection ce mois-ci deux ouvrages sur des langages de référence :

- PHP 5, 3e édition : que diriez-vous de réviser vos connaissances sur PHP ? Cet ouvrage vous offre une plongée intégrale dans ce langage web : gestion date / heure, formulaires, tableaux, gestion des fichiers. Tout y passe. Plutôt bien écrit et clair.

- HTML 2e édition : et non, HTML n'est pas encore mort ! pourquoi ne pas revenir aux fondamentaux du web ? Même si HTML 5 n'est pas abordé, l'ouvrage a le mérite de revenir sur les bases de la programmation HTML, la compatibilité xHTML, le web 2 en général, les feuilles de styles, etc.

SQL – Tête la première

- ***Difficulté : *****
- ***Editeur : O'Reilly***
- ***Auteur : Lynn Beighley***
- ***Prix : 49 €***

SQL continue à vous faire peur ? Souvent à juste titre. Ce langage incontournable des données et SGBD, reste central dans les développements. Ce livre se propose de vous faire apprendre SQL (ou de le redécouvrir), d'une manière moins austère. Les grands classiques SQL sont abordés : SELECT, DELETE, UPDATE, les jointures, les bases multi-tables, les transactions. Bon point : une partie sur la sécurité. Le contenu est très dynamique et plaisant à lire. L'auteur cherche toujours à tenir éveiller le lecteur sur un sujet qui n'est pas forcément excitant. C'est plutôt complet et bien pensé. À recommander aux développeurs en mal de données…

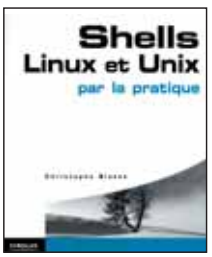
Programmation concepts, techniques et modèles

- ***Difficulté : *****
- ***Editeur : Dunod***
- ***Auteur : collectif***
- ***Prix : 29 €***

Finalement, c'est quoi la programmation ? Pour revenir aux fondamentaux mêmes du code et de sa philosophie, les auteurs se proposent de

reprendre les bases du développement avec les concepts (variables, fonctions, listes, dataflow, objets, etc.), la programmation déclarative, la programmation concurrente, la programmation avec état explicite, l'orienté objet… Le livre s'apparente à un cours. À posséder absolument pour se rappeler des fondations du code !

Shells Linux et Unix par la pratique



- ***Difficulté : *****
- ***Editeur : Eyrolles***
- ***Auteur : Christophe Blaess***
- ***Prix : 39 €***

Le shell est une des bases fondamentales des systèmes Unix. Mais encore faut-il le maîtriser pour l'exploiter au mieux. Qu'est-ce que le langage shell ? Comment le mettre en œuvre ? Comment bien l'utiliser, le coder ? L'auteur tente de vous apporter ses lumières avec un apprentissage progressif. Plutôt orienté étudiant, chaque chapitre se termine par des exercices pratiques pour savoir si l'on a bien assimilé. Vous voulez mieux exploiter votre Linux ? C'est le moment de s'y mettre !

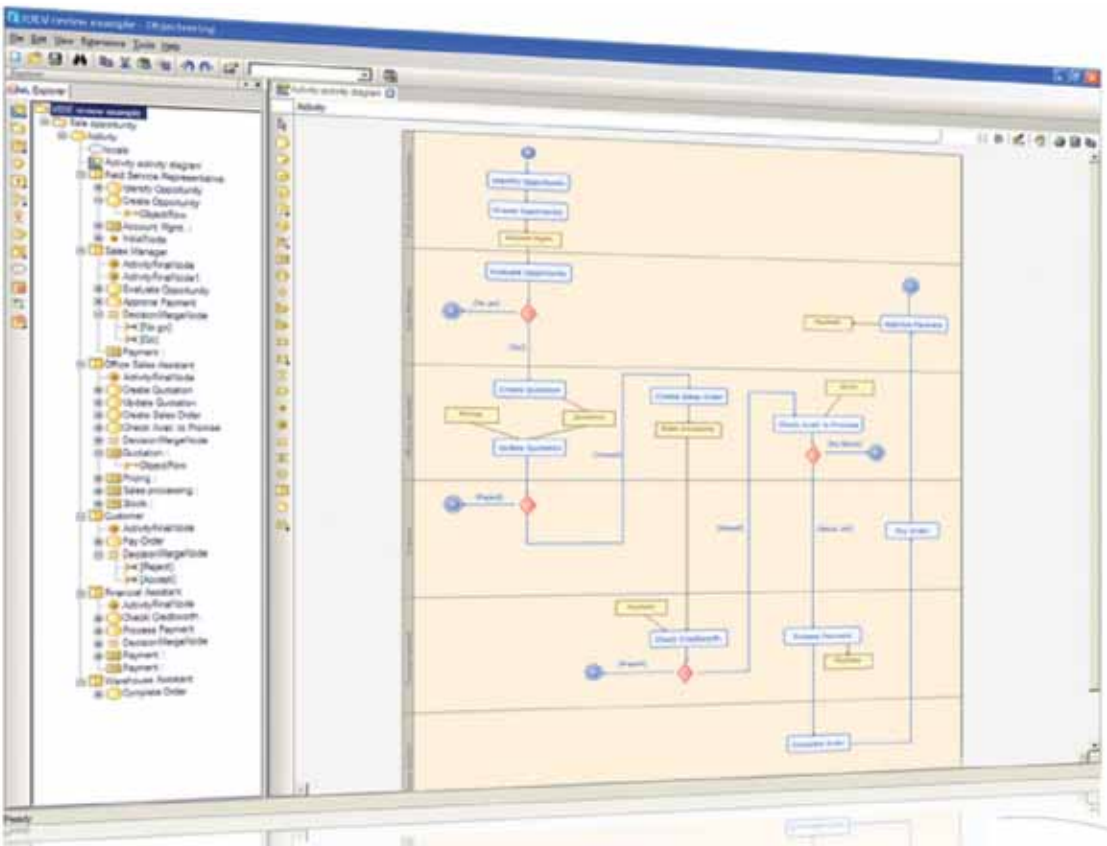
Premières applications web 2.0 avec Ajax et PHP

- ***Difficulté : *****
- ***Editeur : Eyrolles***
- ***Auteur : J-M Defrance***
- ***Prix : 39,90 €***

On parle depuis longtemps d'Ajax, de web 2, de PHP. Mais finalement, comment s'y prendre quand on est habitué à ASP ou à HTML ? On débute par une mise au point sur Ajax : de quoi il se compose, le modèle de développement, les +, les –… Les choses sérieuses débutent avec http et l'objet XMLHttpRequest, la base d'Ajax. Côté infrastructure, on aborde WAMP, Firefox, IE. Les outils de développement ne sont pas oubliés même si l'auteur se limite à Dreamweaver. La partie codage explique comment interagissent Ajax et PHP (ensemble), avec ou sans paramètres

Your projects deserve a tool*

Objecteering Modeler



Objecteering 6.1

guide vos développements par les modèles sur toute la portée du système

Objecteering 6.1 est une solution de modélisation complète et ouverte supportant les standards OMG de modélisation UML 2.1, BPMN, et SysML, la modélisation de l'Architecture d'Entreprise et la modélisation d'une architecture SOA.
Il intègre le support de l'analyse des besoins et de la définition du dictionnaire, assurant ainsi une traçabilité complète sur tout le cycle de vie.
La technologie MDA associée à une ouverture de l'outillage en Java permet de guider le développement à chaque contexte utilisateur, chaque infrastructure technique cible. Ses générateurs sur étagère automatisent le développement d'applications Java, J2EE, .Net C#, C++, SQL…

Objecteering Software, éditeur de l'atelier Objecteering 6, est le spécialiste français UML/MDA pour le développement d'applications guidé par le modèle. Son offre modulaire couvre le cycle de vie de la gestion des exigences jusqu'au déploiement d'application.
Pour plus d'information sur Objecteering 6.1, pour télécharger Objecteering **Free Edition** ou Objecteering **Enterprise Edition**, rendez-vous sur : www.objecteering.com

Architecture d'Entreprise
UML2 BPMN
C# MDA SysML
Java SQL
J2EE SOA
C++

* Vos projets méritent un outil

www.objecteering.com - Tél. : 01 30 12 16 60 - sales@objecteering.com

Objecteering SOFTWARE

The model-driven development company

(par exemple avec GET, POST). L'auteur n'oublie pas JSON, MySQL, ni les plug-in… Une parfaite introduction technique pour le développeur web.

Programmation concurrente et temps réel avec Java

- ***Difficulté : ********
- ***Editeur : Presses polytechniques et universitaires romandes***
- ***Auteur : Luigi Zaffalon***
- ***Prix : env. 64 €***

Tout le monde ou presque connaît Java. Mais on le connaît souvent plus mal dans le monde de l'embarqué, de l'enfoui et surtout dans le monde temps réel, car Java a souffert d'une mauvaise image. Cet ouvrage, un des rares en français, nous plonge dans le sujet. On débute par une longue introduction sur la programmation concurrente, temps réel, car pour bien coder dans ces domaines, il faut maîtriser et parfaitement connaître les notions, le formalisme, les processus, etc. Après cela, on attaque le vif du sujet avec les threads en Java, les verrous et sémaphores, l'exclusion mutuelle, etc. Puis, le temps réel débute (ou plutôt on étend notre travail) par la notion de temps : horloge et minuterie. L'auteur aborde naturellement un élément vital : l'ordonnancement et les threads temps réel, comment gérer la mémoire (un gros problème dans le temps réel et la programmation concurrente). On regrettera (un peu) l'absence d'un chapitre sur la sécurité. Vous l'aurez compris, cet ouvrage est à réserver à des développeurs Java voulant réellement aller très loin dans le langage. Et pour ceux qui veulent en savoir plus sur la concurrence et le temps réel, c'est le moment ! Un must incontournable.

L'art du beau code

- ***Difficulté : ******
- ***Editeur : O'reilly***
- ***Auteur : collectif***
- ***Prix : env. 45 €***

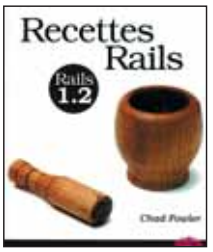
Un autre " must have ", que tout développeur devrait lire et relire tous les mois ! Plus que jamais, le code ne doit pas s'improviser. Les auteurs de ce livre abordent une sujet parfois tabou : comment structurer et avoir un beau code ? Les développeurs, plus de 30, dévoilent ici les bonnes pratiques, les usages, ce qu'il faut éviter, mais aussi les entorses à faire pour contourner un problème. Ce n'est pas un cours mais une succession de thématiques, de problématiques : les expressions régulières, les outils de type subversion, les frameworks de tests, les tests, la génération de code, etc. Chaque chapitre aborde une thème et le développe, le dissèque, fournit des exemples, et surtout une explication approfondie héritée d'une longue expérience de codage ! C'est beau et utile !

C++ pour les programmeurs C

- ***Difficulté : *******
- ***Editeur : Eyrolles***
- ***Auteur : Claude Delannoy***
- ***Prix : 32 €***

Nouvelle édition du Programmez en C++ du même auteur, il est devenu une référence pour les développeurs. L'objectif du livre est de proposer une migration, une compréhension du C++ et de ses concepts aux développeurs C. S'appuyant sur la norme ANSI/ISO, l'auteur prend en compte la bibliothèque standard et l'ensemble des aspects du C++. De très nombreux codes permettent de mieux comprendre les concepts expliqués.

Recettes Rails



- ***Difficulté : ******
- ***Editeur : CampusPress***
- ***Auteur : Chad Fowler***
- ***Prix : 30 €***

Autre livre de recettes, Recettes Rails. Il couvre la version 1.2. En 70 recettes du quotidien, l'auteur vous apportera des solutions sur de nombreux problèmes et sujets. Aussi pratique et didactique que le précédent, il se veut aussi plus accessible. Seul petit regret, l'absence de la sécurité.

AIR pour les développeurs Javascript

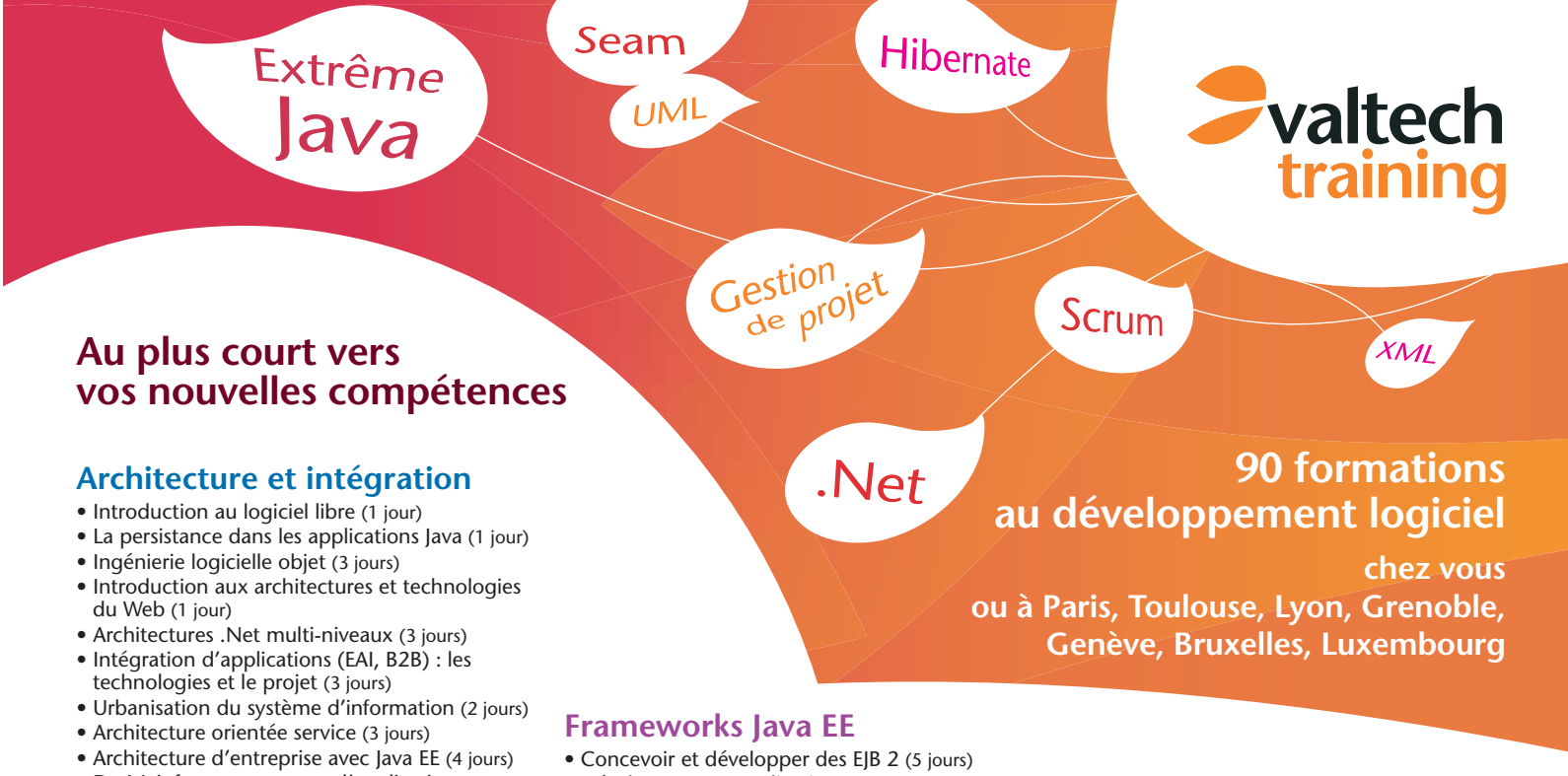
- ***Difficulté : ******
- ***Editeur : O'Reilly***
- ***Auteur : Laurent Debrauwer***
- ***Prix : 39 €***

Le design pattern est désormais une des meilleures pratiques, les plus utilisées par les développeurs. Ce livre s'adresse aux concepteurs et développeurs en Programmation Orientée Objet. Vous y trouverez 23 modèles (patterns) fondamentaux en les illustrant par des exemples pertinents et rapides à appréhender. Pour chaque pattern, l'auteur détaille son nom, le problème à résoudre, la solution qu'il apporte, ses domaines d'application et sa structure générique. Le livre s'accompagne d'un exemple concret décrit à la fois en UML et en Java, sous la forme d'un petit programme complet et exécutable. Pour mieux maîtriser ce livre, des connaissances en UML sont préférables

Dreamweaver CS3

- ***Difficulté : ******
- ***Editeur : Pearson***
- ***Auteur : Georges Brize***
- ***Prix : 42 €***

Vous ne savez pas quoi faire de votre été ? Voici un joli et beau pavé de 860 pages pour tout savoir de Dreamweaver CS3 (en attendant la CS4 dans quelques mois). Pas toujours évident à maîtriser, l'outil se révèle pourtant d'une richesse fonctionnelle incroyable. L'auteur se propose donc de vous faire faire le tour du propriétaire. On démarre par la prise en main de l'outil, l'espace de travail, les fonctions de base à maîtriser. C'est au bout de 270 pages que les choses réellement sérieuses débutent avec la gestion des documents, les CSS, les tableaux, les images, le multimédia, l'interactivité, Flash, les widgets, etc. Dommage que AIR et Flex soient absents, idem pour la mobilité. Heureusement, le framework Spry est lui bien présent (librairie Ajax). Une version d'évolution Mac et Windows de l'outil est incluse.



valtech training

Au plus court vers vos nouvelles compétences

90 formations au développement logiciel

chez vous
ou à Paris, Toulouse, Lyon, Grenoble, Genève, Bruxelles, Luxembourg

Architecture et intégration

- Introduction au logiciel libre (1 jour)
- La persistance dans les applications Java (1 jour)
- Ingénierie logicielle objet (3 jours)
- Introduction aux architectures et technologies du Web (1 jour)
- Architectures .Net multi-niveaux (3 jours)
- Intégration d'applications (EAI, B2B) : les technologies et le projet (3 jours)
- Urbanisation du système d'information (2 jours)
- Architecture orientée service (3 jours)
- Architecture d'entreprise avec Java EE (4 jours)
- Du Mainframe au serveur d'applications (1 jour)

Développement Java et C++

- Introduction technique à Java (1 jour)
- Programmer en utilisant les aspects et les Design Patterns (3 jours)
- Java et la conception objet (5 jours)
- Développement d'un client riche avec SWT et Eclipse RCP (3 jours)
- Atelier Java avancé (5 jours)
- Eclipse, créer son environnement de développement intégré (2 jours)
- Programmation intensive avec Java (5 jours)
- Extrême Java (4 jours)
- Développer une application Corba (4 jours)
- L'essentiel de C++ et la conception objet (5 jours)
- Programmation efficace et avancée en C++ (5 jours)

Microsoft .Net

- C# et la conception objet (5 jours)
- Programmation avec Visual Basic .Net et conception objet (5 jours)
- Programmation intensive avec le Framework .Net (5 jours)
- Développement d'applications Web avec ASP.NET (5 jours)
- Développement d'applications Windows Forms sur la plate-forme .Net (5 jours)
- Développer des applications avec C# et le Framework .Net 3.0 (5 jours)

Gestion de projet

- Gérer des projets avec un processus itératif (4 jours)
- Les méthodes agiles de développement logiciel (1 jour)
- Le Processus Unifié de développement logiciel (2 jours)
- Du recueil des besoins aux exigences : rédiger le cahier des charges (2 jours)
- Gestion de projet (3 jours)
- Manager des hommes dans le cadre d'un projet (2 jours)
- Management de projet (5 jours)
- MSProject (3 jours)
- Gérer les projets agiles avec Scrum (2 jours)
- Gérer les projets agiles avec XP (2 jours)

Frameworks Java EE

- Concevoir et développer des EJB 2 (5 jours)
- Développer une application Java EE avec les EJB 3 (5 jours)
- Gestion de la persistance avec Hibernate (3 jours)
- Mise en œuvre du Framework Seam (3 jours)
- Développement avec le Framework Spring (3 jours)
- Gestion avancée de la persistance avec Hibernate (2 jours)

Analyse, conception et modélisation avec UML

- Introduction technique à l'analyse, la conception et la programmation objet (1 jour)
- Introduction à UML (1 jour)
- Concevoir avec les Design Patterns (5 jours)
- La modélisation métier avec UML (3 jours)
- Analyse et conception avec UML (5 jours)
- La modélisation des systèmes complexes avec UML 2 et SysML (3 jours)
- La modélisation efficace des exigences avec les cas d'utilisation (2 jours)
- Analyse orientée objet avec UML (2 jours)
- D'UML 1 à UML 2 : quoi de neuf, docteur ? (1 jour)
- Modéliser les besoins et analyser avec UML (4 jours)

Oracle

- Introduction technique (1 jour)
- Exploitation (4 jours)
- SQL (3 jours)
- PL / SQL (3 jours)
- Optimisation des requêtes (2 jours)
- Administration (5 jours)
- Tuning (3 jours)

XML et Web Services

- Introduction à la technologie XML (1 jour)
- Introduction aux technologies Web Services (1 jour)
- Développer avec XML (3 jours)
- Développer une application XSL (2 jours)
- Développer des applications Web Services en Java (3 jours)
- Développer des applications XML avec Java (2 jours)

Développement Web

- Développement de pages Web avec HTML, CSS et JavaScript (3 jours)
- Développement, déploiement et administration d'applications Web (Java EE) avec WebSphere (3 jours)
- Développement d'applications Web avec PHP (3 jours)
- Ajax, pour dynamiser le poste client (2 jours)
- Hacking des applications Web (1 jour)
- Développer des applications avec Adobe Flex 3 (5 jours)
- Conception d'applications Web d'entreprise avec Java EE, les Servlets, JSP et Struts (5 jours)
- Développement d'applications Web avec Ruby on Rails (3 jours)
- Développement d'applications Web avec JavaServer Faces (3 jours)
- JavaServer Faces avancé (2 jours)
- Utilisation du Framework Struts pour le développement d'applications Web (3 jours)
- Développer une application Web avec Ajax et le Google Web Toolkit (3 jours)
- Développer des applications pour Adobe Integrated Runtime (2 jours)

Stratégies de développement logiciel

- Test Driven Requirement ou la gestion des exigences dirigée par les tests (2 jours)
- Test Driven Development ou la programmation pilotée par les tests en Java (3 jours)
- Stratégie de test, vérification et validation (3 jours)
- Les fondements de l'IT Infrastructure Library (ITIL) (3 jours)
- Introduction au CMMI (3 jours)
- L'usine logicielle, des concepts à la pratique (3 jours)

www.valtech-training.fr
+33 (0)1 41 88 23 00 - +33 (0)5 62 47 52 00
info@valtech-training.fr

Différences techniques entre ODF et OOXML

+ Tutoriel sur www.programmez.com/tutoriel.php

Pendant de nombreuses années, les éditeurs de logiciels ont produit des applications dédiées à la bureautique chacun de leur côté. Il était difficile pour l'utilisateur d'une application de manipuler avec fiabilité des documents créés par une autre application. Les spécifications techniques des formats de données étaient étroitement gardées secrètes. De ce fait, aucune compatibilité n'était possible à moins, pour les développeurs, de gaspiller un temps considérable à tenter de déchiffrer ces formats fermés.

Dans ce souci d'interopérabilité, il était nécessaire de créer une norme qui permettrait à un document d'être indépendant du logiciel utilisé pour le créer, le modifier, le lire ou l'imprimer.
Un point important, pour toute norme, est la facilité avec laquelle elle peut être implémentée. Une norme qui ne peut pas être facilement mise en œuvre la rend pratiquement inutile.
Open Document Format et Office Open XML sont deux formats de données ouverts et normalisés pour les applications de bureautique telles que traitements de texte, tableurs, présentations, diagrammes, dessins et bases de données. Comme l'objectif initial d'XML est de faciliter l'échange de contenus entre systèmes d'informations, ce langage s'est imposé de lui-même comme base de ces deux nouveaux formats de fichier. Cet article passe en revue les formats ODF et OOXML. Il examine la manière dont ces formats sont en fait constitués. Seules les questions techniques sont abordées; celles concernant la propriété intellectuelle ou les questions juridiques ne sont pas traitées.

Architecture des deux formats

Ces deux nouveaux formats utilisent des standards ouverts (ZIP, XML, PNG, etc.) Ce qui, pour les développeurs, n'est pas négligeable puisque tous ces standards sont supportés par la plupart des langages. Citons par exemple C/C++, .NET, PHP, Java, Python.
Tous les deux sont représentés sous la forme d'un fichier ZIP contenant un certain nombre de répertoires et de fichiers. Nous verrons un peu plus loin leur structure. Essayez de changer l'extension d'un fichier .docx, par exemple, en .zip puis ouvrez-le. Vous pourrez alors visualiser son contenu, extraire un fichier ou une image, comme vous le feriez habituellement avec une archive classique.
L'utilisation d'un fichier compressé permet, outre de réduire la taille occupée par le document, de regrouper l'ensemble des données au sein d'un même fichier afin de pouvoir s'échanger plus facilement et plus rapidement. Stocker les données de façon indépendante offre aussi l'avantage de protéger les documents plus efficacement contre la corruption de données. Si un élément est endommagé, alors les autres ne seront pas affectés.
Un dernier point commun entre les architectures de ces deux formats est qu'ils adoptent le principe de la séparation du contenu et de la présentation. Cependant, alors qu'ODF s'en contente, OOXML respecte l'Open Packaging Convention. Cette spécification, proposée par Microsoft, est également utilisée par le format XPS, concurrent du PDF. Elle permet de structurer un document de façon segmentée en séparant les données XML des données binaires. Ce système de stockage organise ainsi les données logiquement à l'aide de relations.

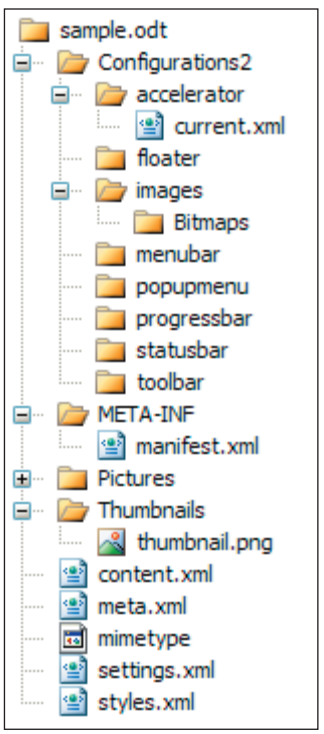


Fig. 1 Structure d'un fichier ODT

Description de la structure d'ODF

Quel que soit le type de document, le format ODF utilise la même structure. Cette structure est composée, au minimum, de 3 répertoires et 5 fichiers. À noter que des répertoires optionnels peuvent être créés suivant le besoin. Si, par exemple, une image est insérée dans le document, alors un répertoire Pictures, qui la contiendra, sera créé à la racine de ce document. Configurations2 regroupe des paramètres propres à l'interface utilisateur de l'application. Chaque sous-dossier stocke donc des informations liées aux menus, aux barres d'outils, aux raccourcis clavier, à la barre d'état, et autres. META-INF contient un fichier, manifest.xml, qui liste tous les éléments composant le document, de la même manière que le fichier équivalent dans les archives JAR.
Thumbnails contient un fichier, thumbnail.png, qui représente le rendu final du document sous la forme d'une vignette. Cette image est utilisée par le système d'exploitation pour l'aperçu rapide. On a donc :

- **content.xml** est le fichier principal; il décrit tout le contenu du document.
- **meta.xml** contient les métadonnées associées au document telles que le nom de l'auteur, la date de dernière révision, et d'autres détails ; elles permettent de gérer l'interopérabilité entre divers types de ressources informatiques.
- **mimetype** contient une seule ligne de texte qui précise simplement le type MIME du document.
- **settings.xml** contient des informations spécifiques à l'application. Ces réglages sont stockés lors de la sauvegarde du document.
- **styles.xml** contient toutes les définitions de mise en page appliquées aux objets décrits dans le fichier content.xml.

Description de la structure d'OOXML

Le format OOXML utilise une structure respectant l'OPC, cela implique une terminologie particulière. En effet, la notion de paquet définit l'ar-

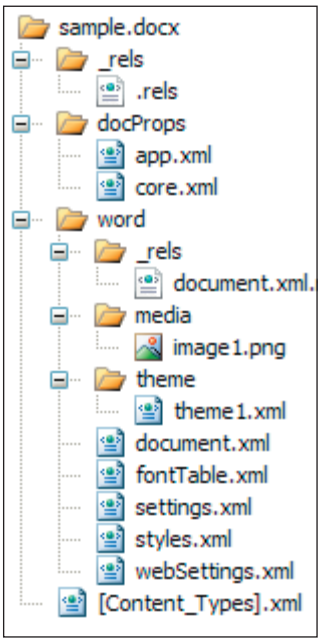


Fig. 2 Structure d'un fichier DOCX

chive ZIP en elle-même, c'est-à-dire le conteneur. Chaque fichier contenu dans un paquet est appelé partie. On distingue deux types de partie : les parties de contenu et les parties de relation.

Les parties de contenu contiennent les informations qui définissent les données et la sémantique du document. Ces informations peuvent aussi bien être du texte, du XML, que des données binaires telles que des images ou des objets OLE. Les parties de relation contiennent une structure XML définie dans les schémas de référence du standard ECMA-376.

Enfin, les relations sont le squelette du document; elles vont permettre de connaître précisément le rôle de chaque partie du paquet et, ainsi, de reconstituer le document dans son intégralité. Par convention, un fichier de relations se situe dans un répertoire _rels et porte l'extension .rels. On distingue, là aussi, deux types de relation : les relations de paquet et les relations de parties. Les relations de paquet spécifient le lien entre le paquet et une partie. Elles se situent logiquement à l'URI /_rels/.rels.

Les relations de parties lient une partie à une autre. Elles se situent donc dans un dossier rels placé à la racine de la partie source et portent le nom complet du fichier de la partie cible dont elles dépendent. Par exemple, le fichier de relations associé au contenu d'un document Word, autrement dit à la partie /word/document.xml, sera situé à l'URI /word/_rels/document.xml.rels. Le fichier [Content_Types].xml recense toutes les parties du paquet et associe à chacune d'elles un type MIME. Les données propres au document vont, quant à elles, être placées dans un répertoire spécifique à l'application utilisée. Il se nommera word pour un document texte, xl pour un document Excel, et ppt pour une présentation PowerPoint. Comme pour le format ODF, on va retrouver des fichiers XML dont document.xml, l'équivalent de content.xml, puis settings.xml et styles.xml. Par contre, leur syntaxe sera différente.

Type de document	OpenDocument Format		Office Open XML	
	Application	Extension	Application	Extensions (2007, 97-2003)
Texte formaté	Writer	.odt	Word	.docx, .doc
Modèle de texte formaté		.ott		.dotx, .dot
Tableur	Calc	.ods	Excel	.xlsx, .xls
Modèle de tableur		.ots		.xltx, .xlt
Présentation	Impress	.odp	PowerPoint	.pptx, .ppt
Modèle de présentation		.otp		.potx, .pot

Tab. 1 Principaux types de document ODF et OOXML

Comparaison des syntaxes XML

La philosophie se cachant derrière la façon dont a été pensé chacun de ces formats affecte grandement le résultat final. Une manière de le démontrer est de définir le contenu d'un document, de l'enregistrer dans les différents formats, puis d'examiner et comparer les parties appropriées des fichiers XML obtenus.

L'exemple consiste en un document d'une page contenant une phrase avec quelques effets sur la police. Comme le montrent les figures 3, on peut remarquer, sans trop de difficulté, que le rendu diffère largement entre les deux formats, ceci étant dû aux choix effectués lors des phases de conception.

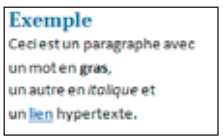


fig. 3 Aperçus de l'exemple en ODT et en DOCX

La représentation en XML du contenu du document selon le format ODF :

```
<text:h text:style-name="Heading_20_1" text:outline-level="1">
  Exemple
</text:h>
<text:p text:style-name="Standard">
  Ceci est un paragraphe avec un mot en
  <text:span text:style-name="T1">gras</text:span>
  , un autre en
  <text:span text:style-name="T3">italique</text:span>
  et un
  <text:a xlink:type="simple" xlink:href="http://www.example.com/">
    <text:span text:style-name="T5">lien</text:span>
  </text:a>
  hypertexte.
</text:p>
```

Et, l'équivalent suivant le format OOXML :

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="Titre1"/>
  </w:pPr>
  <w:r>
    <w:t>Exemple</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t>Ceci est un paragraphe avec un mot en </w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:b/>
    </w:rPr>
    <w:t>gras</w:t>
  </w:r>
  <w:r>
    <w:t>, un autre en </w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:i/>
    </w:rPr>
    <w:t>italique</w:t>
  </w:r>
  <w:r>
    <w:t>et un </w:t>
  </w:r>
```

```
    </w:r>
    <w:hyperlink r:id="rId4" w:history="1">
      <w:r>
        <w:rPr>
          <w:rStyle w:val="Lienhypertexte"/>
        </w:rPr>
        <w:t>lien</w:t>
      </w:r>
    </w:hyperlink>
    <w:r>
      <w:t> hypertexte.</w:t>
    </w:r>
</w:p>
```

Balises et nommage

Pour commencer l'analyse de ces portions de code XML, comparons les balises et leur nommage. OOXML utilise des balises au nom plus court qu'ODF. Les avantages sont, tout d'abord, une économie de l'espace utilisé par les fichiers et, d'autre part, une augmentation de la vitesse d'analyse des données lors de leur conversion en structures internes à l'application. Toutefois, un plus grand nombre de balises est alors nécessaire dans ce format. Le nommage par ODF est donc plus long et plus verbeux, mais il est conforme à la convention d'attribution des noms de balise du XML. L'avantage est qu'il facilite l'interopérabilité dans la mise en œuvre de la norme. L'espace utilisé et les performances de l'analyse des données augmentent, mais sont compensés par le fait que ce format nécessite moins de balises. Cela dit, ces différences ont peu d'importance puisque l'algorithme de compression utilisé par chaque format réduit les balises à une taille quasiment équivalente. Et, sur les machines actuelles, le temps passé à analyser les données est insignifiant par rapport au temps d'accès au disque dur. La lisibilité et la clarté des noms de balises influencent l'adoption d'un format en particulier. Le coût d'implémentation d'une norme comme celle-ci est directement lié aux ressources nécessaires à son développement. Il y a des centaines de balises dans chaque format et moins elles sont intuitives, plus il est difficile pour un développeur de les comprendre et de les mémoriser.

Méthode de "spanning"

Cette méthode consiste à atteler des éléments en les intégrant directement à l'intérieur d'un autre. Pour illustrer ce concept et tenter de le rendre plus clair, prenons l'exemple d'une phrase dont chaque mot subirait une mise en forme particulière. De ce fait, chaque mot serait encadré par une balise word dont les attributs définiraient le style. En suivant cette méthode, chaque mot sera situé à l'intérieur d'une même balise mère, sentence. Alors que, dans le cas contraire, une nouvelle balise sentence sera créée pour chaque mot. Contrairement à ODF, comme le prouve implicitement la balise <text:span>, OOXML ne prend pas en charge cette fonctionnalité. Au début d'un changement de format, OOXML doit fermer le bloc parent et en créer un nouveau pour englober ce changement. On peut le voir en remarquant la prolifération des balises <w:r> et <w:t>. Ce qui, malheureusement, alourdit le code XML et augmente considérablement sa complexité.

Utilisation des normes

Un dernier point, non illustré dans cet exemple, concerne les images insérées dans ce type de document. Sachez simplement qu'ODF a été conçu pour utiliser en priorité, et dans la mesure du possible, des normes déjà existantes. C'est le cas de la balise <draw:frame>, chargée de dessiner une image au sein du document, qui se base sur les normes XLink, parfois aussi appelé XLL, et SVG.

OOXML, quant à lui, semble ne suivre qu'une seule norme, la mise en forme XML elle-même. De même, certains aspects d'OOXML peuvent être extrêmement difficiles à percevoir et à comprendre. C'était, par exemple, le cas des formules définies par l'attribut *eqn* des balises <v:f> dans la version 2003 de Microsoft Office. Depuis, Le format a été amélioré, simplifié notamment par l'arrivée de la balise <v:imagedata>, mais l'insertion d'un graphique implique beaucoup de balises.

Générer un fichier DOCX en PHP

Après avoir présenté la structure d'un document OOXML, nous allons maintenant voir comment la manipuler par la programmation.

Le code source associé à cet article propose un framework rudimentaire et très simpliste. Il va de soi qu'il n'a pas pour vocation d'être utilisé en production mais, simplement, d'illustrer les concepts abordés précédemment. Dans ce cas, il serait plutôt conseillé d'étoffer le code, de le sécuriser, et d'utiliser un modèle, directement issu d'Office, dans lequel il ne suffirait plus que de modifier quelques champs prédéfinis, par exemple. L'objectif de cet exemple est de montrer la facilité avec laquelle il est possible, en PHP, de créer un document minimal "from scratch", c'est-à-dire à partir de rien et contenant les fichiers nécessaires au minimum à sa bonne interprétation par Word. Nous obtiendrons ainsi un document Word comportant le fameux "Hello Office Open XML!".

Pour ce faire, nous allons commencer par implémenter une classe, OpenXML, qui va permettre, au travers d'une API simple, de définir le corps du document, son texte, et de créer chacun des fichiers de manière tout à fait transparente pour l'utilisateur afin que la complexité de la structure interne du paquet lui soit masquée.

Voici un aperçu de l'API de la classe OpenXml modélisée par ce diagramme de classe :

OpenXml
+ object OpenXML (string $filename)
+ string FromHtml (string $text)
+ bool CreateFolder ( )
+ string body

Ensuite, pour plus de clarté, définissons deux variables :

```
$filename="test.docx";
$content="<p>Hello Office Open XML!</p>";
```

$filename qui correspond au nom de fichier du document, et, $content qui correspond au texte qui y sera inséré.

1. Création du dossier racine

La première étape consiste en la création du répertoire dans lequel chaque partie sera stockée.

C'est ici que nous allons utiliser notre API. Pour cela, n'oublions pas d'inclure le fichier.

```
require("lib.ooxml.php");
```

Ensuite, créons l'instance de notre document.

```
$doc=new OpenXml($filename);
```

La première fabrique logicielle entièrement pilotée par les modèles

CodeFluent est un produit de génie logiciel qui permet
d'industrialiser la fabrication d'applications professionnelles manipulant des données sur la plate-forme .NET
en automatisant la création des composants à partir d'une modélisation de votre métier.

CODEFLUENT PROCURE LES BÉNÉFICES SUIVANTS

UN GAIN TRÈS SIGNIFICATIF DE PRODUCTIVITÉ

par l'automatisation des tâches répétitives et la mise en œuvre du modèle métier selon une architecture orientée services.

UNE MAINTENABILITÉ ET UNE QUALITÉ ACCRUES

grâce à l'approche intrinsèque de génération qui évite les erreurs et permet d'effectuer des mises à jour sur toutes les couches à l'aide d'une modification centralisée.

LA LIMITATION DU RISQUE

par la structuration du travail des développeurs autour d'une modélisation objet évolutive qui garantit une mise en œuvre selon les meilleures pratiques d'implémentation SOA sous plate-forme Microsoft.

L'ÉVOLUTIVITÉ DE L'APPLICATION

car la prise en charge de nouvelles technologies et de nouvelles versions de plate-forme est assurée par la mise à disposition régulière de nouveaux producteurs (Windows Presentation Foundation, Office Business Applications, mobilité).

LES OFFRES CODEFLUENT

Pour commander votre licence, envoyez un email à **sales@softfluent.com**

CodeFluent est un produit de la société **SoftFluent** | Ecrivez-nous à : **info@softfluent.com**
Inscrivez-vous gratuitement sur **www.codefluent.com** pour recevoir les informations à jour

```
$doc->body=$doc->FromHtml($content);
if(!$doc->CreateFolder()) { exit(); }
```

À partir de cette instance, nous allons demander à transformer le code HTML en texte simple pour définir le corps du document puis invoquer la création des fichiers. Le résultat obtenu à ce moment de l'exécution du script est représenté par la figure 3. On retrouve bien la structure minimale d'un document OOXML.

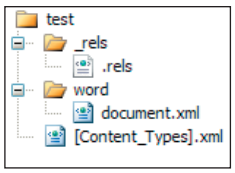


Fig. 4 Structure de notre test

2. Création de l'archive

La seconde étape consiste en la compression des fichiers ainsi générés. Pour cela, à la manière de notre classe OpenXml, nous allons nous simplifier la tâche en utilisant la librairie PclZip. PhpConcept Library Zip est une librairie de compression permettant de gérer des archives ZIP. N'oublions pas, ici non plus, d'inclure le fichier.

```
require("lib.pclzip.php");
```

À partir d'une instance de cette classe, représentant notre document final, nous allons lire le contenu du répertoire généré à l'étape précédente afin d'y inclure chaque fichier.

```
$zip=new PclZip($filename);
$dir=opendir($doc->folder."/");
while($f=readdir($dir)) {
   if(($f != ".")
   && ($f != ".."))
   {
      $zip->add($doc->folder."/".$f, PCLZIP_OPT_REMOVE_ALL_PATH);
   }
}
closedir($dir);
```

À ce moment de l'exécution de notre script, l'archive a été créée dans le dossier des fichiers temporaires du serveur web.

3. Téléchargement du fichier

Accessoirement, nous allons, ici, récupérer notre document en le téléchargeant par l'intermédiaire de notre navigateur Internet.
Pour cela, rien de plus simple que de modifier les données d'en-tête de la page web résultant de l'exécution du script PHP en se référant à la RFC 2616.

```
header("Content-Type: application/octet-stream");
header('Content-Disposition: attachment; filename="'.$filename. '"');
flush();
readfile($filename);
```

4. Nettoyage des fichiers

Enfin, dans un souci de propreté, nous allons pouvoir supprimer les fichiers et dossiers créés précédemment pour le besoin et libérer ainsi un peu de place sur le disque dur du serveur web.
Vous pourrez remarquer l'utilisation de la fonction cleandir développée spécialement pour contourner un manque de PHP concernant la suppression de répertoires non vides.

```
cleandir($doc->folder."/");
unlink($filename);
```

Finalement, une fenêtre pop-up nous proposera alors d'ouvrir notre document ou de l'enregistrer sur un support quelconque.
Cet exemple a été testé avec succès sous Firefox 2.0.0.14 et Internet Explorer 7.0 sur un serveur web Apache 2.2.6 avec PHP 5.2.5 et la librairie PclZip 2.5.

D'autres langages, d'autres SDK

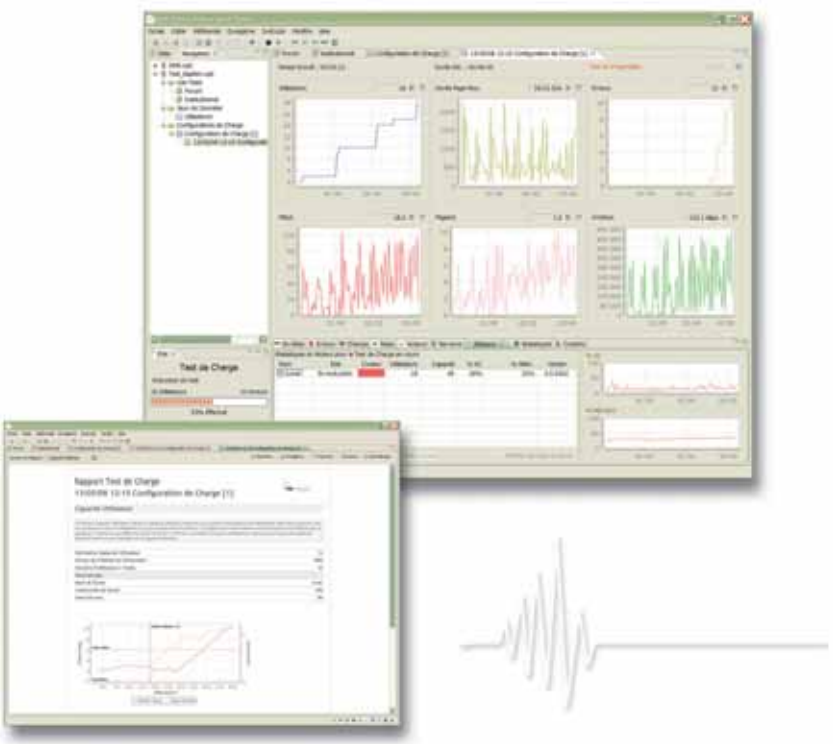
Bien entendu il existe une multitude d'outils de développement, même de haut niveau, pour manipuler de tels formats. Pour le format ODF, l'outil le plus complet est UNO, pour Universal Network Objects. Plus qu'une simple interface de programmation, UNO est le composant central de la suite bureautique OpenOffice.org. Il supporte nativement les langages C/C++, Python et Java. Des adaptations pour d'autres langages, comme Perl, Tcl et Delphi, sont disponibles ou prévues, mais avec des possibilités plus limitées. En Perl, Open Office Document Connector suit le même principe qu'UNO et OOBuilder permet de créer rapidement des documents. L'OpenDocument Fellowship, à l'origine du projet LibOpenDocument, initialement en Python, tient à jour une liste de logiciels supportant le format ODF. Pour le format OOXML, inutile de préciser que le framework .NET 3 le gère parfaitement, grâce aux classes présentes dans l'espace de nom System.IO.Packaging. En dehors du .NET, les langages les plus utilisés pour manipuler OOXML sont Java et PHP. Office ne supporte pas nativement le format ODF, mais Microsoft propose un plug-in très facilement intégrable et permettant la conversion de documents.

+ Tutoriel sur
www.programmez.com/tutoriel.php

Jean-Guillaume L'Hélias
Étudiant en ingénierie informatique - Supinfo

webperformance SUITE

Le Meilleur du Test de Performance Web en toute simplicité et au meilleur rapport qualité-prix



- **Développement automatisé de Cas-Tests et Test de Charge**
- **Rapports d'analyse des Tests de performance et de charge**
- **Analyse des Serveurs et de l'impact sur les performances**
- **nouveau Interface utilisateur et Rapports en Français**

Découvrez comment Web Performance Suite peut faire gagner vos applications Web en Qualité, Fiabilité et Performance

Evaluation gratuite de Web Performance Suite sur http://www.kapitec.com/Pub/WP?id=110

Bénéficiez de notre assistance technique pendant les 15 jours d'évaluation.

kapitec Software

Power Your Web Projects

Tél.: 05 34 27 90 03
sales@kapitec.com

La sérialisation XML

2e partie

Nous avions commencé notre plongée dans la sérialisation avec Programmez ! 109. Pour rappel, nous avions vu comment sérialiser une classe au format XML (avec un XmlSerializer et un SoapFormatter). Il s'agit de regarder maintenant quelles possibilités sont proposées par le framework .NET pour personnaliser cette sérialisation.

Dans le framework .NET, tous les objets ne sont pas sérialisables avec le type de sérialiser XML. Il s'agit par exemple d'objets ne possédant pas de constructeur vide, comme c'est le cas de System.Drawing.Font ou encore d'objets n'exposant pas de membre public, par exemple System.Drawing.Color. Pour s'en convaincre, il suffit de prendre le premier exemple de l'article précédent qui tente de sauver un objet de type Color. Le résultat obtenu est un tag vide dans le fichier XML, car Color n'expose aucun membre public. Il existe plusieurs manières de personnaliser la sérialisation : on peut utiliser les attributs, pour changer le comportement par défaut ou alors implémenter soit même un Serialiser en implémentant l'interface IXmlSerialisable pour créer sa propre sérialisation.

Utilisation des attributs

Il existe plusieurs attributs que l'ont peut placer sur les membres d'une classe (sur la déclaration de la classe, sur les propriétés, sur les énumérations, etc.) pour qu'ils soient traités de manière spéciale durant le processus de sérialisation. Il serait bien trop long de montrer l'usage de tous les attributs disponibles dans le framework. Nous avons donc fait le tri et choisi ceux que nous avons trouvé les plus utiles/pratiques.
Petite remarque au passage : les classes étant des attributs se terminent par convention par 'Attribute'. Lors de l'utilisation de ces classes, cette terminaison est facultative. Ainsi, XmlElementAttribute peut s'écrire plus simplement XmlElement. Voici un petit tableau qui explique brièvement l'utilité des attributs les plus utilisés/les plus pratiques.

Nom de l'attribut	Description
XmlAttribute	Sera représenté comme un attribut. On peut lui passer des paramètres, notamment pour changer le nom de l'attribut. Par exemple XmlAttribute ("MyAttribute"). Utilisation : principalement sur les attributs et les propriétés.
XmlElement	Sera représenté comme un élément. On peut lui passer des paramètres, notamment pour changer le nom de l'élément. Par exemple XmlElement ("MyElement"). Utilisation : principalement sur les attributs et les propriétés.
XmlText	Sera représenté comme un texte dans un nœud. Typiquement <MyNode>Xml Text</MyNode>. Utilisation : principalement sur les attributs et les propriétés.
XmlIgnore	Ne sera pas représenté dans le fichier XML. Utilisation : principalement sur les attributs et les propriétés.
XmlEnum	Sera représenté comme une énumération. On peut lui passer des paramètres, notamment pour changer le nom de l'élément. Par exemple XmlEnum("ID"). Utilisation : sur les attributs.
XmlRoot	Permet de définir l'élément racine du document XML. Utilisation : sur les interfaces, structures, énumérations et classes.
XmlArray	S'applique sur une collection. Utilisation : principalement sur les attributs et les propriétés.
XmlArrayItem	S'applique sur les éléments d'une collection. Utilisation : principalement sur les attributs et les propriétés.

Reprenons le code de l'exemple vu dans l'article précédent et ajoutons quelques attributs et propriétés pour tester la personnalisation XML à l'aide des attributs décrits ci-dessus. Voici la nouvelle classe Person que nous obtenons

```
[Serializable] // Requis pour la sérialisation SOAP
public class Person
{
    protected int _age = 0;
    protected Sex _sex = Sex.U;

    // Constructeur vide, requis par la sérialisation XML
    public Person() { }

    public Person(int age, Sex sex)
    {
        this._age = age;
        this._sex = sex;
    }

    // Les propriétés publiques DOIVENT être en lecture et
    // écriture. Si tel n'est pas le cas, une exception sera
    // levée lors de la sérialisation !
    // Sera un élément appelé 'Nom' d'une personne
    [XmlElement("Nom")]
    public string Name { get; set; }

    // Sans spécification supplémentaire, une propriété est transformée
    // en élément qui a comme nom le nom de la propriété, ici Sexe.
    public Sex Sexe
    {
        get { return this._sex; }
        set { this._sex = value; }
    }

    // Evite d'avoir un élément vide puisque cette structure ne possède
    // pas de membres publics. On indique au sérialiser d'omettre cette
    // propriété.
    [XmlIgnore()]
    public Color PreferredColor { get; set; }
}
```

Rajoutons l'énumération qui nous permettra de tester l'attribut *XmlEnum*

```
public enum Sex
{
    [XmlEnum(Name = "Men")]
    M,
    [XmlEnum(Name = "Women")]
    F,
    [XmlEnum(Name = "Unknow")]
    U
```

```
    }
```

Pour complexifier un peu, nous faisons une dernière classe qui dérive de Person, un étudiant

```
public class Student : Person
{
    public Student() { }

    public Student(int age, Sex sex, double avgGrade) : base(age, sex)
    {
        this.AverageGrade = avgGrade;
    }

    // Sera un attribut appelé 'NoteMoyenne' d'un étudiant
    [XmlAttribute("NoteMoyenne")]
    public double AverageGrade { get; set; }

    // Sera un text au niveau de l'élément Student
    [XmlText()]
    public string Comment { get; set; }
}
```

Il ne nous reste à présent plus qu'à faire une classe qui utilise la structure que l'ont vient de mettre en place. Disons une classe Conference qui contiendra tout simplement une collection de personne.

```
// Change le nom de l'élément racine
[XmlRoot("MyConference")]
public class Conference
{
    /// Indique quels objets vont se trouver dans la collection.
    [XmlArrayItem(typeof(Person)), XmlArrayItem(typeof(Student))]
            /// Définit le nom de la balise qui englobera la collection.
    /// Si la collection est null, cet élément sera omis).
    [XmlArray("Persons", IsNullable = false)]
    public List<Person> person = null;
}
```

Voilà, nous avons terminé !
Nous remplissons et sérialisons maintenant notre objet Conference pour voir la sortie qui va être produite. Voici le code utilisé

```
var conf = new Conference()
{    person = new List<Person>()
    {   new Person(24, Sex.M)
            { Name = "Bidou", PreferredColor = Color.Blue },
        new Person
            { Name = "Jean" },
        new Student(22, Sex.F, 18.5d)
            { Name = "Jessica", PreferredColor = Color.Pink,
              Comment = "Etudiante en psychologie" }
    }
};

using (var xWriter = XmlWriter.Create(path, new XmlWriterSettings()
    { Encoding = Encoding.UTF8 }))
{
    var xSer = new XmlSerializer(typeof(Conference));
     xSer.Serialize(xWriter, conf);
}
```

Et voici la sortie

```
<?xml version="1.0" encoding="utf-8" ?>
<MyConference xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <Persons>
  - <Person>
      <Nom>Bidou</Nom>
      <Sexe>Men</Sexe>
    </Person>
  - <Person>
      <Nom>Jean</Nom>
      <Sexe>Unknow</Sexe>
    </Person>
  - <Student NoteMoyenne="18.5">
      <Nom>Jessica</Nom>
      <Sexe>Women</Sexe>
      Etudiante en psychologie
    </Student>
  </Persons>
</MyConference>
```

Le fichier XML obtenu est bien celui souhaité :

- Le membre privé 'age' n'a pas été sérialisé
- La propriété publique PreferredColor n'a pas été sérialisée
- La propriété Name est devenue un élément avec comme nom " Nom "
- Sexe est devenu une propriété car nous n'avons donné aucune indication supplémentaire
- AverageGrade a été transformé en attribut avec pour nom " NoteMoyenne "
- Le commentaire est devenu un texte au niveau du Student
- Les valeurs de l'énumération Sex ont été remplacées par celle que nous avons spécifiée dans l'attribut XmlEnum
- L'élément racine de notre fichier s'appelle MyConference, comme nous l'avons indiqué avec XmlRoot
- Notre collection de Person s'appelle " Persons " comme nous l'avons spécifié avec XmlArray

Nous venons de voir comment personnaliser la sérialisation XML à l'aide des attributs. Cette méthode peut s'avérer pratique si la personnalisation n'est pas trop poussée. Dans le cas contraire, il faudra passer par l'interface *IXmlSerialisable* qui permet de personnaliser totalement la sérialisation mais qui demande un travail (beaucoup) plus grand, comme nous allons le voir à présent.

Utilisation de l'interface IXmlSerialisable

Cette interface permet donc la mise en forme personnalisée pour la sérialisation et la désérialisation. Elle expose 3 méthodes :

- GetSchema

Cette méthode devrait retourner null. Voici ce que MSDN indique :
" Lors de la sérialisation ou de la désérialisation d'un objet, la classe XmlSerializer n'effectue pas de validation XML. C'est la raison pour laquelle il est généralement plus sûr d'omettre les informations de schéma en fournissant une implémentation simple de cette méthode, par exemple en retournant null."

- ReadXml

C'est la méthode qui sera appelée par la méthode Déserialize du sérialiser.

- WriteXml

C'est la méthode qui sera appelée par la méthode Sérialize du sérialiser. Les méthodes ReadXml et WriteXml prennent en paramètre un objet de type XmlReader ou XmlWriter qui permettent de facilement lire ou écrire du XML. Créons donc une classe qui implémente IXmlSerialisable et remplissons les trois méthodes de l'interface.

```
public class Person : IXmlSerializable
{
    protected Sex _sex = Sex.U;

    // Constructeur vide, requis par la sérialisation XML
    public Person() { }
    // Comme avant
    public string Name { get; set; }
    // Comme avant
    public Sex Sex
    {
        get { return this._sex; }
        set { this._sex = value; }
    }

    // Retourne null comme le spécifie MSDN
    public XmlSchema GetSchema()
    {
        return null;
    }

    // Lecture des valeurs
    public void ReadXml(XmlReader reader)
    {
        reader.MoveToAttribute("Name");
        this.Name = reader.Value;
        reader.Read();
        if (reader.IsStartElement("PersonalInfo"))
        {
            reader.ReadStartElement("PersonalInfo");
            this._sex = (Sex)(Enum.Parse(typeof(Sex), reader.ReadString()));
            reader.ReadEndElement();
        }
    }

    // Ecriture des valeurs
    public void WriteXml(XmlWriter writer)
    {
        writer.WriteAttributeString("Name", this.Name);
        writer.WriteStartElement("PersonalInfo");
        writer.WriteElementString("Sex", this._sex.ToString());
        writer.WriteEndElement();
    }
}
```

Puisque nous faisons une sérialisation manuelle, il nous faut encore indiquer que le header (< ?xml version ….) doit être écrit. Evidemment, cette action doit se faire tout au début. Nous corrigeons donc notre méthode de sérialisation en faisant appel à la méthode *WriteStartDocument* qui s'occupera de générer le début du document. Voici le nouveau code pour l'écriture du fichier

```
using (var xWriter = XmlWriter.Create(path, new XmlWriterSettings()
    { Encoding = Encoding.UTF8 }))
{
    var xSer = new XmlSerializer(typeof(Person));
    xWriter.WriteStartDocument(true);
    xSer.Serialize(xWriter, pers);
}
```

Nous obtenons le fichier XML souhaité

```
<?xml version="1.0" encoding="utf-8" ?>
<Person Name="Bidou">
- <PersonalInfo>
    <Sex>M</Sex>
  </PersonalInfo>
</Person>
```

La lecture du fichier XML se fera sans modification du code utilisé dans les exemples précédents (en appelant la méthode Deserialize)

Conclusion

Aujourd'hui incontournable, la sérialisation XML est très simple à mettre en place puisque sans personnalisation supplémentaire, elle nécessite très peu de ligne de code. Les attributs nous offrent un peu de souplesse pour changer le comportement par défaut du serialiser, mais une sérialisation totalement customisée devra nécessairement faire appel à l'interface IXmlSerialisable. Cette dernière solution peu s'avérer un peu plus coûteuse en temps, lorsqu'il s'agit de programmer manuellement la sérialisation d'objets complexes.

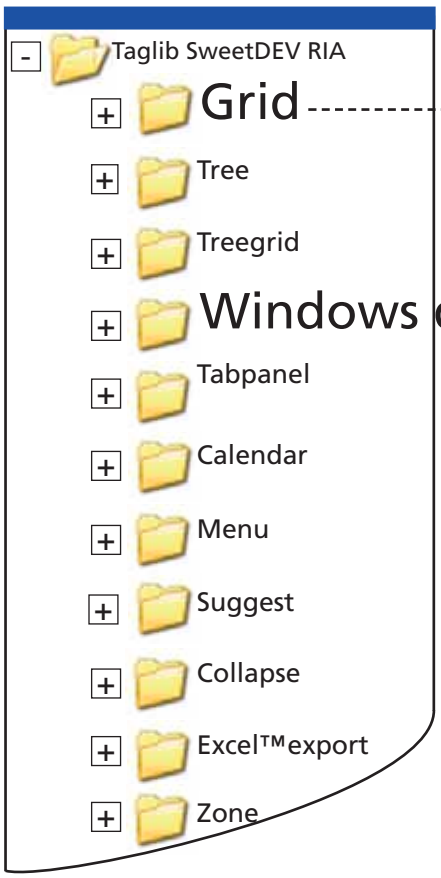
Nicolas Biedermann - *MVP C#*

ideo TECHNOLOGIES

SweetDEV RIA

Piochez dans nos tags Ajax pour développer vos interfaces Web 2.0 !

SweetDEV RIA™ est une **bibliothèque de composants Ajax, prêts à l'emploi**, pour le développement d'applications Web "riches".



Glisser/déplacer les colonnes, tri multi-colonnes, cases à cocher, pagination, scrolling horizontal, contenus multiples dans les cellules, gestion de préférences...

Gérer des composants "Window" au sein d'une structure "Layout" pour définir des IHM comparables à celles de Netvibes™ ou iGoogle™.

Pourquoi choisir SweetDEV RIA ?

- **La simplicité d'utilisation**
- **Le support technique en France**
- **Le développement de composants sur mesure**
- **Une expertise en architecture et ergonomie**
- **Des formations**
- **Un projet Open Source**

JSP | **Ajax** | **Web 2.0** | **J2EE** | **Licence Apache 2**

Téléchargez la dernière version
www.ideotechnologies.com

sweet DEV RIA

Rails et les tests

Rails est qualifié de framework pour développement web agile. Les méthodes agiles, conçues pour faciliter l'adaptation aux changements, ne se conçoivent pas sans tests.

En effet, chaque modification du code peut entraîner un changement inattendu du comportement du programme. Les tests sont un moyen extrêmement efficace de garder son code sous contrôle afin d'éviter les bugs inattendus. Allant même plus loin, les méthodes de développement piloté par tests (TDD – Test driven development) préconisent l'écriture des tests avant l'écriture du code. Cette technique force le développeur à penser son programme avant de l'implémenter. Il subsiste cependant un problème : tester chaque ligne de code prend du temps, beaucoup de temps. Il n'est pas rare de voir des applications contenant plus de lignes de tests que de lignes de code. C'est dans le but d'accélérer la création de tests que de nombreux frameworks ont vu le jour. Rails intègre son propre framework de tests qui, comme nous allons le voir, peut aider à améliorer grandement la productivité.

Premier test avec Rails

Commençons par créer une application rails qui gérera la création d'articles pour Programmez.

```
Dans la console

rails programmez
cd programmez
```

Ensuite, créons un échafaudage (scaffold) pour la ressource Article. Un Article aura simplement un titre (title) et un contenu (content).

```
Dans la console

ruby script/generate scaffold Article title:string content:text
```

Cette commande génère non seulement les fichiers du modèle, du contrôleur et des vues correspondants à Article, mais également les fichiers destinés à contenir les tests relatifs à la ressource Article. Le dossier test de l'application Rails contient 5 sous-dossiers. Chacun de ceux-ci contient le code relatif à une partie différente des tests.

- unit : les tests des modèles de l'application
- functional : les tests des contrôleurs
- integration : les tests permettant de tester des scénarios d'utilisation
- fixtures : les exemples de données
- mocks : les simulacres d'objets (nous ne les aborderons pas dans cet article)

Dans un premier temps, nous allons nous intéresser au dossier unit. Ce dernier contient le fichier article_test.rb qui, comme son emplacement et son nom l'indiquent, contient les tests d'unité du modèle Article.

```
Fichier test/unit/article_test.rb

require File.dirname(__FILE__) + '/../test_helper'

class ArticleTest < ActiveSupport::TestCase
  # Remplacez ceci avec vos propres tests
  def test_truth
    assert true
  end
end
```

Le premier test que nous allons réaliser consiste à vérifier qu'un article peut bien être créé dans la base de données. Étant donné que le générateur Rails a lui-même créé le fichier du modèle que nous testons, nous nous attendons à ce que le test passe. Ajoutons donc notre méthode de test test_should_create_record à notre classe ArticleTest.

```
Fichier test/unit/article_test.rb

def test_should_create_record
  article = Article.create
  assert article.valid?, "Article n'est pas valide"
end
```

Avant de lancer notre test, nous devons créer la table de la base de données qui contiendra nos articles. Encore une fois, Rails a déjà fait le travail pour nous. Il suffit de lancer la commande :

```
Dans la console

# Rails 2.0.2, crée automatiquement la base de données sqlite3
rake db:migrate
```

Nous pouvons ensuite lancer les tests du modèle Article :

```
Dans la console

rake test:units

...(liste des fichiers utilisés)...
Started
..
Finished in 0.43 seconds.
2 tests, 2 assertions, 0 failures, 0 errors
```

Comme attendu, nos deux tests se sont déroulés sans encombre.

Tester avant de coder

Maintenant que nous avons vu comment fonctionnaient les tests unitaires, nous allons mettre en pratique le développement piloté par tests en testant une fonctionnalité de notre programme qui n'est pas encore codée. Ajoutons une nouvelle règle à la logique métier de notre application: "nous ne désirons pas qu'un article soit enregistré dans la base de données s'il ne possède pas de titre". Ensuite, exprimons cela sous forme de test et exécutons-le.

```
Fichier test/unit/article_test.rb

def test_should_not_be_valid_if_no_title
  article = Article.create
  assert !article.valid?, "Article est valide alors qu'il ne devrait pas"
```

```
end
Dans la console

rake test:units

...(liste des fichiers utilisés)...
Started
..
Finished in 0.501 seconds.

1) Failure:

Article est valide alors qu'il ne devrait pas.
<false> is not true.

3 tests, 3 assertions, 1 failure, 0 error
```

Comme attendu, le test ne passe pas car nous n'avons pas encore ajouté la logique métier correspondant au cas testé. Pour valider la présence d'un attribut, nous allons utiliser la validation validate_presence_of présente dans ActiveRecord :

```
Dans le fichier app/models/article.rb

  class Article < ActiveRecord::Base
    validates_presence_of :title
  end
```

Nous relançons la suite de tests et... toujours une erreur. Mais ce n'est plus la même. Le test test_should_create_record précédemment écrit a joué son rôle. Comme la logique métier a changé, le test n'est plus correct. En effet, il n'est plus possible de créer un enregistrement de Article s'il n'a pas de titre. Nous modifions donc le test test_should_ create_record pour qu'il reflète la logique métier.

```
Fichier test/unit/article_test.rb

def test_should_create_record
  article = Article.create(:title=>"Les tests dans Rails")
  assert article.valid?, "Article n'est pas valide"
end
```

Nous lançons la suite de tests une fois de plus. Cette fois, ils passent sans encombre. Nous pouvons être satisfaits et continuer à coder tout en sachant que si une règle métier est enfreinte, nous serons immédiatement avertis par les tests.

Se faciliter la vie... toujours plus...

Comme si l'écriture de tests n'était pas encore suffisamment facilitée, le framework de tests inclus dans Rails propose une série de raccourcis pour améliorer l'expressivité du code de tests et réduire sa taille.

Assertions

La méthode *assert* que nous avons utilisée pour notre premier test nous a permis de tester la vérité de notre assertion. Cette méthode est le coeur du processus de test. C'est en effet la méthode assert qui détermine si un test est valide ou non. Cependant, elle est limitée car elle ne permet que de tester la vérité d'une expression. Quand les conditions à tester se complexifient, les expressions vérifiées par assert peuvent devenir longues et peu lisibles. Pour éviter cela, une série d'autres assertions ont été définies.

Assertion	Est vérifiée si...
assert(expression, [msg])	expression est true
assert_equal(obj1, obj2, [msg])	obj1 == obj2
assert_same(obj1, obj2, [msg])	obj1.equal? obj2
assert_nil(obj, [msg])	obj.nil?
assert_match(regexp, chaine, [msg])	la chaine correspond à regexp
assert_in_delta(espéré, réel, delta, [msg])	les nombres réel et espéré sont dans un delta l'un de l'autre
assert(_nothing)_raise (exceptions){ block }	si une exception contenue dans la liste exceptions est rencontrée dans le bloc
assert_instance_of(class, obj, [msg])	obj est une instance de class
assert_kind_of(class, obj, [msg])	obj descend de class
assert_respond_to(obj, :method, [msg])	obj à une méthode :method
flunk([msg])	jamais vérifié : permet de signaler qu'un test n'est pas encore écrit

Dans notre exemple, nous pourrions remplacer l'instruction assert article.valid? par assert_valid article. Bien qu'ici, l'économie de caractères soit négligeable, nous gagnons en expressivité avec la seconde écriture. Les assertions sont disponibles dans plusieurs librairies. Les assertions de base reprises dans le tableau proviennent du framework Test::Unit[1]. De plus, il existe également des assertions spécifiques à Rails [2]. La boîte à outils ZenTest en propose encore d'autres [3].

Assistants

Bien souvent, les cas de tests sont spécifiques à notre logique métier et nous ne trouvons pas d'assertion correspondante. Nous devons alors faire appel à des assistants (helpers) tout comme nous le ferions pour les vues. Les assistants de tests sont placés dans le fichier test/test_helper.rb. La méthode assert est utilisée pour vérifier qu'une expression est vraie. Dans notre test test_should_not_be_valid_ if_no_title, nous désirons vérifier que l'expression article.valid? est fausse. Nous allons donc définir une méthode deny qui va remplir ce rôle.
Fichier test/test_helper.rb

```
  def deny(condition, msg = nil)
    assert !condition, msg
  end
```

Nous pouvons dès lors réécrire notre test :

```
Fichier test/unit/article_test.rb

  def test_should_not_be_valid_if_no_title
    @article = Article.create
    deny @article.valid?, "Article est valide alors qu'il ne devrait pas"
  end
```

Méthodes spéciales

Pour les deux tests réalisés dans notre exemple, nous avons chaque fois commencé par créer une instance de la classe Article. Il y a fort à parier que, comme nous testons Article, nous aurons encore besoin d'une instance de cette classe dans nos autres tests. Pour rester concis, nous pouvons utiliser la méthode *setup*. Cette méthode sera

lancée avant chaque test et permet d'initialiser les ressources qu'ils utilisent. Nous pouvons dès lors refactoriser notre exemple comme suit :

Fichier test/unit/article_test.rb

```
def test_should_create_record
    assert @article.valid?, "Article n'est pas valide"
  end

  def test_should_not_be_valid_if_no_title
    @article.title = nil
    deny @article.valid?, "Article est valide alors qu'il ne devrait pas"
  end

  def setup
    @article = Article.create(:title=>"Les tests dans Rails")
  end
```

Nous voyons que nous gagnons encore en lisibilité. A l'opposé de la méthode setup, il existe une méthode *teardown* qui sera exécutée après chacun des tests.

Fixtures

Tout programmeur sait que les résultats des tests dépendent non seulement de la logique testée mais aussi du domaine des données. Les fixtures nous permettent de créer des exemples de données qui seront utilisés lors des tests. Les fixtures se trouvent dans le dossier test/fixtures et sont exprimés au format YAML. Nous pouvons créer un exemple de données pour notre article.

Fichier test/fixtures/articles.yml

```
tests:
  title: Les tests dans Rails
  content: Rails est qualifié de framework pour développement web agile...
  due_date: 2008-01-18 23:59:59
  publication_date: 2008-02-01
```

Nous pouvons utiliser ces données exemples de données à l'aide de la méthode fixtures :

Fichier test/unit/article_test.rb

```
fixtures :articles
```

Nous utilisons alors des objets créés à partir des exemples de données en utilisant l'instruction nom_du_modèle(:nom_des_données),dans notre cas : articles(:tests). Grâce aux fixtures, nous pouvons encore diminuer la taille du code de nos tests en modifiant la méthode setup.

Fichier test/unit/article_test.rb

```
  def setup
    @article = articles(:tests)
  end
```

Tests fonctionnels

En plus des modèles, nous avons besoin de tester la mécanique de l'application. Quand une requête HTTP est envoyée, le résultat est-il bien celui attendu ? Dans Rails, ce sont les contrôleurs qui sont chargés de traiter les requêtes. Les tests correspondant aux contrôleurs sont appelés "tests fonctionnels" et sont situés dans le dossier "test/functional". Rails propose plusieurs méthodes destinées à simuler le comportement d'une application web. Les plus utilisées sont get, put, post et delete qui simulent la requête d'une page. Lors de la création de l'échafaudage, Rails a déjà généré les fichiers de tests fonctionnels.

Extrait du fichier test/functional/articles_controller_test.rb

```
  def test_should_get_new
    get :new
    assert_response :success
  end

  def test_should_create_article
    assert_difference('Article.count') do
      post :create, :article => { :title => "test" }
    end

    assert_redirected_to article_path(assigns(:article))
  end
```

Nous voyons qu'il existe des assertions spécifiques pour vérifier les résultats des requêtes :

- assert_response vérifie le code de statut renvoyé par la requête http
- assert_redirect_to vérifie que la requête nous a bien redirigés vers la page attendue

Il existe encore d'autres assertions pour tester les contrôleurs. Par exemple, assert_tag vérifie le code html renvoyé.

Tests d'intégration

Contrairement aux tests fonctionnels qui testent les contrôleurs indépendamment les uns des autres, les tests d'intégration nous aident à tester les interactions entre les contrôleurs. Ces tests sont particulièrement utiles lorsqu'il s'agit de tester les scénarios pour lesquels l'application est conçue. Chaque scénario testé est écrit dans une classe ScenarioTest placée dans un fichier scenario_test.rb du dossier test/integration. Tous les mécanismes présentés précédemment peuvent être utilisés pour les tests d'intégration. On aura souvent une succession de plusieurs requêtes séparées par des assertions portant sur les résultats attendus.

Les tests dans Rails ne s'arrêtent pas à ce qui est présenté dans cet article. De nombreux autres frameworks de tests sont greffables à Rails. Certains, tels que Rspec [4], proposent d'exprimer les tests sous forme de spécification. D'autres, tels que Selenium [5], permettent de tester le code javascript contenu dans les pages.

Jean-Baptiste Escoyez
Co-fondateur de Belighted (www.belighted.com) : Formation et développement Ruby on Rails - Bloggeur sur Frailers.net (www.frailers.net)

Références

[1] http://ruby-doc.org/core/classes/Test/Unit/Assertions.html
[2] http://api.rubyonrails.com/classes/ActionController/Assertions.html
[3] http://zentest.rubyforge.org/ZenTest/classes/Test/Rails.html
[4] http://rspec.info
[5] http://www.openqa.org/selenium/

PHP5 : développement piloté par les tests avec PHPUnit

Les tests, au même titre que la documentation, font partie d'un bon code. Un code testable est un code maintenable. Tester son code permet de mieux appréhender le changement de spécifications et de valider le bon fonctionnement d'un algorithme.

Il existe différentes familles de tests, nous allons étudier dans cet article les tests unitaires. Ensuite, nous présenterons rapidement et installerons le framework de tests PHPUnit qui est l'une des références dans son domaine. Nous continuerons avec un projet très simple dont nous écrirons les tests d'abord, puis les classes ensuite, en refactorisant au fur et à mesure, jusqu'à obtenir un code stable et évolutif.

Pré-requis : des connaissances de base en Programmation Orientée Objet sont nécessaires.

Sur un projet informatique, plus le temps passe, plus :

- Les spécifications évoluent
- Des modifications de l'existant sont effectuées (ajouts, modifications, suppressions)
- Le code se complexifie et devient touffu
- Des bugs apparaissent

Comment s'assurer que l'introduction de code, ou d'un correctif, ne modifie pas le comportement de l'application en un point donné ? : **en écrivant des tests.** Évidemment, écrire des tests ne signifie pas que le code utile fourni n'est pas bugué. Mais cela permet de vérifier l'impact produit par un changement, sur l'ensemble de l'application.

Les tests unitaires, lorsqu'ils sont écrits convenablement, offrent donc de nombreux avantages :

- Très bon outil pour mettre immédiatement en avant les régressions, et pour s'assurer que la même régression ne se reproduira pas.
- Fournir à l'auteur du code et aux lecteurs la certitude que les corrections insérées produisent bien l'effet recherché.
- Permet de découvrir les effets de bord du système.

Mais qui teste les tests unitaires ? La réponse est simple : le code. Le code et les tests unitaires sont deux ensembles totalement complémentaires qui se répondent l'un l'autre. Cet article a pour but de vous faire comprendre l'utilité des tests unitaires, en pratiquant de façon simple le développement piloté par les tests.

Souvent appelée " Test Driven Developpement ", cette pratique est utilisée massivement dans certaines méthodes agiles de programmation, comme l'eXtreme Programming.

Le développement piloté par les tests se résume en quelques étapes :

- Ecrire quelques tests simples (représentant ce que l'on appelle un 'scénario de tests')
- Les exécuter : constater qu'ils échouent tous
- Écrire un code minimal répondant aux tests et les faisant passer
- Factoriser ce qui a été écrit
- Réitérer en rajoutant des tests (Fig.1).

Notre projet simpliste :

Nous allons créer un programme qui va gérer des personnes dans un immeuble. Nous n'implémenterons que quelques fonctionnalités, au fur et à mesure. Rien de complexe, le but étant de comprendre la démarche. Nous essayerons au maximum d'étoffer le scénario de tests, avant de toucher au code de la classe 'métier' (aussi appelée 'classe utile', ou 'code utile').

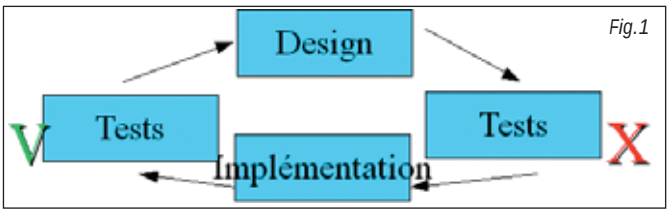


Fig.1

Tester ?

Tester est simple, vous-même le faites tous les jours : un echo, quelques var_dump , le tour est vite joué. Mais il y a un problème avec une telle technique : comment s'assurer que ces tests resteront écrits et ne seront pas modifiés ? Comment automatiser leur lancement ? Comment savoir si ce qu'ils affichent à l'écran correspond à un état attendu, ou au contraire à un état indésirable ? Ce n'est pas possible. Il faudrait, lorsqu'on introduit un changement dans du code utile, savoir immédiatement si ce changement a eu un impact sur le reste du code, ou pas. Les echo et autres fonctions du même type sont très bien pour un débogage rapide et succinct, mais à l'échelle d'une application complète, ceci ne peut clairement plus convenir. Un outil de test est nécessaire. Un outil spécifiquement créé pour tester du code (et même une application complète en simulant, par exemple, un navigateur web et des utilisateurs).

Installer PHPUnit

PHPUnit est un projet PHP très réputé dans le domaine de la testabilité logicielle. Il est calqué sur Junit, reconnu en Java. En plus d'offrir un cadre complet de tests unitaires pour votre application, PHPUnit permet d'analyser la couverture du code afin de s'assurer que toutes les lignes sont testées (couvertes), et qu'aucun code mort n'existe. De plus, PHPUnit offre des outils pour écrire une documentation agile, exporter les résultats des tests dans différents formats, automatiser les tests, et même lier l'application serveur à un navigateur au travers de Selenium, pour les tests d'intégration, et ainsi simuler une navigation (url).

Le dernier éditeur PHP de Zend : Zend Studio For Eclipse (ZS6,0) intègre un plug-in PHPUnit relativement pratique, directement dans l'environnement de travail donc. Le site Internet du projet PHPUnit est http://www.phpunit.de/. Pour installer l'outil de test, il est recommandé d'utiliser l'installeur PEAR, qui automatise tout le processus d'installation.

Installer PEAR avec WampServer

Si PEAR n'est pas installé, nous allons rapidement détailler son installation avec WampServer, qui est un programme permettant d'installer à la

fois Apache, PHP et MySQL sur Windows, en une seule passe.
D'abord, allez dans votre dossier de PHP, ici il s'agit de C:\wamp\bin\php\php5.2.5, puis exécutez **go-pear.bat** et lorsqu'on vous demande le type d'installation à utiliser, laissez 'system', par défaut, et appuyez sur entrée pour continuer (Fig.2).

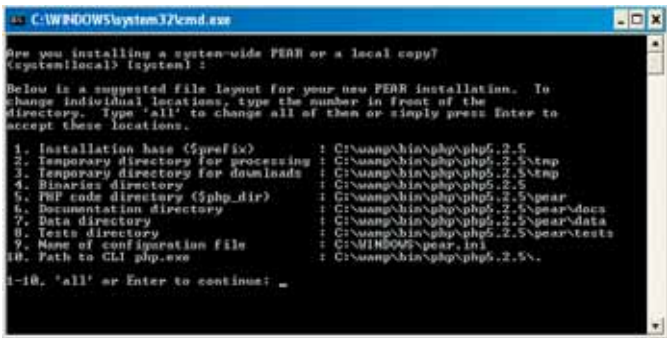


Fig.2

Gardez les chemins par défaut, faites entrée. Lorsqu'on vous demande d'ajouter l'include_path de PEAR à php.ini, dites oui mais faites attention, Wamp utilise un fichier php.ini situé dans le dossier apache2 et non dans le dossier php, il est possible que vous ayez à faire cette modification d'include_path manuellement. Ajoutez ensuite votre dossier php au path du système afin de pouvoir avoir accès à pear.bat depuis n'importe où avec l'invite de commandes de Windows. Maintenant, tapez les commandes suivantes dans une invite de commandes :

```
pear channel-discover pear.phpunit.de
pear install phpunit/PHPUnit
```

Tout devrait bien se passer, vérifiez votre PATH Windows sinon (commande 'path' dans votre invite). Une installation manuelle (hors PEAR) est aussi possible, voyez http://www.phpunit.de/pocket_guide/3.2/en/installation.html pour plus d'informations.

Tester l'installation de PHPUnit :

PHPUnit se pilote en ligne de commandes, et utilise donc le CLI de PHP. Tapez **" PHPUnit --help "** pour tester la bonne installation des paquets et voir la liste des options disponibles. Globalement, pour tester un script, il faut taper

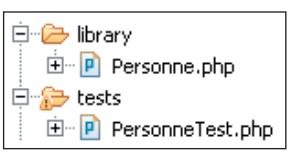
```
PHPUnit <nom_du_script.php>,
```

si le script a le même nom que la classe, ce sera le cas dans nos exemples, mais en général pas dans des cas concrets, utilisez alors

```
PHPUnit <nom_de_la_classe> <nom_du_script>
```

Organisation du projet

Dans notre projet, nous disposerons de 2 dossiers, un regroupera le(s) test(s), l'autre la(es) classe(s). Nous n'aurons dans cet article qu'un seul fichier de tests, et une seule classe utile. Les noms des fichiers correspondent aux noms des classes, par convention. Les noms des classes de tests se nomment suivant la convention <nom_de_la_classe_testée>Test.



Écrire son premier test

Nous allons commencer par écrire les tests de notre classe " Personne ". Souvent, le développement piloté par les tests est assimilé à du développement guidé par le comportement (Behavior Driven Developpement). Le test va décrire la manière dont doit se comporter la classe (et ses objets). Ceci doit être fait de manière très précise : Plus les tests sont précis et nombreux, plus ils couvrent un ensemble de cas d'utilisations larges, et plus le risque de bug est faible.
Évidemment nous commencerons doucement :

```
<?php
require_once '../library/Personne.php';

class PersonneTest extends PHPUnit_Framework_TestCase
{
    private $_personne;

    protected function setUp()
    {
        $this->_personne = new Personne('julien');
    }

    public function testPersonneInitialisee()
    {
        $this->assertEquals('julien',$this->_personne->getName());
    }

    public function testPersonneChangeEtage()
    {
        $this->_personne->setEtage(6);
        $this->assertEquals(6,$this->_personne->getEtage());
    }
}
```

Une classe de test doit hériter de PHPUnit_Framework_TestCase (en général). Il est important de noter que chaque méthode de tests :

- doit être publique
- doit commencer par le mot 'test'
- doit décrire au mieux le test réalisé, même si son nom doit être très long
- Il est conseillé d'écrire ses méthodes en CamelCase afin que le générateur de documentation agile puisse fonctionner

Tous les tests sont effectués de manière totalement isolée des autres. La méthode protégée setUp() est appelée avant chaque tests, et le framework va tester toutes les méthodes commençant par le mot 'test'. Également une méthode tearDown() peut être utilisée pour faire du nettoyage : elle est de son côté appelée à la fin de chaque méthode de tests. Ici notre classe teste un objet de la classe Personne. Elle s'en affecte une instance dans une propriété protégée $_personne, qui sera utilisée dans tous les tests. Rappelons que si un des tests change l'objet Personne, celui-ci sera remis en état pour le test suivant (isolement des tests). Tous les tests fonctionnent sur des assertions, donc des vérifications méticuleuses. La lecture du test doit permettre de comprendre ce que l'on attend de la classe (de l'objet) testée. Dans notre exemple nous avons :

- *TestPersonneInitialisee()* va s'assurer que l'objet est correctement construit, on voit d'entrée que celui-ci doit posséder une méthode getName().

• *TestPersonneChangeEtage()* est tout aussi explicite : tester qu'une personne puisse changer d'étage de manière convenable.

Jouons le test (Fig.3).

L'erreur est explicite, le fichier n'existe pas, c'est normal nous ne l'avons pas encore créé. Créons le au bon endroit, et tant qu'à faire, créons en la classe :

library/Personne.php :
```
<?php
class Personne
{

}
```

Rejouons maintenant le test (Fig.4). Il manque la méthode getName() à notre classe Personne. Ici déjà, les tests nous guident, il faut créer toutes les méthodes utilisées :

```
<?php
class Personne
{
    public function getName() { }

    public function setEtage($etage) { }

    public function getEtage() {    }
}
```

PHPUnit possède une option de lancement : --verbose qui permet lorsqu'une erreur survient, d'afficher plus de détails. Nous allons ici l'utiliser par défaut, en modifiant le lanceur PHPUnit pour ne pas être obligé de rajouter l'option à chaque appel. Nous supposerons donc sur les captures d'écran suivantes, que cette option est activée.

Une fois encore : relançons le test, après avoir écrit les 3 méthodes, mais sans les remplir (Fig.5). Nos 2 tests ont échoué, mais on s'y attendait vu que les méthodes ne contiennent rien et que nous avions indiqué dans les tests que ces mêmes méthodes devaient renvoyer quelque chose.

Il faut donc remplir les méthodes afin de faire en sorte que les tests passent. Voyons ceci :

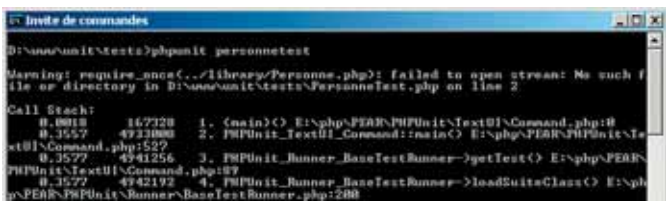


Fig.3

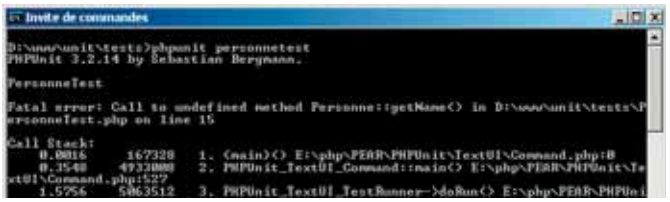


Fig.4

```
class Personne
{
    public function setEtage($etage) {      }

    public function getEtage()
    {
        return 6;
    }

    public function getName()
    {
        return 'julien';
    }
}
```

Voilà ! Bien évidemment, la classe finale ne ressemblera pas à ceci. Si nous rajoutons un test pour une personne 'Elodie', alors cela va échouer à nouveau. De même si on envoie la personne à l'étage 8, la classe retournera 6 tout le temps, ce qui échouera.

Il faut donc refactoriser le code, c'est-à-dire changer sa structure interne, mais sans que cela ne se remarque sur son API externe (la manière dont on l'utilise).

Voici une classe un peu mieux organisée :

```
class Personne
{
    private $_nom;
    private $_etage;

    public function __construct($nom)
    {
        $this->_nom = $nom;
    }
```

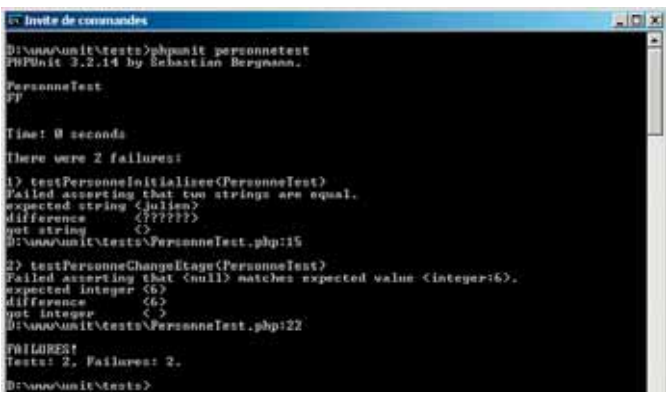


Fig.5

Fig.6

www.sauvons-les-kangourous.com

www.sauvons-les-kangourous.fr

Evitez toute confusion, déposez votre nom de domaine en .com ET en .fr !

amen
IN WEB WE TRUST

2 € HT/an
soit 2,39 € TTC/an
au lieu de 12 € HT/an, soit 14,35 € TTC/an
Du 1er juin
au 31 juillet 2008
Code promo : .FR-A2E

LE PACK "WEB NOM+.FR" : VOTRE NOM DE DOMAINE .FR, DE NOMBREUX SERVICES, FACILEMENT, RAPIDEMENT ET À 2 € HT/AN SEULEMENT !

- Votre nom de domaine en .fr
- Une redirection web transparente
- Un compte de messagerie
- Un antivirus et antispam
- Au choix : un blog, un album photo ou un outil de gestion de contenu
- Un outil de création de site en ligne (2 pages)
- Votre boutique en ligne "e-commerce Free"
- Un hébergement de 1 Go
- 25 € offerts pour promouvoir votre site sur GOOGLE

Adhérent AFNIC

Pour plus de renseignements : 0892 55 66 77 (0.34 €/mn) ou www.amen.fr
NOMS DE DOMAINE - EMAIL - HÉBERGEMENT - CRÉATION DE SITE - E-COMMERCE - RÉFÉRENCEMENT

```
    public function setEtage($etage)
    {
        $this->_etage = $etage;
    }

    public function getEtage()
    {
        return $this->_etage;
    }

    public function getName()
    {
        return $this->_nom;
    }
}
```

Les tests passent toujours après cette petite modification. Donc celle-ci n'a pas changé la manière dont la classe réagit, c'est bon signe. Rajoutons maintenant d'autres tests. Les tests unitaires : c'est simple (bien que ça se complique en réalité, comme se compliquent les classes testées) : il s'agit d'un ping-pong entre la classe de test, et la classe testée. Le test montre clairement et nettement ce que l'on attend de la classe testée, et celle-ci est le répondant du test. Nous allons continuer notre test en simulant que notre visiteur se déplace dans les étages de l'immeuble. Voici un nouveau test à rajouter à notre scénario :

```
public function testPersonneChangeEtageAvecEtageNegatifEtNonEntier()
    {
        $this->_personne->setEtage(4.2);
        $this->assertEquals(4,$this->_personne->getEtage());
        $this->_personne->setEtage(18.76);
        $this->assertEquals(18,$this->_personne->getEtage());
    }
```

Remarquez : nous avons rajouté un test, lorsque PHPUnit les exécute, chaque test qui passe est représenté par un point, chaque test qui échoue est représenté par un F. Les tests incomplets par un I, et les erreurs PHP par un E (sauf si elles sont fatales, là : tout s'arrête forcément, comme une classe inexistante par exemple).

Ici il y a un problème. En envoyant la personne à l'étage 4,2 , je m'attends à ce que son étage soit 4, mais il est 4,2 . Un étage ne pouvant être décimal, corrigeons la classe Personne en forçant la donnée en entrée à passer en entier :

```
public function setEtage($etage)
    {
        $this->_etage = (int)$etage;
    }
```

Rejouons le test maintenant (Fig.8).

Tout va bien.

Remarquez qu'à chaque modification, il faut relancer les tests, et ne jamais effacer un test sous prétexte qu'il passe. Justement, il va servir de test de régression afin de savoir si ce que nous modifions dans notre classe ne change pas le comportement qu'on lui a assigné (via sa classe de Tests).

Par exemple pour implémenter une méthode __toString() nous devons modifier nos tests, de cette manière là :

```
public function testPersonneInitialisee()
    {
        $this->assertEquals('julien',$this->_personne->getName());
        $this->assertEquals('julien',(string)$this->_personne);
    }
```

Et dans la classe Personne, le code répondant :

```
    public function __toString()
    {
        return $this->getName();
    }
```

(Fig.9).

Tout est OK.

Notre projet est terminé. Maintenant, il faudrait pouvoir compter le nombre de personne dans chaque étage, faire attention à ce que l'on ne puisse pas insérer la même personne dans deux étages différents (on la coupe en deux ?), ...

Avec les tests unitaires, c'est simple, car on écrit d'abord ce que l'on veut que la classe et ses objets fassent, comment nous souhaitons les voir réagir.

Ce n'est qu'ensuite que nous écrivons la classe utile, jusqu'à ce qu'elle passe tous les tests.

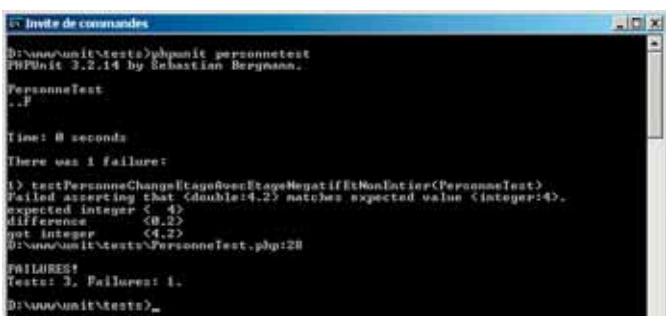


Fig.7

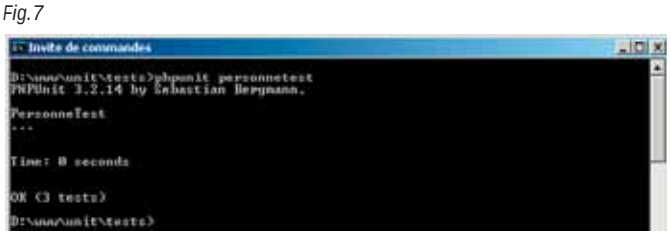


Fig.8

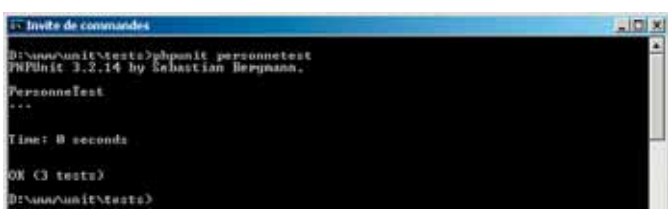


Fig.9

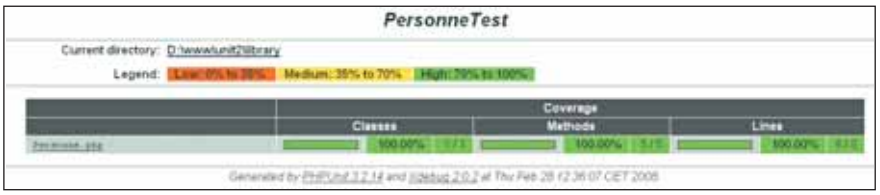


Fig.10 : L'analyse de couverture de code, par dossier, reportée en HTML par PHPUnit.

Fig.12 : Les tests PHPUnit dans Zend Studio.

```php
public function getEtage()
{
    return $this->_etage;
}

public function getName()
{
    return $this->_nom;
}

public function __toString()
{
    return $this->getName();
}

public function getSexe()
{
    return $this->_sexe;
}
```

Fig.13 : L'analyse de couverture par PHPUnit dans Zend Studio

Autres options de PHPUnit

Documentation agile

L'option --testdox-html permet de générer une documentation agile au format HTML, elle ressemble à ceci :

Personne

- Personne initialisee
- Personne change etage
- Personne change etage avec etage negatif et non entier
- Chainage set etage

Si vos méthodes de tests sont bien écrites en CamelCaseSyntaxe, alors PHPUnit est capable de séparer les mots.
Cela paraît inutile, mais les scénarios de tests, ou les suites de tests comportent souvent plusieurs centaines, voire des milliers de tests. La documentation agile devient très utile dans un tel cas.

Couverture de code

L'option de couverture de code est très intéressante. Vous devez posséder l'extension PHP Xdebug, disponible sur http://xdebug.org/, pour faire fonctionner cette option, sans cela l'option ne sera pas activée et PHPUnit vous dira qu'il ne la connaît pas. L'option est --coverage-html. La couverture de code va analyser les lignes sur lesquelles passent vos tests, et celles qui ne sont jamais testées. Ceci permet de repérer le code mort, c'est-à-dire du code issu d'opérations de refactorisation, mais laissé à l'abandon et oublié. Aussi, l'opération d'analyse de couverture de code permet de repérer les parties du code d'une classe qui n'ont pas été testées du tout, et qui peuvent donc potentiellement contenir des bugs (Fig.10). Notre classe a été totalement testée, toutes les lignes de code ont été exécutées au moins une fois, dans tout le scénario de test qu'on a écrit. Voyons le détail (Fig.11). Les lignes vertes ont été testées. En face de ces lignes, correspond un numéro, qui représente le numéro du test qui est passé dessus.

Fig.11 - L'analyse de couverture de code, par fichier, reportée en HTML par PHPUnit.

Intégration à Zend Studio For Eclipse

Début 2008 est sorti le nouvel IDE de Zend consacré à PHP : Zend Studio For Eclipse. Il est, comme son nom l'indique, basé cette fois ci sur Eclipse. Un point très intéressant de cette nouvelle version est l'intégration complète de PHPUnit. Ainsi, il est possible d'écrire des classes de tests dont les squelettes sont calqués sur des classes 'métier', mais aussi de lancer les procédures de tests directement dans l'IDE, avec analyse du code couvert (Fig.12 et 13).

Conclusion

PHPUnit est une bibliothèque puissante. Ici nous n'en avons exploité qu'un petit pourcentage. Cet outil est capable d'exécuter des tests de charge, des tests d'intégration ... Il possède un support de plus en plus soutenu des 'Mock Object' : ces objets déguisés qui servent à simuler une dépendance dont le package testé a besoin (une base de données, un serveur quelconque ....). Aussi, PHPUnit reconnaît pas mal de tags de documentation, comme @assert, pour générer des squelettes de classes de tests, @covers ou encore @codeCoverageIgnoreStart. De même, il est possible de partager ses objets testés entre les classes de tests. Évidemment, il faut écrire les tests. Seule une bonne connaissance de la conception logicielle permettra d'écrire des tests puissants, utiles et efficaces. En effet, pour rendre un code pleinement testable, déformable, mais toujours fonctionnel, l'analyse de la communalité et des variations, ainsi que l'application des design patterns, devient vite indispensable.

Julien PAULI *travaille chez Anaska où il est formateur sur la technologie PHP. Attaché au génie logiciel et à l'intégration des pratiques de programmation reconnues pour PHP en entreprise, il est aussi contributeur actif au projet ZendFramework, dont il assure également les formations.*

+ Tutoriel sur
www.programmez.com/tutoriel.php

Automatiser les tests d'IHM avec UI Automation Framework

Les tests de l'interface utilisateur ont toujours été un sujet difficile à adresser autrement que par les tests manuels utilisateurs. Dans cet article, nous allons montrer comment il est possible d'automatiser les tests d'IHM en .NET en utilisant le Microsoft UI AutomationFramework. Les techniques sont réutilisables, quelle que soit la technologie de rendu (Win32, Windows Forms, WPF, …).

Remarque : *ces tests IHM sont bien entendus possibles dans d'autres langages. Des tests à considérer par tous les développeurs !*

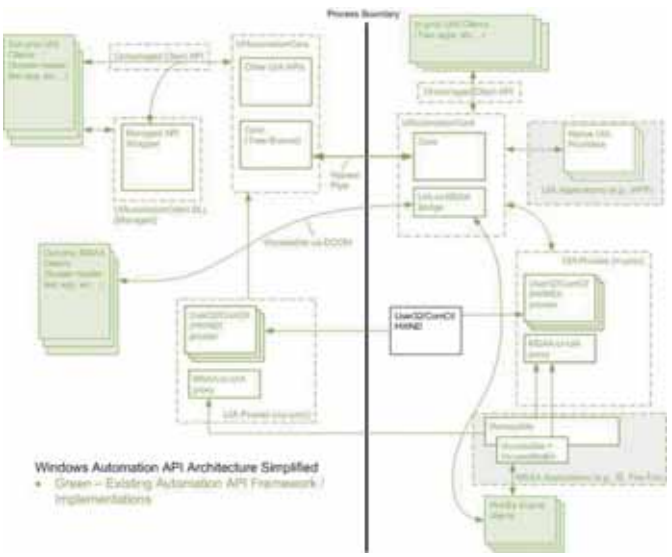
Les tests ont pris une part importante des projets informatiques car ils sont le garant de la qualité du logiciel produit et, dans les processus de test en continu, permettent de tester la non-régression induite par les nouveaux développements et de trouver les problèmes le plus tôt possible, ce qui minimise le coût de correction. Parmi les différentes techniques de tests, il en est une qui est difficile ou chère à mettre en place : le test des interfaces applicatives. Chère, car peu d'outils existent sur le marché, et difficile, car les outils existants ne supportent que quelques technologies d'interfaces graphiques ou utilisent des techniques de test qui les rendent trop sensibles aux changements et évolutions de l'IHM pendant les phases de développement. Pourtant les tests d'Interface Homme-Machine – IHM - sont les seuls permettant de réellement tester l'application de bout-en-bout en simulant les actions de l'utilisateur. Le test d'IHM est une technique de " black-box testing " très utile pour les tests fonctionnels pour lesquels la connaissance interne du comportement ou de l'implémentation de l'objet testé n'est pas nécessaire. La plupart du temps, les tests IHM sont assimilés aux tests utilisateurs et l'application est validée par un ou plusieurs utilisateurs sur la base d'un cahier de tests regroupant l'ensemble des scénarii utilisateurs devant être joués, puis validés ou rejetés par l'utilisateur testeur. Etant donné le temps consommé par ce type de tests, ils ne sont effectués généralement qu'aux étapes importantes de livraison d'un projet et sur des applications nécessitant une très bonne qualité. Dans cet article, nous allons présenter une technologie qui permet de simplifier les tests d'IHM et de les automatiser afin de pouvoir les rejouer plus souvent et ainsi améliorer la qualité des développements d'applications riches.

Introduction à Microsoft UI Automation

MS UI Automation est une technologie disponible dans Windows XP, Vista, Windows Server 2003 et ultérieur, et utilisable par les développeurs .NET à partir du .NET framework 3.0. UI Automation Framework est une API .NET permettant de naviguer dans les objets de l'interface, d'obtenir des informations ou d'exécuter des actions sur les contrôles graphiques. Successeur de Microsoft Active Accessibility (MSAA), UIAF permet aux utilisateurs ayant une accessibilité réduite de pouvoir utiliser l'outil informatique en ayant des solutions telles qu'un lecteur d'écran pour les malvoyants par exemple. Cette capacité technique le rend tout à fait intéressant dans le cadre des tests d'IHM puisqu'à la place d'un lecteur d'écran, nous pourrions imaginer un robot testeur d'IHM simulant les actions de l'utilisateur. Le point fondamental de ce framework évolutif est qu'il n'est pas dépendant d'une technologie de rendu particulière et peut être utilisé avec des technologies Win32, Windows Forms, WPF ou encore HTML. Son indépendance vis-à-vis des technologies de rendu le positionne comme une technologie pérenne pour la mise en place de tests d'IHM automatisés et son architecture le rend par nature évolutif et ouvert à l'intégration d'autres technologies de rendu.

L'architecture

UI Automation se compose de deux parties : la partie cliente, celle qui permet d'obtenir des informations sur les éléments visuels de l'interface, et la partie fournisseur, celle qui permet de fournir des informations aux applications clientes. Ainsi, pour reprendre l'exemple du test, l'application utilisant la partie cliente est l'application de test de l'interface, et l'application testée est celle qui utilise la partie fournisseur d'informations. Le schéma ci-dessous présente l'architecture du Microsoft UI Automation.



Ce schéma d'architecture montre les deux parties communiquant des informations l'une à l'autre par l'intermédiaire du composant UIAutomationCore. Un des fondamentaux de cette architecture est qu'il n'existe pas de dépendances directes entre une application fournisseur et une application cliente. L'application cliente peut obtenir des informations sur l'application fournisseur sans avoir besoin de référencer l'un de ses composants ou d'avoir une dépendance quelconque. L'application cliente n'interagit avec l'application fournisseur que par l'intermédiaire d'UIAutomationCore et les éléments visuels de l'interface.

Les composants

Comme nous l'avons expliqué précédemment, UIAF peut s'utiliser de deux manières différentes :

- Consommer de l'information sur les éléments de l'interface (partie cliente),
- Ou fournir de l'information supplémentaire sur des éléments visuels (partie fournisseur).

Dans le premier cas, le développeur utilisera la " Client API " pour implémenter des tests ou consommer des informations. Dans le second cas, le développeur soucieux de rendre son application plus accessible utilisera la " Provider API ".

- Provider API : fournit un ensemble d'interfaces permettant d'émettre des informations sur les éléments d'IHM. Ce module se compose de 2 assemblies : UIAutomationProvider.dll et UIAutomationTypes.dll.
- Client API : Ce sont des classes .NET permettant d'obtenir/consommer des informations des éléments de l'IHM. Ce module se compose de 2 assemblies : UIAutomationClient.dll et UIAutomationTypes.dll.
- Une assembly " UIAutomationCore.dll " : c'est le code sous-jacent qui permet aux clients de l'API de consommer de l'information mise à disposition par les fournisseurs (" providers ").
- Une assembly " UIAutomationClientsideProviders.dll " : Un ensemble de fournisseurs d'automation UI pour le support des contrôles standard.

L'automation model

Dans cette partie, nous détaillerons les éléments principaux de l'API cliente et du composant UIAutomationCore. Les éléments de l'automation model que nous allons voir vont nous permettre d'implémenter des tests d'IHM très facilement quelle que soit la technologie de rendu. Pour faire bref et simple, le modèle peut se résumer de la manière suivante :

- L'automation tree : c'est l'arbre dans lequel une application cliente peut naviguer pour obtenir des informations ou interagir avec des éléments de l'interface. Le nœud racine de l'automation tree est le Bureau Windows (Desktop) et est accessible par l'objet AutomationElement.RootElement.
- Les automation elements : ce sont les éléments de l'interface, sous-entendu appartenant à l'automation tree. N'importe quel élément de l'interface peut-être obtenu via ses coordonnées X/Y, ou par le fait qu'il possède le focus, ou depuis un handle Windows (HWND) ou encore en naviguant au travers de l'automation tree.
- Les modèles de contrôles (Control Patterns) : un modèle de contrôle décrit les caractéristiques et fonctions d'un contrôle. C'est une interface avec des propriétés et méthodes. Par exemple, le modèle " TogglePattern " indique que le contrôle supporte la capacité à être coché ou décoché : cela peut-être une case à cocher (checkbox) ou un élément de menu sélectionnable par coche.
- Les types de contrôles (Control Types) : comme indiqué, cela représente le type d'un contrôle quelle que soit sa technologie de rendu. Ainsi, un bouton Win32, Windows Forms ou WPF aura la propriété ControlType égale à " Button ".
- Les propriétés (AutomationProperty) : Ils existent deux catégories de propriétés : les propriétés natives, communes à tous les contrôles (exemple : Name, Enabled…), et les propriétés spécifiques aux modèles de contrôles.
- Les événements : La notification d'événements est un élément fondamental dans l'architecture du Microsoft UI Automation ; Ils permettent à une application cliente d'être notifiée dès que quelque chose se produit dans l'interface.

UI Spy

UI Spy est un utilitaire livré avec le Windows SDK Update pour Windows Vista. Il permet aux développeurs et testeurs de vérifier le niveau d'accessibilité d'une application. UI Spy donne des informations sur les propriétés et les modèles de contrôle supportés par un élément de l'interface, les événements générés sur l'interface ou une sous-partie de celle-ci, etc.

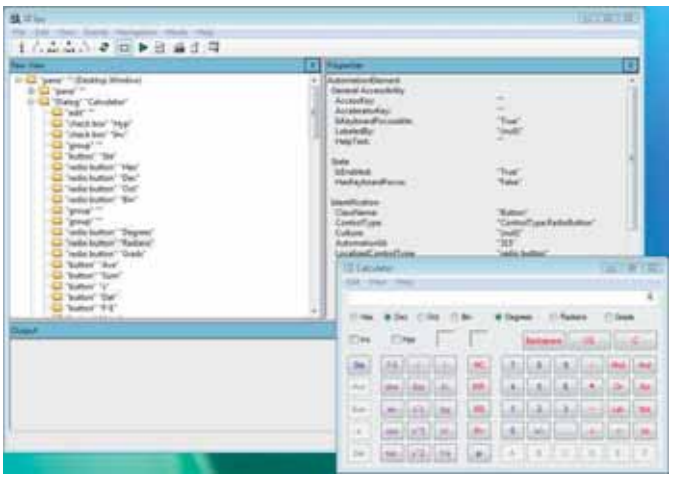


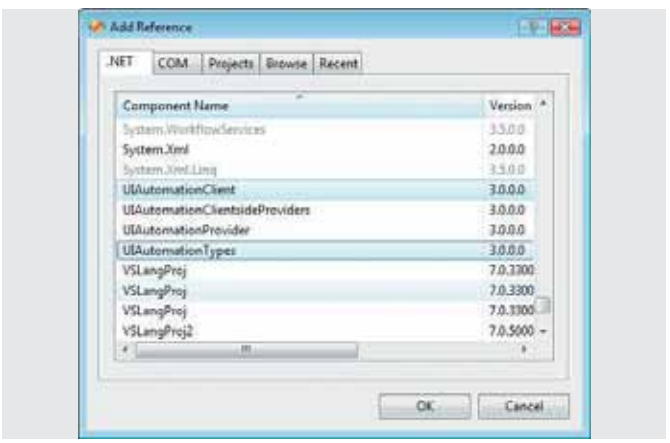
Illustration d'UI Spy avec l'application Calculator de Windows. L'automation tree est affichée dans la partie gauche de UI Spy et les propriétés de l'élément sélectionné dans la partie droite. La partie basse permet d'obtenir des informations de sortie notamment les événements générés.

Développer ses tests d'IHM

Pour illustrer le développement de tests d'IHM, nous utiliserons une application Windows simple : la calculatrice. Nous allons simuler un utilisateur qui souhaite calculer le produit de 5 et 3. Cet exemple simple nous permet de nous familiariser avec le Microsoft UI Automation, indépendamment de la technologie de rendu de l'application testée (en l'occurrence, vous pouvez appliquer ces techniques sur vos propres applications).

L'application de test

Pour créer une application de test, exécutez votre IDE favori et créez un projet .NET framework 3.0. Une application Console peut suffire. Ajouter ensuite les références aux assemblies UIAutomationClient.dll et UIAutomationTypes.dll et déclarez le namespace " System.Windows.Automation ".



Récupération d'un élément de l'arbre

Les techniques de base consistent à utiliser différentes méthodes pour récupérer un élément de l'arbre visuel. Il est possible d'utiliser l'objet TreeWalker pour naviguer parmi les éléments avec des méthodes telles que " FindAll " pour récupérer tous les éléments de l'arbre qui satisfont une condition. Attention toutefois aux ressources consommées par les techniques de navigation car effectuer des recherches sur tout l'arbre des contrôles peut être très lent. La plupart du temps, nous utiliserons la fenêtre principale de notre application comme élément racine à notre navigation/nos recherches.
Le code ci-dessous montre la récupération de l'AutomationElement racine de la fenêtre de la calculatrice :

```
...
using System.Windows.Automation;

namespace SimpleTestCalc
{
    class Program
    {
        static void Main(string[] args)
        {
            Process calculator = Process.Start("calc.exe");
            // Attendre 5 secondes pour le démarrage de la calculatrice.
            Thread.Sleep(5000);
            // Récupération de l'AutomationElement racine de la calculatrice.
            AutomationElement calcDialog = AutomationElement.FromHandle
(calculator.MainWindowHandle);
            Console.WriteLine("Titre de la fenêtre: {0}", calcDialog.Current.
Name);

            Console.WriteLine("[ENTREE] pour sortir.");
            Console.ReadLine();
        }
    }
}
```

Interagir avec l'IHM pour automatiser les tests

Pour implémenter maintenant notre scenario de test, nous devons simuler successivement l'appui sur les boutons " 3 ", " * ", " 5 " et enfin " = " pour obtenir le résultat et vérifier que cela fait bien " 15 ". Pour cela, il est nécessaire de récupérer une référence sur chacun des éléments visuels puis d'appliquer les modèles de contrôles (patterns) supportés : InvokePattern pour les boutons et TextPattern pour le champ de saisie de la calculatrice afin de récupérer le résultat. Le code est le suivant :

```
Condition name3Condition = new
PropertyCondition(AutomationElement.NameProperty, "3");
AutomationElement button3 = calcDialog.FindFirst(TreeScope.Children, name3
Condition);

Condition nameMulCondition = new
PropertyCondition(AutomationElement.NameProperty, "*");
AutomationElement buttonMul = calcDialog.FindFirst(TreeScope.Children,
nameMulCondition);

Condition name5Condition = new
PropertyCondition(AutomationElement.NameProperty, "5");
AutomationElement button5 = calcDialog.FindFirst(TreeScope.Children, name
5Condition);

Condition nameEqCondition = new
PropertyCondition(AutomationElement.NameProperty, "=");
AutomationElement buttonEq = calcDialog.FindFirst(TreeScope.Children, name
EqCondition);

Condition typeEditCondition = new
PropertyCondition(AutomationElement.ControlTypeProperty, ControlType.Edit);
AutomationElement resultEdit = calcDialog.FindFirst(TreeScope.Children,
typeEditCondition);

InvokePattern clickButton =
button3.GetCurrentPattern(InvokePattern.Pattern) as InvokePattern;
clickButton.Invoke();

clickButton = buttonMul.GetCurrentPattern(InvokePattern.Pattern) as Invoke
Pattern;
clickButton.Invoke();

clickButton = button5.GetCurrentPattern(InvokePattern.Pattern) as Invoke
Pattern;
clickButton.Invoke();

clickButton = buttonEq.GetCurrentPattern(InvokePattern.Pattern) as Invoke
Pattern;
clickButton.Invoke();

TextPattern resultPattern = resultEdit.GetCurrentPattern(TextPattern.Pattern)
as TextPattern;
int result = int.Parse(resultPattern.DocumentRange.GetText(2));
Debug.Assert(result == 15, "Erreur, le résultat n'est pas égal à 15");
```

Les exemples de code livrés avec le Automation Framework

Microsoft fournit quelques exemples d'implémentation avec UIAF pour montrer les différentes techniques d'interaction avec l'IHM. Un des exemples est un moteur d'enregistrement des actions de l'utilisateur sur une interface en WPF. A noter que WPF supporte nativement UIAF et permet de positionner des valeurs sur certaines propriétés d'UIAF directement dans le XAML (notamment l'AutomationId).
Concernant le moteur de génération des tests, la solution de cet exemple contient 3 projets :

- ExecuteScript : Projet d'exécution du script enregistré pendant la phase de tests.
- ScriptGeneratorClient : Projet de démarrage qui permet d'exécuter le lanceur d'applications chargé d'intercepter les événements de l'IHM.
- ScriptGeneratorTarget : Application exemple sur laquelle sera effectué l'enregistrement des actions.

Lancez le moteur en démarrant le projet ScriptGeneratorClient. Dans l'interface du générateur de script, cliquez sur " Start Target " et attendez que l'application exemple (ScriptGeneratorTarget) apparaisse. Ensuite cliquer sur " Start UI Automation " pour démarrer l'enregistre-

ment, puis utilisez l'interface de l'application ScriptGeneratorTarget pour modifier ou sélectionner les valeurs affichées dans les contrôles WPF. Lorsque vous avez fini vos tests, cliquez " Stop UI Automation " dans l'application ScriptGeneratorClient et cliquez sur " Generate Script " pour obtenir le code C# correspondant aux tests que vous avez effectués. Cliquez sur " Copy to Clipboard ", fermez les deux applications et retournez dans Visual Studio. Ouvrez le fichier " Script.cs " dans le projet " ExecuteScript ". Supprimez le contenu du fichier puis collez le script enregistré. Enregistrez les modifications faites dans le fichier " Script.cs " puis exécutez le projet " ExecuteScript ". Cliquez sur le bouton " Start Target " et attendez que l'application cible apparaisse. Cliquez ensuite sur le bouton " Execute Script ", vous verrez l'ensemble des actions précédemment enregistrées se rejouer automatiquement.

Trois autres exemples sont fournis :

- FetchTimer : présente une utilisation du cache d'UIAF pour l'amélioration des performances.
- Highlighter : Exemple de modification de l'apparence du focus sur une interface.
- TrackFocus : Exemple de suivi du changement de Focus depuis le bureau Windows.

Conclusion

Dans cet article, nous avons introduit le UI Automation Framework et les éléments principaux qui permettent aux développeurs d'accéder et d'interagir avec les éléments visuels d'une interface par programmation. Nous avons rapidement montré comment implémenter un test d'IHM en .NET pour une application WIN32 (la calculatrice de Windows). Cet exemple est généralisable à tous types d'applications (Windows Forms, WPF, etc.) et peut-être enrichi avec les nombreuses possibilités d'UIAF. A noter que des solutions de tests d'IHM apparaissent sur le marché et notamment le projet UIA Verify sur CodePlex ou encore l'outil de tests de la société Mpoware. Enfin, la future version de TFS (Rosario) devrait proposer une solution de test d'IHM. À vous de jouer !

Références

- La référence MSDN : http://msdn.microsoft.com/en-us/library/ms753107.aspx
- Exemples du Microsoft UI Automation : http://msdn2.microsoft.com/fr-fr/library/aa358520.aspx
- Mpoware/Outil de tests d'IHM: http://www.mpoware.com/fr-fr/Products/AutomatedUITestsTool.aspx
- UI Automation Verify (UIA Verify) Test Automation Framework: http://www.codeplex.com/UIAutomationVerify
- Tests d'IHM dans TFS Rosario : http://ozgrant.com/2008/04/22/tfs-rosario-automate-a-manual-test-add-validation/

Frédéric QUEUDRET

CTO de la société Mpoware, accélérateur d'innovation. Société française d'édition de logiciels et de prestations de services sur la création d'outils de productivité pour les développeurs Visual Studio.

http://www.mpoware.com/

Développez FUN!

Tout numéro d'été de Programmez ! qui se respecte se doit de proposer des sujets ludiques, fun. Ce numéro 110 ne dérogera pas à la tradition. Le côté fun s'exprime de différentes manières. Dans ce présent dossier, nous avons voulu montrer toute cette diversité. Le jeu aura bien entendu une place de choix. Nous vous proposons en effet trois sujets très forts : XNA, l'outil de création de jeux de Microsoft, Popfly Game, qui vous permet en quelques clics de créer votre jeu en ligne, et enfin Lego Mindstorms NTX. Que vient faire Lego dans notre dossier ? Mindstorms propose une base robotique particulièrement intéressante et les possibilités de développement sont infinies. Pour preuve, vous verrez comment créer une manette de jeux Bluetooth en Java ! Après les prouesses de la Wii, vous allez pouvoir goûter à celles de vos briques en plastique… Enfin, nous reviendrons aux jeux, cette fois, côté emploi et carrière. Le développeur de jeux a-t-il un avenir ? Peut-on y faire carrière ? Quelles sont les tendances actuelles ? Toutes les réponses dans notre enquête. Mais comme le fun n'est pas uniquement le jeu, on abordera deux sujets particulièrement excitants. Le premier concerne l'interface. Comment créer une interface à la iPod Touch en .Net ? Pour cela, vous prenez WPF, un peu de code, de l'imagination, et vous obtenez une belle interface au look du baladeur numérique d'Apple… Le second sujet est tout aussi intrigant. Comment créer votre propre langage XML ? Vous découvrirez que finalement, ce n'est pas si compliqué, l'auteur de Jaxe vous le prouvera !

Maintenant, passons à vos devoirs d'été !

■ **François Tonic**

Jeux vidéo : tendances, public, marché

Programmer, développer des jeux, oui ! Mais lesquels ? Et selon quelles modalités ? Le monde du jeu vidéo est certes en perpétuelle évolution technologique, mais les mentalités des joueurs évoluent elles aussi sans cesse. Voici un aperçu des grandes tendances ludico-économiques actuelles et prévisionnelles, histoire de savoir vers où diriger votre code…

No-life ou casual ?

En 2005, World of Warcraft sur PC d'un côté, et, de l'autre, l'amélioration des performances graphiques des téléphones portables, ont lancé la grande tendance qui prédomine aujourd'hui et guide le développement ludique. D'un côté, des joueurs " no life ", impliquant tout leur temps dans des univers persistants, jeux au développement et à la maintenance lourde. D'un autre côté, des joueurs occasionnels, s'adonnant brièvement, et pour le fun, à de petits exercices ludiques (" casual gaming "), jeux à la programmation légère, sinon ultra-légère. Et entre les deux, depuis un an, la WII : un compromis imprévisible (si ce n'est peut-être par Nintendo ?) entre le hardcore et le casual gaming, avec ses titres " familiaux ", ou son genre désormais établi des " exercices d'entraînement ", cérébraux, visuels, physiques, etc. Bref, un premier panorama triparti, structuré comme une assemblée politique : aux deux extrêmes, d'un côté le hardcore, de l'autre le casual gaming. Et au centre la Wii (et la DS), augmentées d'un nombre considérable de joueurs online assez indécis.

Plates-formes : tendances

Au-delà de cette première distinction, une fragmentation plus importante se dessine, c'est celle des plates-formes, et elle est décisive pour les développeurs. En jeu d'action 3D, ou même souvent de stratégie/aventure, les consoles Xbox et PlayStation ont relégué le PC à l'arrière-plan. En casual gaming, la répartition se fait entre PC, Wii/DS, et téléphones mobiles, le critère Internet étant déterminant. Enfin, en jeux à univers persistants, le PC garde sa suprématie. Avec deux restrictions : on note d'abord un " allègement " de ces jeux (mmorpg plus rapides, directement jouables en ligne, parfois même " casual ", comme… Barbie !). Ensuite, la croissance très rapide des possibilités de gestion et de communication des plates-formes mobiles laisse à penser que ces univers y seront bientôt pleinement exploités. Moralité : l'adéquation plate-forme / genre de jeu / type de développement, très forte jusqu'à 2006, est en voie de s'estomper, et à deux ou trois ans d'échéance, de disparaître. " Une bonne licence devra être portée sur toutes les plates-formes, et selon des modes de jeu similaires, prédit Jeff Weinberg, du VGS (Video Games Survey). Le défi des éditeurs est d'offrir très prochainement le même jeu de rôle, avec quasiment les mêmes performances d'intégration, sur PC, PS3, console portable, et téléphone mobile. Pour les studios de développement, c'est un casse-tête technologique inédit, avec des spécialisations multiples. Cette globalisation doit entraîner paradoxalement un morcellement des studios. "

L'industrie lourde

Face à cette uniformisation et en même temps fragmentation des plates-formes, des concepts, et des publics de jeu, l'industrie de l'édition a donc, elle, tendance à provoquer ses propres regroupements. La preuve, ce sont les rapprochements récents (et très étroits) entre Electronic Arts et Take Two Interactive, entre Vivendi Games et Activision, ou encore, entre édition et développement, les acquisitions de Ageia (effets physiques) par Nvidia ou de Kynogong (intelligence artificielle) par Autodesk : *" Côté édition, l'industrie connaît une phase de concentration jamais vue. "* affirme Laurent Michaud, analyste à l'IDATE. Et c'est économiquement logique, selon Henri Baetz, de KPMG Audit : *" la surenchère technologique conduit à des investissements de plus en plus chers qui peuvent aller jusqu'à 30% du chiffre d'affaires. En même temps, le démarrage d'un cycle de production/création est relativement long et donc lourd en termes financiers. Ces impératifs exigent alors une démarche structurée, des projets bien identifiés, avec un retour sur investissement de plus en plus court. Il faut donc établir des modèles économiques complexes entre constructeurs, éditeurs, studios de développement, et distribution. "*

Deux choix antagonistes

Que deviennent les studios de développement dans cette complexe machinerie ? Ils n'ont jamais que deux choix antagonistes : soit se diversifier en permanence pour répondre aux demandes toujours nouvelles des éditeurs-

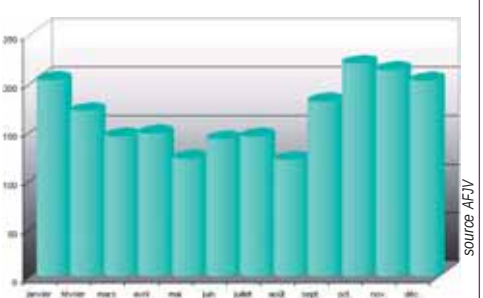
Jeux video made in France : les chiffres de l'emploi (AFJV)

En avril dernier, une étude très complète de l'AFJV (Association Française pour le Jeu Video, www.afjv.com) a montré tout le dynamisme du secteur en matière d'emploi. Voici quelques chiffres extraits de cette étude, qui en disent long :

Entre 2005 et 2008 : 2400 offres d'emploi sur le site, 92/mois de moyenne.

De 2007 à 2008 : offres en hausse de 20%.

Régions leaders : Paris-Ile de France 65 %, Rhône-Alpes 16 %. Elles le sont aussi en nombre d'entreprises du secteur (environ 430 pour toute la France). Attention : si vous êtes dans le Limousin, c'est la seule de nos régions sans aucun acteur vidéo-ludique !



Saisonnalité du nombre des offres d'emploi jeux vidéo

Postes à pourvoir : le développement est largement en tête, avec 35% des offres, les activités graphiques n'arrivant qu'en deuxième position, assez loin, avec 22%. Donc, rangez vos crayons de couleur, et restez fidèle au code, c'est toujours la demande principale !

Timing : sans être fortement saisonniers, les besoins d'emploi du jeu vidéo connaissent une nette augmentation en septembre, octobre, et novembre. Visez juste !

clients, soit se spécialiser agressivement sur un secteur très pointu, jeux mobiles en 2D, Mmmorpg online, etc. La solution de la diversification est bonne pour l'emploi, mais fragilisante : les programmeurs deviennent légion en tant que simples intervenants ponctuels sur un projet. D'un autre côté, la spécialisation pointue est, elle, dangereuse : la fluctuation des tendances implique aujourd'hui une visibilité à seulement court ou moyen terme, hormis sans doute sur le secteur de l'action 3D pour consoles, valeur sûre et pérenne. Mais encore faut-il pouvoir suivre le rythme de progression technologique des plates-formes… Bref, quelle que soit la solution choisie, la création d'un studio de développement important est depuis 2006-2007 une aventure très lourde. Pour Henri Baetz, *" la start-up doit partir à la recherche d'investisseurs, en s'adossant à des fonds de capital-risque, ou par des introductions en bourse, avec à terme l'adossement à des grands groupes qui ont la capacité de se développer. "*

Exact, mais cette vision des choses est cependant excessive : le casual gaming, les petits jeux online et les plates-formes mobiles, offrent désormais une véritable niche, c'est-à-dire des opportunités de démarrage " en douceur " d'un petit studio spécialisé.

De l'individu au studio

Car la démarche progressive, presque " garage ", devient très fréquente dans le marché actuel, parce qu'il la favorise : les exemples abondent de programmeurs ou de développeurs individuels qui, après des interventions multiples et ponctuelles sur des projets de studios différents, trouvent alors l'ouverture pour créer leur propre studio, unique et durable. Ils s'appuient ainsi sur la spécialisation de développement dont ils ont fait l'expérience sur le terrain, y ayant vérifié à la fois leur propre performance et le niveau de demande des éditeurs ou grands studios. En outre, l'augmentation des petits développements du casual gaming permet, avant de s'attaquer à des projets plus ambitieux, de doser au coup par coup le besoin d'investissements, de collaborateurs, et de mise à jour technologique. Bref, dans la pyramide du développement, il s'agit de monter d'un cran pour devenir soi-même " consommateur " d'intervenants ponctuels. C'est assurément la piste à privilégier, hors acrobaties lourdes de capital-risques ou autres entrées en bourse.

J.M. Maman

Epitech Game Developer Laboratory

L'école d'informatique française possède un laboratoire de développement dans les jeux vidéo depuis 2002. Il est né sur les " cendres " d'une association consacrée à cette activité. Très orienté console, le PC s'est imposé même si depuis quelque temps, la console revient en force, notamment avec la DS et le Xbox 360.

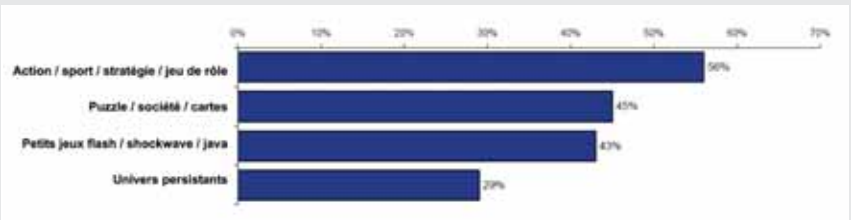
Ce laboratoire ne fait pas partie proprement dit du cursus mais il permet de réaliser des projets libres pour les étudiants de 2e année. Cela permet ainsi d'acquérir les bases de la programmation de jeux, les interactions, les librairies. Surtout, la partie technique change quasiment chaque année. Ainsi côté 3D, l'an dernier, les développeurs utilisaient surtout Ogre 3D, cette année c'était Cube2. Grâce à la signature de partenariats avec des éditeurs de jeu, les étudiants du laboratoire travaillent sur des projets concrets, en développant des maquettes, des prototypes jouables. Outre le game play, il s'agit aussi pour ces étudiants d'élaborer et de proposer des idées via ces projets. Le dernier partenariat est l'un des plus excitants du lab. Il s'agit, pour trois étudiants, de développer les nouvelles versions des pilotes et SDK du projet Majook. Majook est une plate-forme pour objets communicants basée sur une table capable de reconnaître les objets que l'on y dépose (même principe que le produit Microsoft Surface).

François Tonic

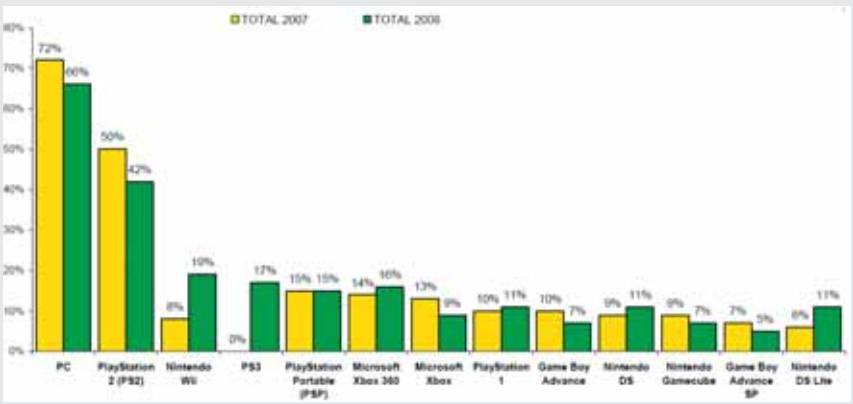
Europe : sondage des tendances ludiques (Nielsen)

L'institut d'études d'opinion Nielsen a réalisé, en avril 2008, une prospection globale des habitudes de jeu au niveau de toute l'Europe. Les réponses des joueurs de tous les pays confirment pleinement les tendances annoncées : montée du casual gaming, mélange des types de joueurs sur le online, grignotement du PC par les consoles de jeu les plus familiales (Nintendo).

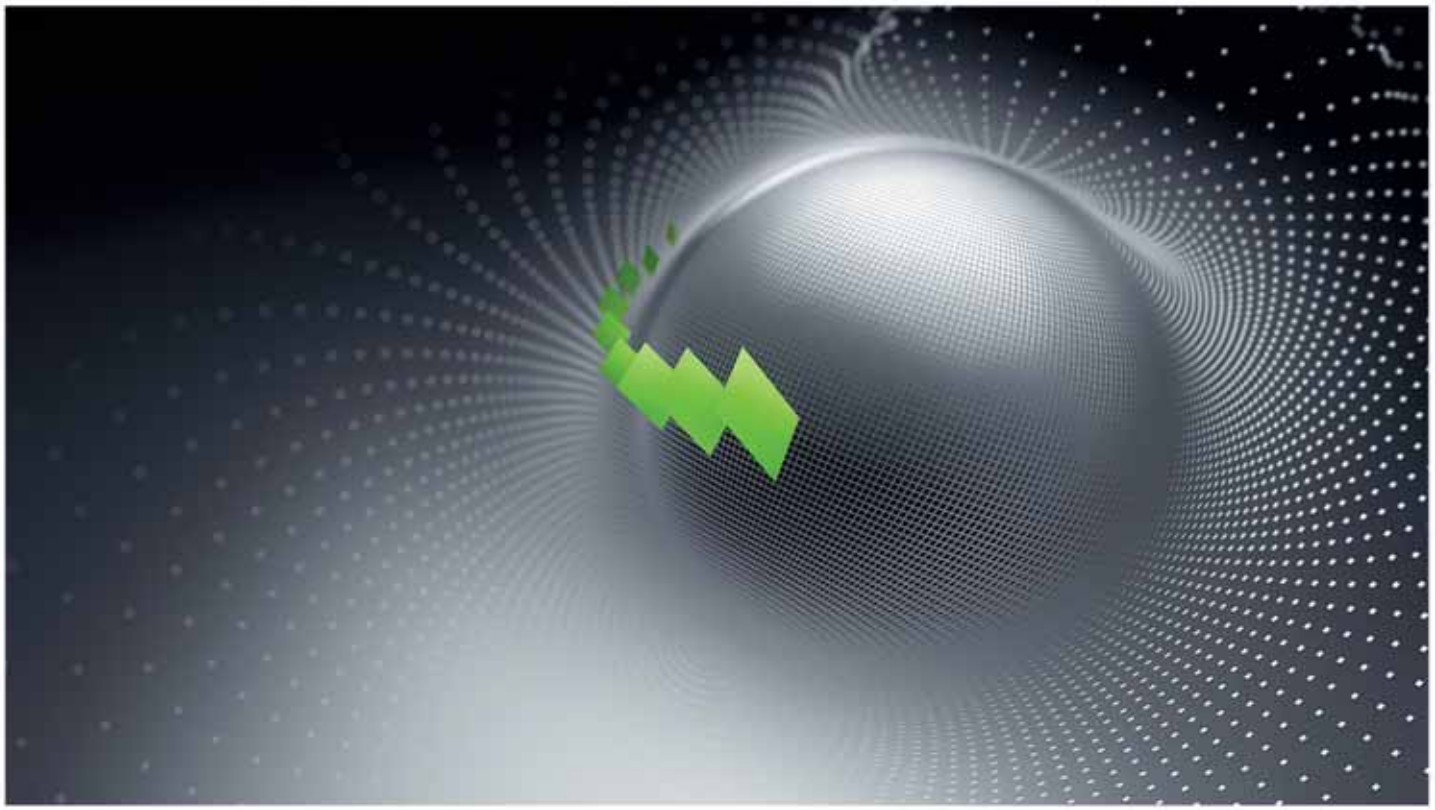
Jeu en ligne : Types de jeux pratiqués en ligne :



Plates-formes de jeu : entre 2007 et 2008, le public PC s'effrite de 14%, au profit de celui des consoles Nintendo (Wii et DS, seules en hausse) et bien sûr de la PS3 (nouvelle venue). On note donc à la fois un glissement du PC vers les consoles, et du jeu " dur " vers le jeu " léger " et familial (dont Nintendo est emblématique). L'ascension de la PS3 est relativement décevante par rapport aux scores de la Wii / DS. Sur quelle plate-forme jouez-vous ?



NVIDIA® CUDA™



NVIDIA CUDA est le seul environnement de développement en langage C au monde, permettant l'accès aux GPU NVIDIA. CUDA permet aux développeurs d'utiliser les GPU NVIDIA pour résoudre les challenges de calculs les plus complexes telles que les forages pétroliers et gaziers, la gestion des risques financiers, le design des produits, l'imagerie médicale et la recherche.

Démultipliez la puissance du GPU!
Avec le « CUDA Challenge », NVIDIA encourage la communauté Européenne des développeurs à créer des softwares qui exploiteront au mieux la puissance de traitement des GPU NVIDIA.

De très beaux cadeaux récompenseront les meilleurs projets !

Ne ratez pas l'opportunité de découvrir les dernières informations sur CUDA (communiqués de presse, documentation...)

Pour participer au CUDA Challenge
ou
Pour recevoir les dernieres informations sur CUDA
Rendez-vous sur www.nvidia.fr/cuda

Créez vos mini-jeux en quelques clics

Dans le numéro de juin nous avions découvert l'éditeur Microsoft Popfly permettant de réaliser des mashup sans une ligne de code. Cette technologie, encore en version bêta, évolue sans cesse et vient d'acquérir un nouveau module, lui-même en version alpha, nommé Popfly Game Creator. Il permet de créer des mini-jeux intégrables dans une page web, toujours avec le même concept : aucune connaissance en programmation requise.

La technologie *Silverlight* était l'offensive de *Microsoft* pour concurrencer *Adobe Flash,* devenu incontournable pour réaliser des sites web interactifs. Pour concurrencer *Flash,* il fallait s'attaquer à son domaine de prédilection à savoir, bien sûr, ces petits jeux en ligne intégrés dans une page web et qui occupent tant de pauses déjeuners. *Popfly Game Creator* vient donc apporter la simplicité de développement propre à *Popfly* à la création de mini - jeux, en favorisant ainsi une fois de plus la diffusion de Silverlight et de son plug-in. Comme pour la création de *Mashup*, l'interface est *full Silverlight,* tout se passe en ligne en se loggant avec un simple *Live ID* sur le site www.popfly.com. Cependant, le module de création de jeux est complètement indépendant de celui de création de *Mashup.* Son interface est légèrement plus complexe à apprendre et à utiliser mais en contre-partie, les possibilités qu'il offre sont bien plus importantes. Nous retrouvons le concept de développement possible sans une ligne de code, et avec toujours la possibilité d'intégrer du *JavaScript* et du *XAML* quand cela devient nécessaire. La principale différence entre les deux modules étant la façon d'intégrer ce code. Pour la création de *Mashup*, les blocs en *JavaScript* et *XAML* étaient créés d'un côté et ils étaient assemblés ensuite via une interface très simple en *drag and drop*. Avec *Popfly Game Creator,* nous commençons par créer le jeu via une interface simple et sans ligne code et dès qu'on a besoin d'une action spécifique, nous pouvons ajouter une petite fonction *JavaScript* à un élément du jeu. Pour créer ou modifier un élément graphique, nous pouvons alterner à tout moment entre une interface de dessin à la souris et le code *XAML* correspondant. Ainsi, l'intégration de code est beaucoup plus facile car nous pouvons agir uniquement là où l'on a besoin de fonctionnalités plus avancées, par contre le code créé n'est plus réutilisable aussi facilement par les autres utilisateurs. Tandis que pour les Mashup l'intégration de *XAML* et de *JavaScript* est bien moins aisée car il faut créer un nouveau bloc pour chaque besoin spécifique, mais une fois créé, le bloc peut s'intégrer dans n'importe quel *Mashup.* Pour découvrir les possibilités offertes par *Popfly Game Creator*, nous allons créer un jeu très simple où le joueur dirigera un vaisseau qui doit rentrer à sa planète natale, en évitant les aliens et les tourelles spatiales. Nous commencerons par réaliser la base du jeu uniquement, via l'interface, à la souris qui offre déjà pas mal de possibilités. Ensuite nous verrons comment éditer le *XAML* et le JavaScript en ajoutant l'affichage du score en fin de partie, ce qui n'est pas possible avec l'interface par défaut.

Création du jeu Back From Space

Après vous être loggé sur le site www.popfly.com avec un *Live ID,* allez dans la section *Create Stuff puis Game* et vous allez rentrer dans l'interface de création de jeux. Vous avez le choix entre reprendre l'un des modèles de jeux proposés par *Popfly Game Creator* dans la section template, reprendre le jeu d'un autre utilisateur dans la section *search for more* ou bien repartir de zéro dans la section *start from scratch.* Nous allons choisir cette dernière possibilité afin d'explorer en détail le moteur de création de jeux. Vous arrivez ensuite dans l'interface de création qui se décompose en 4 sections principales. La section Actors permet de choisir et paramétrer tous les acteurs qui interagiront dans le jeu. La section Scenes permet de designer les différents niveaux et phases du jeu, en plaçant des acteurs. La section *Game* permet de définir les variables qui seront communes à tout le jeu. Et enfin, la section *Play* permet de tester et de debugger le jeu réalisé (Fig.1).

Choix et configuration des acteurs

Dans le menu, sélectionnez la section *Actors* représentée par une étoile. Nous allons commencer par le vaisseau du joueur. Dans le menu à droite, section *vehicles,* cliquez sur le vaisseau *Fighter Jet* et il sera ajouté à la liste des acteurs du jeu en haut à droite (Fig.2).

Fig.1

Fig.2

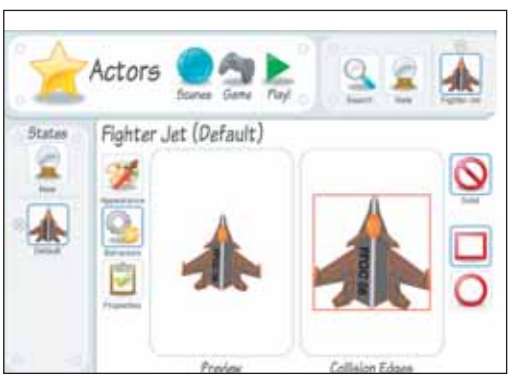


Fig.3

Cliquez dessus dans cette liste et vous accèderez à l'interface pour le personnaliser. Nous allons configurer son comportement, pour cela cliquez sur *Behaviors* (Fig.3).

Par défaut, il n'aura que la propriété de disparaître s'il quitte la scène, nous allons la maintenir mais ajouter le fait que cela produise un *Game Over* pour que le joueur reste dans l'espace de jeu. A gauche, cliquez sur *Scene*, un nouveau comportement s'ajoutera. Cliquez sur *Event* puis *Scene Leave* pour changer la condition de lancement de ce comportement. Ensuite, toujours dans ce même comportement, cliquez sur *Scene* et sur *Lost* (Fig.4).

Maintenant, il faut gérer les déplacements de notre héros. Dans le menu *Shortcuts,* sélectionnez *I want this actor to ... move up/down/left/right by pressings keys (top view)* cela ajoutera automatiquement 4 comportements de déplacement en fonction des 4 touches directionnelles (Fig.5).

En plus de se déplacer, il faut aussi que notre héros tourne en fonction de la direction prise. Sélectionnez *motion* à gauche puis dans le paramètre *Event* sélectionnez *Keyboard* et cliquez sur la touche correspondant à la flèche vers le haut du pavé directionnel. Dans le paramètre *Motion,* sélectionnez *Rotate* en haut, puis la direction nord, en bas relative *to scene* et enfin à droite *Continue Moving: While Receiving Event.* Répétez l'opération précédente pour les 3 autres cas et en ne changeant que la touche et la direction de rotation. Notre héros est fin prêt pour l'aventure, passons à l'objectif, la planète natale. Cliquez sur Search pour accéder à la liste des acteurs disponibles, et dans la section *Outdoors* prenez l'acteur *Little Planet.* La configuration va être plus simple, nous avons juste à définir qu'en cas de collision avec le vaisseau du héros, le joueur a gagné. Dans les comportements de la planète, sélectionnez Scene. Sur le paramètre Event, choisissez *Collision* et configurez le en : *Raise this event when Fighter Jet and Myself collide.* (Fig.6).

Pour le paramètre Scene, sélectionnez *Won*. Passons maintenant aux méchants du jeu, nous allons implémenter des vaisseaux aliens et des tourelles à plasma. Retournez au choix des acteurs avec *search,* et dans *Vehicles* prenez l'acteur *Cartoon Spaceship*. Dans ses comportements nous allons commencer par gérer ses déplacements afin qu'il ait un déplacement régulier de haut en bas. Pour cela, sélectionnez le comportement *Motion.* Dans le paramètre *Motion* sélectionnez la direction sud, puis *Continue Moving: For Duration 1.5* en cochant les *cases Reverse When Done* et Repeat For Ever. Il faut ensuite qu'il y ait un *Game Over* en cas de collision avec le héros, pour cela, procédez comme pour la planète mais en prenant bien sûr la scène *Lost* et *non Won.* Occupons nous maintenant de la tourelle, dans la section Building prenez *Tower – Lightning*. Dans ses comportements sélectionnez *Shoot.* Dans le paramètre Event, choisissez *Timer puis Random every 1 to 5 seconds* (Fig.7). Pour le paramètre projectile, sélectionnez *Fire Bullet* et pour le paramètre Motion sélectionnez la direction *Left.* Répétez la même opération en changeant juste la direction en *Down.* Quittez maintenant l'interface des comportements, vous verrez que les deux projectiles de la tourelle ont été rajoutés à la liste des acteurs. Nous allons donc maintenant définir leurs comportements. Sélectionnez le premier, dans ses comportements sera déjà implémenté le comportement *DisappearOnCollision.* Nous allons le maintenir mais l'adapter pour qu'il n'interagisse qu'avec le héros. Dans le paramètre *Event,* remplacez *Solid* par any instance of Fighter Jet. Il faut aussi qu'il y ait un *Game Over* en cas de collision avec le héros, pour cela procédez exactement comme pour le *Cartoon Spaceship.* Il reste à configurer l'autre projectile qui aura exactement le même comportement que le premier, pour éviter la tâche laborieuse de tout refaire, il suffit de supprimer tous ses comportements et selectionnez *I want this actor to act like ... fire bullet.* Le choix et la configuration des acteurs du jeu sont finis. Nous allons pouvoir passer à la configuration des scènes du jeu.

Configuration des scènes

Dans le menu principal de l'interface de création de jeux, choisissez la section *Scenes*. Pour ce jeu très simple nous allons nous limiter à un seul niveau. Vous serez par défaut dans la scène *Main* qui sera la scène correspondant à notre unique niveau. Commençons par définir ses dimensions, en haut dans la case *Width* entrez 710 et dans la case *Height* entrez 250. Ensuite cliquez sur Viewport qui correspond à ce que le joueur verra à l'écran, et entrez *Width* : 325 et *Height* : 250. Nous allons maintenant choisir un fond qui placera le joueur dans l'ambiance de l'espace, cliquez sur Background et sélectionnez le fond Blue Stars (Fig.8).

Placez maintenant les acteurs sur la scène comme ci-dessous, et réduisez leur dimension à environ 50 à 70 pixels de large en maintenant la touche shift appuyée pour conserver les proportions (Fig.9).

Il ne reste plus qu'à définir le comportement de la scène. Comme pour un acteur, cliquez sur le bouton *Behaviors*. Il n'y a besoin que d'ajouter un déplacement automatique de la fenêtre de vue en fonction des déplacements du vaisseau du héros. Pour cela, il suffit de sélectionner en haut à droite *I want this scene to have viewport following fighter jet 1.* Cette première version du jeu est maintenant finie il ne nous reste plus qu'à la tester.

Test du jeu

Pour tester votre jeu il vous suffit de cliquer sur le bouton *play* en haut à droite et vous pourrez directement jouer. Il y a trois options disponibles qui sont très utiles pour débugger le jeu. Vous pouvez afficher la console qui donne une liste exhaustive de toutes les actions faites dans le jeu.

Fig.4

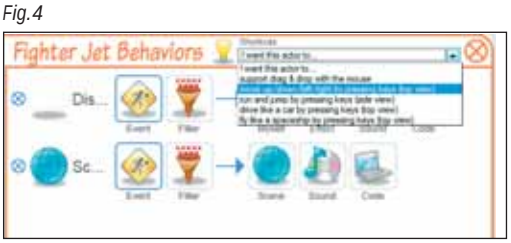


Fig.5

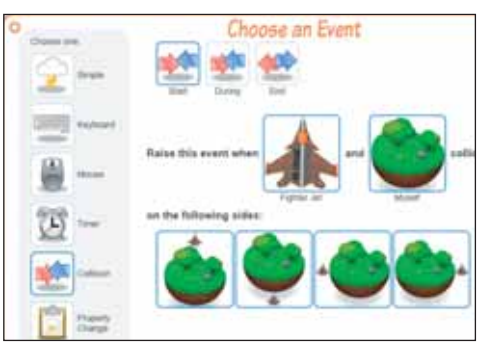


Fig.6

Fig.7

L'option *Performance* donne une idée des ressources machines nécessaires pour faire tourner le jeu à tout moment. Enfin l'option *Edge* permet de visionner les formes qui servent à la détection de collisions qui ne sont pas exactement les mêmes que celles des graphismes (Fig.10).
On a donc vu, qu'il était possible de créer un petit jeu rapidement et surtout sans la moindre ligne de code. Cependant, l'interface pourrait sembler un peu limitée car nous ne pouvons réaliser que les actions prévues dans les menus qui, même si elles offrent pas mal de possibilités, ont leurs limites. Heureusement, il est possible d'ajouter facilement du *JavaScript* et du XAML là où l'on en a besoin pour apporter des fonctionnalités spécifiques. C'est ce que nous allons voir maintenant à travers l'ajout de l'affichage du score en fin de partie.

Ajout de JavaScript et de XAML

Nous allons donc ajouter l'affichage du score en fin de partie, le calcul de ce score se fera de la manière suivante : en cas de victoire, le joueur aura 1000 points moins 10 fois le nombre de secondes passées sur le jeu. Nous n'avons donc besoin que de deux variables, le score et le temps. Le score étant une variable présente par défaut, nous n'avons que le temps à ajouter. Allez dans la section *Game,* puis cliquez sur *add a property* en mettant *time* dans la case name et 0 dans la case *initial value*. Il faut maintenant que cette variable s'incrémente à chaque seconde, ce qui peut se faire toujours sans *JavaScript*. Pour cela, retournez dans la section Scenes et dans les comportements de la scène Main, ajoutez un comportement de type *property.* Pour le paramètre *Event,* sélectionnez *Timer* puis *Every 1 seconds.* Enfin dans le paramètre *Property* sélectionnez *Change the value of the Game property time Add: 1*. Il faut aussi qu'en cas de victoire le score augmente de 1000 points. Allez dans les comportements de la scène *Won* et ajoutez y un comportement de type *property.* Dans le paramètre *property* de ce comportement, sélectionnez *Change the value of the Game property time Add: 1000*. Nous pouvons maintenant passer à l'insertion de code *XAML* et *JavaScript.*

Ajout de XAML

Retournez aux acteurs et cliquez sur New, ce qui créera un acteur vierge, sur le nouvel acteur créé, cliquez sur *Appearance.* Vous accéderez à une interface *WYSIWYG* permettant de designer votre acteur. Cependant *Popfly Game Creator* étant encore en version Alpha, la plupart des outils ne sont pas encore disponibles dans cette interface. Heureusement, en cliquant sur le bouton *Switch to XAML* vous pouvez entrer n'importe quel code *XAML* créé sous *Blend* ou à la main, il définira le design de votre acteur. Placez-y donc le code XAML suivant :

```
<Canvas Width="114" Height="98">
<TextBlock FontSize="36" Text="score" TextWrapping="Wrap" Canvas.
Left="8" Canvas.Top="0" Width="115" Height="60"/>
<TextBlock Name="print_score" FontSize="36" Text="0" TextWrapping
="Wrap" Canvas.Left="9" Canvas.Top="43" Width="89" Height="55"/>
</Canvas>
```

Ajout de JavaScript

Il ne reste plus qu'à gérer la mise à jour du bloc de texte en fonction du Score. Toujours dans le même acteur, cliquez sur Behaviors et sélectionnez *custom*. Pour le paramètre E*vent,* sélectionnez *Property Change*, puis *Raise this Event when the Game property Score changes to any value.* Enfin cliquez sur le paramètre Code et vous aurez la possibilité d'insérer un bloc de code *JavaScript* qui s'exécutera à chaque fois que ce comportement sera activé. Entrez le code suivant :

```
this.GetVisualRoot().FindName("print_score").Text=(Game.GetValue("Score")
-10*Game.GetValue("time")).toString();
```

Il ne reste plus qu'à ajouter cet acteur à la scène de victoire. Retournez dans la section *Scenes* et cliquez sur la *scène Won* en haut. Faites y glisser l'acteur que l'on vient de créer. Le tour est joué ! Vous pouvez maintenant tester votre jeu et vous verrez qu'en cas de victoire le score sera affiché. Vous pourrez trouver le jeu qui a servi à cet article à cette adresse :
http://www.popfly.com/users/Ehau78/Back%20From%20Space%20pour%20le%20magazine%20Programmez

Conclusion

En guise de conclusion comparons *Popfly Game Creator* à son principal concurrent à savoir *Flash. Popfly Game Creator* est bien plus rapide à utiliser et surtout à appréhender d'autant qu'il s'adresse au plus grand nombre car il ne requiert pas de maîtriser la programmation. De plus il est entièrement gratuit, accessible depuis n'importe quel ordinateur et ne nécessite pas d'installation puisque tout se passe en ligne. Alors que de son côté *Flash* nécessite la maîtrise d'*Actionscript,* un langage qui lui est propre, ce qui rend la période d'apprentissage et le temps de développement beaucoup plus longs, et en version professionnelle, sa licence s'élève tout de même à environ 800€. Cependant, dans la pratique les rendus finaux de *Popfly Game Creator* sont, pour l'instant en tout cas, beaucoup plus grossiers et ont un aspect beaucoup moins professionnel qu'un jeu flash bien réalisé. Cela risque donc de limiter le champ d'application aux jeux amateurs et il est peu probable que les professionnels se tournent vers cette solution pour réaliser des jeux promotionnels ou autres.

■ **Eric Hauchecorne**
étudiant à l'ESIEE – hauchece@esiee.fr

Fig.8

Fig.9

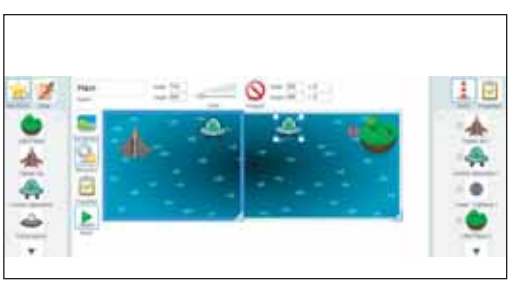


Fig.10

XNA : développer vos jeux en VB !

Sur le CD ROM

Dans le n°93 de Programmez !, nous avions abordé XNA. Pour rappel, il s'agit d'un environnement de développement de Microsoft pour les jeux 2D – 3D, sur PC et Xbox. Il s'appuie sur le Framework .NET, facilitant considérablement le développement de jeux vidéos, ce qui le rend donc accessible à toute personne souhaitant créer son propre jeu, qu'il soit développeur ou qu'il ait suffisamment de volonté pour apprendre. Tout cela gratuitement, grâce aux versions express de Visual Studio. Cette technologie arrive aujourd'hui à une certaine maturité avec la version 2.0 et la 3, en cours de finalisation.

XNA est, à la base, conçu pour le développement en C#, il existe quelques subtilités permettant son développement en VB2005 et 2008 (respectivement 8 et 9), laissant ainsi une souplesse dans le choix du langage.

Pré-requis : Afin de pouvoir procéder à l'élaboration d'un projet XNA, vous devez disposer au minimum : de Windows XP SP2, d'une carte graphique supportant DirectX 9, de Visual Studio 2005 ou 2008 (ou, le cas échéant, des versions express de Visual C# (nécessaire pour installer XNA) ainsi que de Visual Basic), et enfin de XNA Game Studio. Pour appuyer cet article vous pourrez trouver un exemple de jeu. Il s'agit d'un jeu de casse-briques. Il y a 2 sources : pour la première partie de l'article nous nous occuperons de la source nommée Casse-briques.

Création du projet

Une fois installé, démarrez un nouveau projet VB.NET, choisissez " Application Windows " et nommez-le comme vous le désirez, puis validez (Fig.1). Maintenant, cliquez sur Projet – Ajoutez une référence puis ajoutez les 7 références suivantes (Fig.2). Enfin, validez. N'oubliez pas de les rajouter à chaque nouveau projet XNA VB.NET.

XNA Device

La première chose à faire lors de la création d'un projet XNA est la création de la connexion avec la carte graphique. Elle se fait grâce à un " adapter " appelé " device ". Pour cela, allez sur le code du formulaire et, au niveau des imports ajoutez ces lignes :

```
Imports Microsoft.Xna.Framework;
Imports Microsoft.Xna.Framework.Audio;
Imports Microsoft.Xna.Framework.Content;
Imports Microsoft.Xna.Framework.GamerServices;
Imports Microsoft.Xna.Framework.Graphics;
Imports Microsoft.Xna.Framework.Input;
Imports Microsoft.Xna.Framework.NET;
Imports Microsoft.Xna.Framework.Storage;
```

Ceci va inclure les composants XNA au code de votre projet. Nous devons maintenant déclarer la variable qui va contenir l'objet " Device ". En dessous de " Public Class Form1 " inscrivez ces lignes :

```
'Variables
Private Device As Graphics.GraphicsDevice = Nothing
```

Cependant, pour le moment nous avons défini notre variable de l'objet mais celle-ci n'est connectée à aucun *adapter* de la carte graphique.

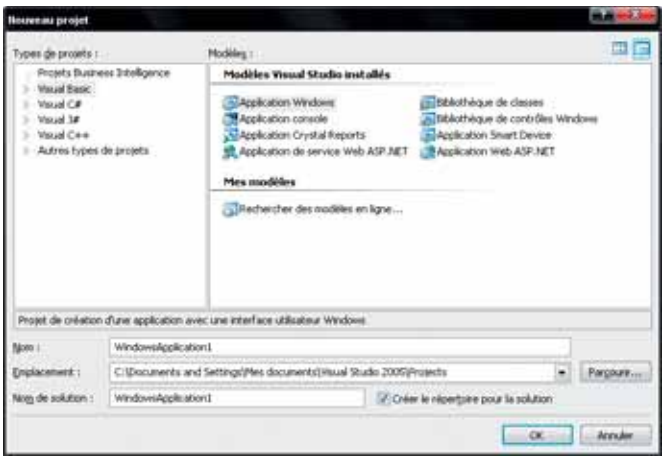


Fig.1

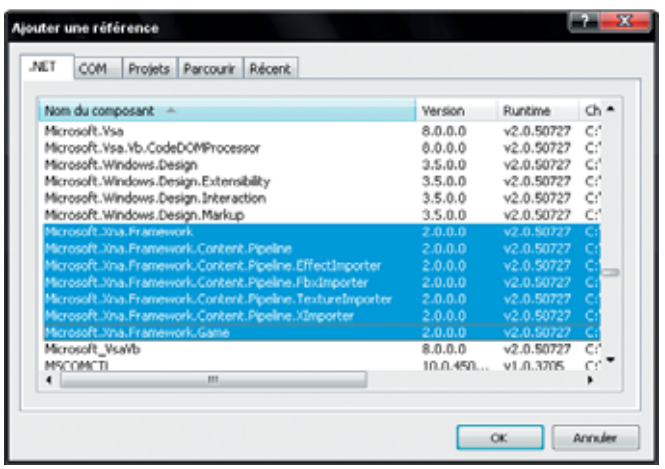


Fig.2

Ajoutez maintenant ces lignes juste en dessous :

```
Private Function IntializeGraphics()
Try
Dim presentParams As New PresentationParameters
presentParams.SwapEffect = SwapEffect.Discard
Dim XNAGraphicsAdapater As Microsoft.Xna.Framework.Graphics.
GraphicsAdapter
= Graphics.GraphicsAdapter.Adapters.Item(0)
Device = New Graphics.GraphicsDevice(XNAGraphicsAdapater, Device
Type.Hardware, Me.Handle, presentParams)
```

```
AddHandler Device.DeviceReset, AddressOf OnResetDevice
Return True
Catch ex As Exception
MsgBox(ex.Message)
Return False
End Try
End Function
```

Cette fonction nous a permis de nous connecter à l' " adapter " de la carte graphique. Nous l'avons incluse dans un bloc try-catch pour nous prévenir de tout problème éventuel. Les " Presentation parameters " sont utilisés pour définir comment le " device " est créé. Vous pouvez le modifier afin de voir les changements que cela entraîne. En cas d'erreur, le message apparaîtra dans la *message box.*
La fonction suivante va simplement recréer le " device " s'il a été supprimé ou utilisé par un autre processus :

```
Public Sub OnResetDevice(ByVal sender As Object, ByVal e As EventArgs)
If Device Is Nothing Then
'Démarre l'initialisation
IntializeGraphics()
End If
End Sub
```

Nous avons, maintenant besoin d'initialiser le tout au démarrage :

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
 System.EventArgs) Handles MyBase.Load
IntializeGraphics()
'force Windows à utiliser la méthode OnPaint pour rafraichir la fenêtre
Me.SetStyle(ControlStyles.AllPaintingInWmPaint Or ControlStyles.Opaque,
 True)
End Sub
```

Le Me.SetStyle permet de créer la fenêtre comme nous le désirons.
Nous allons maintenant gérer la fermeture sur l'appui de la touche echap :

```
Private Sub Form1_KeyDown(ByVal sender As Object, ByVal e As System.
Windows.Forms.KeyEventArgs) Handles Me.KeyDown
Select Case e.KeyCode
Case Is = Windows.Forms.Keys.Escape
Application.Exit()
Exit Sub
End Select
End Sub
```

Nous écrivons maintenant la méthode OnPaint qui va se charger d'appeler les bonnes méthodes et fonctions à chaque rafraîchissement qui est de quelques millisecondes (souvenez- vous que l'on en a exigé l'appel dans le Form1_Load) :

```
Protected Overrides Sub OnPaint(ByVal e As PaintEventArgs)
OnResetDevice(Device, Nothing)
'Effectue le rendu graphique à l'écran
Render()
'Force la fenêtre à se redessiner
Me.Invalidate()
End Sub
```

Nous contrôlons d'abord (OnResetDevice) que le " device " est toujours actif, ensuite nous appelons le rendu graphique (Render) puis nous forçons la fenêtre à se redessiner (Me.Invalidate).
Enfin, nous créons la méthode dans laquelle nous allons insérer nos rendus graphiques :

```
Public Sub Render()
If Device Is Nothing Then Return
'Applique un fond bleu comme fond de fenêtre
Device.Clear(ClearOptions.Target,  Microsoft.Xna.Framework.Graphics.
Color.Blue, 1.0F, 0)

Device.Present()
End Sub
```

Cet extrait de code vous a permis de mettre un fond bleu sur votre fenêtre.

Ajout d'une image
Ajoutons maintenant un fond et une image dont nous gèrerons le déplacement au clavier. Pour cela, faites un clic droit dans l'explorateur de solution sur votre solution et ajoutez un dossier. Nommez le " Ressources ".
Trouvez sur internet ou sur votre PC une image que vous voulez mettre en fond et une petite image d'un rectangle, puis effectuez un glisser-déposer de ces images dans le dossier
Ressources que vous venez de créer. Il faut savoir que notre image se décomposera en 4 variables que nous allons ajouter juste en dessous de la déclaration de " device " :

```
'fond d'écran
Private BkgTex As Texture2D = Nothing
Private BkgSB As SpriteBatch = Nothing
Private BkgRect As Rectangle = Nothing
Private BkgTexCreationParams As TextureCreationParameters = Texture
CreationParameters.Default
```

BkgTex va contenir la texture elle-même (l'image), BkgSB sera le sprite batch que XNA va utiliser pour dessiner cette texture, BkgRect contiendra les coordonnées de notre image et BkgTexCreationParams contiendra les paramètres d'affichage.

Ajoutez, maintenant cette partie du code qui dessinera la texture :

```
'Défini les détails du fond d'écran
Private Sub DefineBackground()
BkgRect.Height = Me.Bounds.Height
BkgRect.Width = Me.Bounds.Width
BkgSB = New SpriteBatch(Device)
BkgTex = Texture2D.FromFile(Device, "../../Ressources/LeNomDeVotre
ImageDeFond.png", BackgoundTexCreationParams)
End Sub
```

Puis, ajoutez dans la fonction InitializeGraphics(), juste au dessus de return true : DefineBackground() ce qui aura pour effet de charger les propriétés du fond d'écran au démarrage. Enfin, dans la méthode Render(), ajoutez ces lignes au dessus de Device.Present :

```
BkgSB.Begin(SpriteBlendMode.AlphaBlend)
BkgSB.Draw(BkgTex, BkgRect, Color.White)
BkgSB.End()
```

Cela aura pour effet de demander le traçage de votre fond à chaque rafraîchissement. Si vous faites F5, vous devriez voir votre image de fond dans la fenêtre.
Maintenant, créez les mêmes variables en remplaçant Bkg par Barre :

```
Private BarreTex As Texture2D = Nothing
Private BarreSB As SpriteBatch = Nothing
Private BarreRect As Rectangle = Nothing
Private BarreTexCreationParams As TextureCreationParameters = Texture
CreationParameters.Default
```

Maintenant, ajoutez une variable nommée posX :
Private PosX as double. Nous allons maintenant gérer son dessin :

```
'Défini les détails de la barre
Private Sub DefineBarre()
BarreRect.Height = 45
BarreRect.Width = 15
BarreRect.Top=me.height – 100
BarreRect.Left=me.width/2 – BarreRect.Width/2
BarreSB = New SpriteBatch(Device)
BarreTex = Texture2D.FromFile(Device, "../../Ressources/LeNomDeVotre
ImageDeRectangle.png", BackgoundTexCreationParams)
End Sub
```

Enfin, dans la méthode Render(), ajoutez ces lignes au-dessus de Device.Present :

```
BarreSB.Begin(SpriteBlendMode.AlphaBlend)
BarreSB.Draw(BarreTex, BarreRect, Color.White)
BarreSB.End()
```

Et dans la fonction InitializeGraphics(), juste au-dessus de return true : DefineBarre () pour charger ses propriétés.
Puis, on va, dès à présent, gérer le mouvement de la barre en fonction des déplacements de la souris :

```
Private Sub Form1_MouseMove(ByVal sender As Object, ByVal e As System.
Windows.Forms.MouseEventArgs) Handles Me.MouseMove
    'on va s'assurer que la barre ne quitte pas l'aire du jeu
    If  e.X – BarreRect.Width/2 < 0 Then
        BarreRect.Left = 0
    ElseIf e.X + BarreRect.Width/2>me.Width Then
        BarreRect.Left = me.Width – BarreRect.Width
    Else
        BarreRect.Left = e.X – BarreRect.Width/2
    End if
End Sub
```

La compilation et la création du setup

Etant considéré comme une application standard en VB.NET, la compilation ainsi que la création du setup (fichier d'installation) se fait de la même manière que pour une de ces applications et ce, en utilisant l'assistant publication (Générer ➔ Publier…).

Pour aller plus loin…

Les personnes ayant déjà développé en XNA sous C# doivent se demander ce qu'il advient du " content pipeline ". En effet, jusqu'à présent, les éléments actifs (" assets ") sont chargés au coup par coup à partir du disque dur. Inutile de préciser que cela ralentit considérablement le jeu. C'est pourquoi en C# il existe le content pipeline qui charge et lit ces données (les fichiers .xnb) en mémoire vive au moment où le jeu nécessite le moins de ressources.
Il y a une astuce pour créer et gérer ces " content pipeline " en VB.NET. Cependant, ceci est d'un autre niveau, c'est pourquoi, une autre source (du même jeu) est mise à disposition nommée " Casse-briques Content_Pipeline ".
Celle-ci utilise les " content pipeline ". La difficulté ne résulte pas dans l'utilisation, mais plutôt dans leur création. Pour cela, nous insérons dans notre solution un nouveau projet en mode console. Celui-ci va, en fait, générer un fichier .csproj en dynamique puis convertir les ressources en fichier .xnb. Ces fichiers ne seront générés qu'une seule fois et ne nécessiteront plus aucune intervention (vous pourrez même supprimer vos fichiers images). Il suffit ensuite, au sein de notre projet de base, de charger ces fichiers xnb une fois, pour pouvoir les utiliser à volonté au cours du jeu.
De plus, vous verrez dans la seconde source que je n'ai plus utilisé un formulaire Windows mais que celui-ci est fait automatiquement et que, héritant de la classe Game, vous aurez ainsi accès à d'autres possibilités comme le " GameTime ".

Conclusion

Cet article aura permis de montrer qu'il est possible de créer des jeux en VB.NET. Enfin, le design d'une application joue, de nos jours, un rôle prépondérant dans les applications utilisateur. Dès lors, pourquoi ne pas utiliser la technologie XNA pour incorporer des fenêtres qui permettront à l'utilisateur une manipulation graphique des possibilités de vos applications ? En effet, comme vous avez pu le voir, vous avez les mêmes méthodes que dans une utilisation Windows de base (contrôle de la souris et du clavier). Alors pourquoi ne pas rendre vos applications MDI plus attractives ? Libre à vous, maintenant, de vous lancer dans le développement de jeux vidéo.

Références

- Microsoft Visual Studio Express Editions :
http://www.microsoft.com/express/
- Microsoft XNA Game Studio :
http://www.microsoft.com/downloads/details.aspx?FamilyId=DF80D533-BA87-40B4-ABE2-1EF12EA506B7&displaylang=en
- Casse-briques.zip
- Casse-briques_Content_Pipeline.zip

■ **Florian DUBOIS**
Etudiant et membre du laboratoire .NET de SUPINFO
http://www.labo-dotnet.com/

1re partie

Je crée un “ProgrammezML” avec Jaxe

Jaxe est libre et gratuit, téléchargeable sur SourceForge : http://jaxe.sourceforge.net/Jaxe.html. Cet article explique comment créer un fichier de configuration pour Jaxe incluant le langage XML, la définition des menus d'insertion des éléments XML, et la méthode d'affichage des éléments. Il donne aussi un exemple de feuille de style XSLT. L'exemple donné, baptisé “ProgrammezML”, a permis de créer avec Jaxe ce fichier que vous êtes en train de lire.

Jaxe est un éditeur qui permet de créer à la fois son propre langage XML et un environnement d'édition personnalisé pour celui-ci. Jaxe est donc configurable pour un langage XML donné. Il faut tout d'abord créer un fichier de configuration dans lequel on définit cet environnement d'édition des éléments du langage XML. Contrairement à d'autres éditeurs qui sont, soit en mode texte, soit en mode "WYSIWYG" (What You See Is What You Get), Jaxe est un éditeur "WYSIWYM" (What You See Is What You Mean) : l'environnement d'édition est graphique, mais ne correspond pas directement à la publication, qui est une opération indépendante de l'édition. Jaxe facilite la création de documents XML valides (c'est-à-dire conformes au schéma XML) grâce à une validation en temps réel des documents en cours d'édition. L'utilisation de Jaxe passe donc par les étapes suivantes (Fig.1) :

- Création de l'environnement d'édition
- fichier de configuration de Jaxe
- éventuellement le fichier du schéma XML (si le langage n'est pas défini dans le fichier de config)
- éventuellement la feuille de style XSLT
- Édition des documents
- Publication.

Fig.1

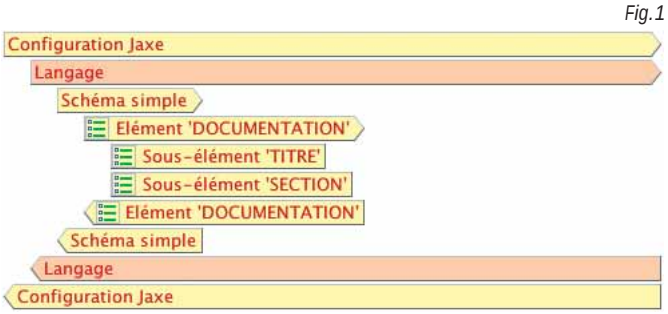


Fig.2

Création du fichier de configuration de Jaxe

Les fichiers de configuration se situent dans le dossier config de Jaxe, et leur nom se termine en _config.xml. Par exemple, le fichier de configuration des fichiers de configuration (dont l'élément racine est CONFIG_JAXE), se nomme CONFIG_JAXE_config.xml. On commence donc par lancer Jaxe, choisir le bouton "Nouveau" et sélectionner "CONFIG_JAXE - Fichier de configuration de Jaxe".

Il existe deux options possibles pour la création d'un fichier de configuration de Jaxe:

- Utiliser un fichier de schéma XML pour définir le langage XML à utiliser, et définir juste les menus et la méthode d'affichage des éléments dans le fichier de configuration.
- Définir un langage XML en utilisant une syntaxe simplifiée à l'intérieur du fichier de configuration.

On commencera ici par la syntaxe simplifiée, qui se contente de définir les éléments, leurs attributs, les sous-éléments, et quels éléments peuvent contenir du texte. On montrera ensuite comment cela peut être fait avec un schéma XML dans un fichier externe au fichier de configuration.

Définition du langage XML à l'aide d'un schéma simplifié

On commence la définition du langage XML de la configuration en insérant un élément Langage sous Configuration Jaxe. A l'intérieur de l'élément Langage, on choisit d'insérer un Schéma simple pour pouvoir définir le langage XML dans le fichier de config. Il faut alors définir *tous* les éléments du langage XML. Commençons par l'élément racine, que l'on appellera DOCUMENTATION. On insère un élément Elément sous Schéma simple, en donnant son nom et en spécifiant qu'il ne peut pas contenir de texte (d'autres sous-éléments pourront contenir du texte, mais pas l'élément racine) (attention à ne pas mettre d'espace dans les noms d'éléments). On indique ensuite quels éléments seront autorisés sous DOCUMENTATION. En l'occurrence, on prévoit déjà l'ajout des éléments TITRE et SECTION, dont on peut ajouter la référence sous forme de sous-éléments (Fig.2). Ces éléments TITRE et SECTION doivent maintenant être définis. On procède de la même façon, en autorisant le texte sous TITRE mais pas sous SECTION. Voici la liste des éléments que l'on va définir, avec les sous-éléments autorisés :

- DOCUMENTATION
  - TITRE
  - SECTION
- TITRE
- SECTION
  - TITRE
  - PARAGRAPHE

- LISTE
- IMAGE

• PARAGRAPHE
- TITRE
- EMPHASE
- CODE
- LIEN

• LISTE
- EL

• EL
- LISTE
- EMPHASE
- CODE
- LIEN

• IMAGE
• EMPHASE
• CODE
• LIEN

Certains éléments auront des attributs, que l'on définira à chaque fois avec l'élément Attribut.

• IMAGE aura un attribut fichier (obligatoire) pour spécifier le fichier de l'image.

• LIEN aura un attribut href (obligatoire) avec l'URL du lien.

Le texte sera autorisé sous : TITRE, PARAGRAPHE, EL, EMPHASE, CODE et LIEN.

Fig.3

Pour éviter de répéter les élément mélangés au texte à la fois dans PARAGRAPHE et dans EL, on pourra définir un ensemble contenant les éléments EMPHASE, CODE et LIEN, et indiquer cet ensemble comme sous-ensemble dans les éléments PARAGRAPHE et EL.

On peut remarquer à ce point que l'élément *Langage* s'affiche en orange dans Jaxe, ce qui signifie qu'il n'est pas valide. En effet, il est obligatoire de spécifier au moins un élément racine possible pour le langage (ceci est obligatoire même quand on a spécifié un fichier de schéma XML WXS au lieu de définir un langage simplifié). Dans notre exemple, la racine est DOCUMENTATION. On l'indique avec l'élément Racine dans le fichier de config, juste après Schéma simple. Voilà ce que donne la définition du langage XML dans Jaxe à ce point (Fig.3).

Définition du langage XML à l'aide d'un fichier de schéma

L'autre option pour la définition du langage XML consiste à utiliser un fichier de schéma "WXS" (W3C XML Schema). On le crée dans Jaxe en choisissant "nouveau" puis en sélectionnant la configuration "schema - Schéma XML du W3C".

A la création du fichier, Jaxe demande les attributs à mettre sur l'élément racine. On pourrait ici définir un espace de noms pour les éléments du langage, mais ce n'est pas nécessaire dans notre cas. On peut se contenter de préciser que la langue utilisée sera le français, en mettant la valeur fr à l'attribut xml:lang. A l'intérieur de l'élément racine, chaque élément est défini avec l'élément *element*. Cet élément des schémas s'affiche dans Jaxe sous le nom *Elément*. On le voit apparaître dans les éléments du panneau d'insertion quand le curseur est à l'intérieur de Schéma. Commençons donc par définir l'élément racine, DOCUMENTATION. Il suffit de créer un *Elément* et de donner la valeur "DOCUMENTATION" à l'attribut name. Si c'était un type simple (comme un nombre ou une chaîne de caractères), on pourrait utiliser l'attribut type pour définir le type de l'élément DOCUMENTATION. Comme l'élément DOCUMENTATION peut avoir plusieurs sous-éléments, on insère à l'intérieur de *Elément* un élément *complexType* qui s'affiche dans Jaxe sous le nom *Type complexe*. A l'intérieur de *Type complexe,* on doit définir les règles à respecter pour l'ordre et le nombre de sous-éléments possibles. Ici, on décide de définir une séquence avec un titre optionnel, et un certain nombre de sections après (au moins une). Ces règles sont impossibles à définir de façon aussi précise avec la syntaxe simplifiée de Jaxe, et l'on voit donc rapidement l'intérêt d'utiliser des fichiers de schémas. On insère donc une Séquence sous Type complexe, dans laquelle on place des références vers les éléments TITRE et SECTION. Les références se créent avec un *Elément* dont on utilise l'attribut ref au lieu de l'attribut *name.* Pour chacun de ces éléments *Séquence* et *Elément,* on peut utiliser les attributs *minOccurs* et *maxOccurs* pour choisir le nombre minimum d'occurrences et le nombre maximum d'occurrences. Pour un nombre maximum infini, on utilise la valeur spéciale unbounded. Voilà ce que cela donne dans Jaxe (Fig.4).

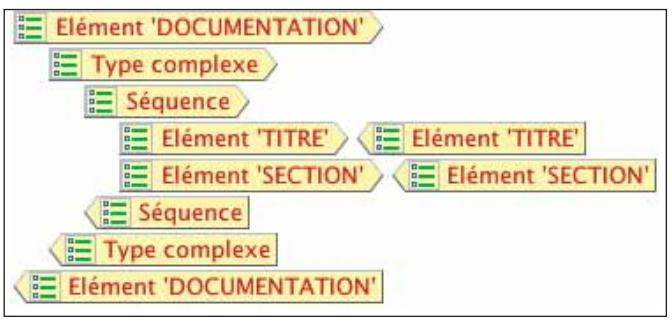


Fig.4

Comme les attributs ne sont pas tous affichés ici, il peut être utile de regarder le code source correspondant pour voir tout le contenu XML d'un coup (Fig.5).

```
<xs:element name="DOCUMENTATION">
<xs:complexType>
<xs:sequence>
<xs:element maxOccurs="1" minOccurs="0" ref="TITRE"/>
<xs:element maxOccurs="unbounded" minOccurs="1" ref="SECTION"/>
</xs:sequence>
</xs:complexType>
</xs:element>
```

Fig.5

L'élément TITRE est plus simple à définir : on crée juste un Elément au même niveau que l'autre avec les attributs name="TITRE" et

type="xs:string". Le préfixe xs ici est lié à l'espace de noms des schémas, ce qui signifie qu'il s'agit d'un type de base et non d'un type défini dans le nouveau schéma.

La définition de l'élément SECTION est l'occasion d'utiliser des règles un peu plus complexes : on souhaite ici définir une séquence commençant par le titre, suivi d'un mélange sans limite d'éléments PARAGRAPHE, LISTE et IMAGE. Voilà ce que cela donne dans Jaxe (Fig.6) et le code correspondant (Fig.7).

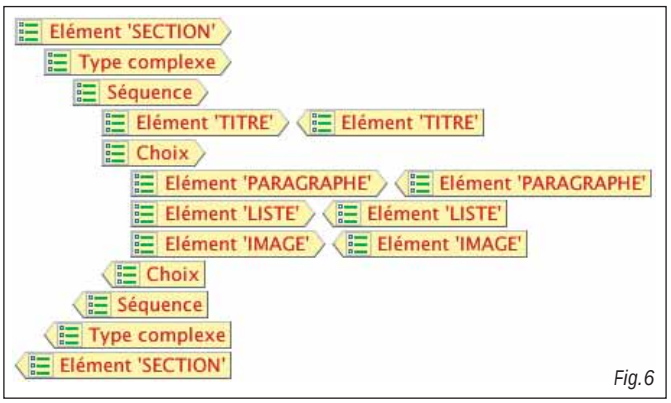


Fig.6

```
<xs:element name="SECTION">
<xs:complexType>
<xs:sequence>
<xs:element maxOccurs="1" minOccurs="0" ref="TITRE"/>
<xs:choice maxOccurs="unbounded" minOccurs="1">
<xs:element ref="PARAGRAPHE"/>
<xs:element ref="LISTE"/>
<xs:element ref="IMAGE"/>
</xs:choice>
</xs:sequence>
</xs:complexType>
</xs:element>
```
Fig.7

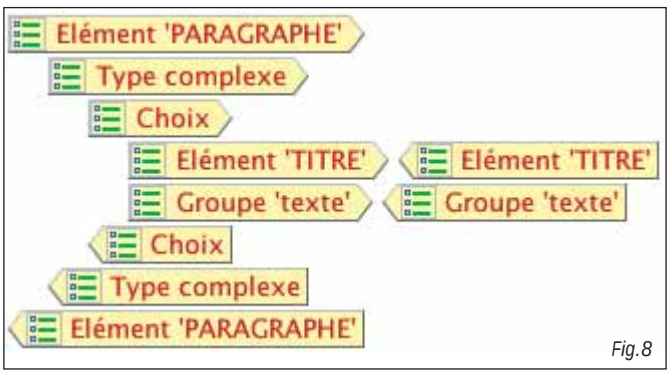


Fig.8

L'élément PARAGRAPHE, que l'on définit ensuite, a une particularité : il peut contenir du texte, mais aussi d'autres éléments. Son type doit être un Type complexe avec l'attribut mixed="true". Pour référencer les éléments que l'on peut mélanger à du texte, on peut utiliser un Groupe, une sorte d'élément virtuel permettant de référencer plus facilement un ensemble d'éléments dans le schéma, sans que cela apparaisse dans les documents XML conformes à ce schéma. On peut par exemple nommer ici ce groupe "texte", et y faire référence dans les règles de l'élément PARAGRAPHE avec un Groupe avec l'attribut ref="texte" (Fig.8).

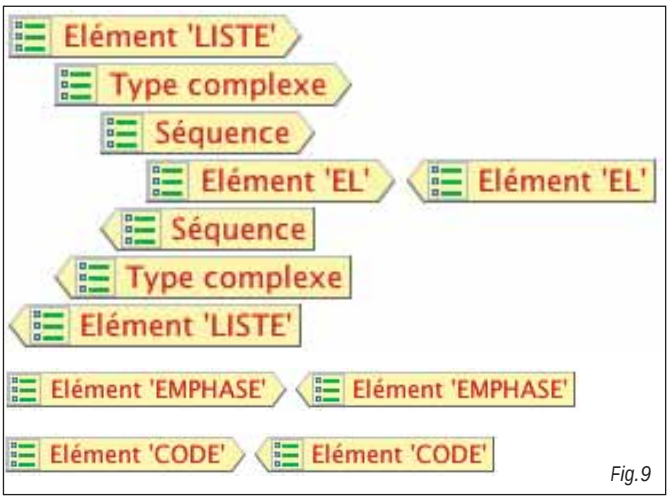


Fig.9

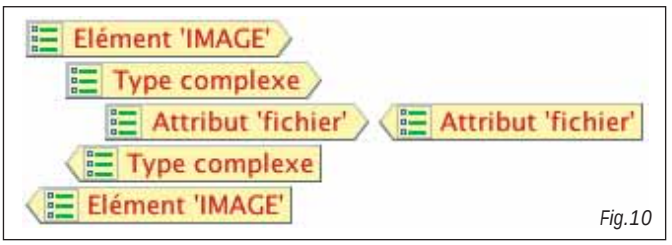


Fig.10

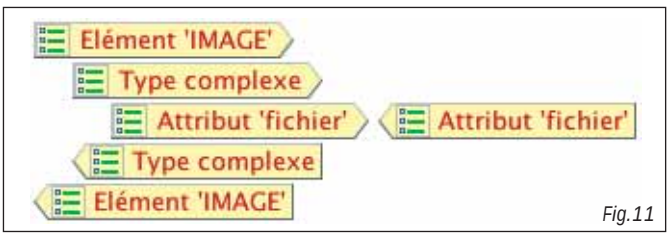


Fig.11

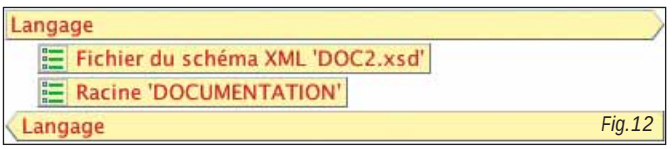
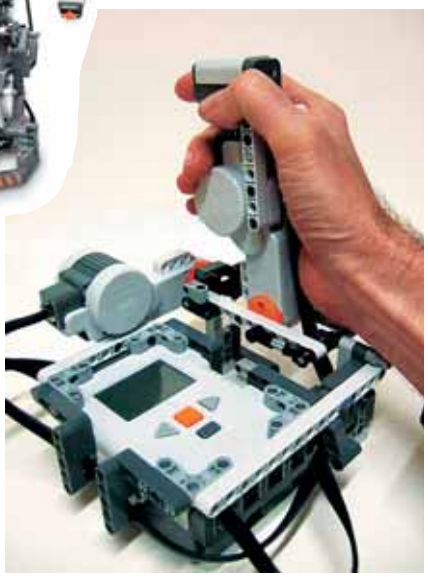


Fig.12

Les autres éléments sont définis un peu de la même manière : LISTE est une séquence de EL, EL est un choix de LISTE et du groupe texte (pour permettre la création de sous-listes), EMPHASE et CODE sont définis comme TITRE sous la forme d'un simple *xs:string* (Fig.9).
Les éléments IMAGE et LIEN ont aussi une particularité : ils ont un attribut. Un élément avec des attributs peut être défini comme type complexe même s'il n'a pas de sous-élément, les attributs étant définis à l'intérieur de Type complexe. Par exemple, l'attribut "fichier" de l'élément IMAGE est défini avec un élément Attribut ayant les attributs name="fichier", type="xs:string" et use="required" (Fig.10 et 11).
On obtient au final le fichier DOC2.xsd (voir dans le cédérom - ou le site - le fichier tutorial_jaxe_programmez/DOC2.xsd).
Dans le fichier de config, on fait référence au fichier du schéma en indiquant le chemin relatif, et en précisant quel est l'élément racine du langage. On place en général le fichier du schéma au même endroit que le fichier de config, dans le répertoire config de Jaxe (Fig.12).

■ **Damien Guillaume** - *Créateur de Jaxe* - *Damien.Guillaume@obspm.fr*
■ **Marie-France Landréa** - *Marie-France.Landrea@obspm.fr*

LEGO Mindstorms NXT, Bluetooth, Java : le trio infernal !



Le 4 janvier 2006, LEGO a annoncé la sortie d'un jeu de construction d'un genre tout particulier, permettant… la construction de robots ! Bien sûr, LEGO n'en était pas à son coup d'essai et ce kit de construction, baptisé Mindstorms NXT, était le successeur du Robotics Invention System (RIS) lancé en 1998 et fruit d'une collaboration de près de 10 ans avec le MIT.

Si RIS a rencontré un succès inespéré, il avait pourtant un bon nombre de limitations. NXT apporte une ergonomie accrue, mais aussi le support de Bluetooth, sujet de cet article. Nous allons, pour illustrer le côté à la fois amusant et puissant de NXT, développer un joystick fait maison qui pilotera en Bluetooth une application de dessin écrite en Swing.

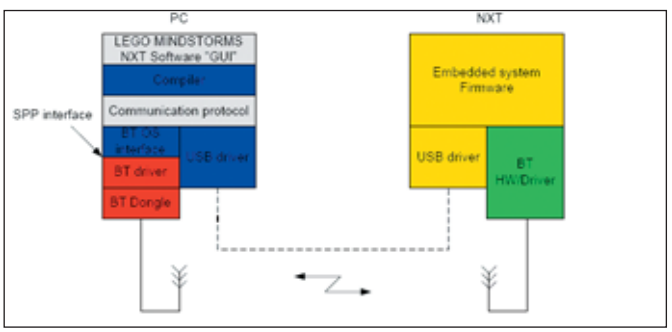
Les outils à notre disposition

Nous utilisons ici uniquement la boîte de construction standard (Ref. 8527), cette boîte est disponible dans le commerce pour environ 300 euros. Elle contient outre 577 éléments d'assemblage LEGO (provenant en grande partie de la série LEGO Technic) : une brique de programmation, 3 servomoteurs interactifs et 4 capteurs (ultrasons, son, lumière et contact). La brique de programmation est équipée d'un processeur ARM7, d'un écran LCD de 64 pixels, de 3 ports de connexion pour les moteurs, 4 ports de connexion pour les capteurs et d'une connectivité USB et Bluetooth. La brique utilise un firmware maison qui consiste en plusieurs modules, dont ceux des capteurs et des moteurs, et une machine virtuelle. Celle-ci permet notamment d'envoyer et recevoir des commandes, télécharger des fichiers (typiquement des sons ou des programmes) ou encore démarrer/arrêter un programme. Les formats des programmes sont propriétaires mais leur documentation est téléchargeable sur le site de LEGO, ce qui a permis à des projets d'écrire des compilateurs comme NBC (compilant un code source proche de l'assembleur) ou NXC (compilant un code proche du C). Mais d'autres projets sont allés encore plus loin comme le projet LeJOS qui a porté une JVM pour la brique NXT, recréant complètement un firmware. Toutefois, LEGO fournit une version modifiée de LabView (de National Instruments) permettant de programmer un robot par simple glisser-déposer de briques de programmation, simplifiant énormément la programmation pour les néophytes.

Réalisation du joystick

Afin de réaliser un joystick, il faut pouvoir mesurer l'angle d'un axe vertical dans deux directions : x et y. Les servomoteurs sont parfaits pour cela. En effet, ceux-ci peuvent fonctionner en pas à pas ou en rotation libre (ces 2 modes étant pilotés par la brique) mais aussi en mode capteur, en permettant de fournir à tout moment le nombre de rotations effectuées ou encore l'angle courant du moteur. Pour réaliser le joystick, nous allons tout simplement utiliser un modèle existant développé par Philippe Urbain et dont les instructions de montage sont disponibles à cette adresse : http://philohome.com/nxtjoystick/joystick.htm. Ce joystick est initialement conçu pour piloter un autre engin NXT mais nous n'utiliserons pas le programme à embarquer dans la brique.

Communication Bluetooth



La brique permet une communication Bluetooth avec 3 autres briques ou PC. La brique utilise pour cela un processeur BlueCore conforme à la norme Bluetooth qui réalise les fonctionnalités radios de base (Base-Band) en utilisant le profil Bluetooth SPP (Serial Port Profile) puis envoie les données reçues à l'UART de l'ARM7 pour la communication série.
LEGO a défini un protocole de communication sur le port série permettant de faire passer soit des commandes systèmes soit des commandes " directes ". Parmi les commandes systèmes, on trouve par exemple : GetFirmwareVersion, GetDeviceInfo, Boot, Read, Write.

Pour ce qui est des commandes directes, celles-ci sont plutôt dédiées à un usage courant par les programmes :

- GetOutputState, pour connaître les valeurs d'état des moteurs
- GetInputValue, pour lire les capteurs
- PlaySoundFile
- MessageWrite
- MessageRead

La liste est plus longue mais je donne ici uniquement celles que nous allons utiliser. Pour passer les commandes, le protocole se construit comme suit :

Length, LSB	Length, MSB	Command Type	Command	Byte 5	Byte 6	Etc.

D'abord la taille totale du message (sur 2 octets en little endian) est envoyée, cette taille ne comprenant pas les 2 octets de taille eux-mêmes. Puis vient le type de commande qui précise s'il s'agit d'une commande système, d'une commande directe ou encore d'une réponse à une commande. Le MSB de cet octet définit si le message nécessite une réponse (0) ou pas (1). L'octet suivant donne le code de la commande. Par exemple, la commande PlayTone porte le numéro 3 (0x03). Enfin, suivant les commandes, un payload peut être nécessaire notamment pour passer les arguments de la commande. Ce payload est différent pour chaque commande. Il faut noter que les deux octets de taille ont été introduits par LEGO pour réduire les problèmes de calcul de taille lors du passage des données de l'ARM vers BlueCore. Par ailleurs, un autre problème, de taille, est que même si le SPP permet une communication dans les 2 sens, le processeur BlueCore, lui, a besoin de pas moins de 30 ms pour passer d'un sens à un autre, ce qui induit 60 ms de délai entre deux messages demandant une réponse. Nous verrons par la suite que ceci peut poser problème pour l'approche retenue dans ce projet.

Initier la connexion

Réalisons maintenant notre communication entre le joystick et le PC. Pour l'occasion, j'ai choisi de développer le projet sous Windows, les outils livrés par LEGO fonctionnant sous ce système. Il faut noter que la communication Bluetooth fonctionne plus ou moins bien en fonction du matériel présent dans votre PC. Pour ma part, j'utilise un simple dongle USB de chez Trust qui coûte 14 euros. La JSR82 du JCP spécifie une API permettant de tirer parti des matériels Bluetooth. Malheureusement, le consortium soutenant Bluetooth est très fermé et il est difficile de trouver des implémentations libres. Fort heureusement on peut trouver Bluecove RXTX et Bluez. J'ai choisi d'utiliser BlueCove qui a eu le mérite de fonctionner tout de suite. Pour communiquer en Bluetooth avec le joystick il faut remplir 4 étapes :

1. Chercher le device - **2.** Chercher le service - **3.** Détermination de l'URL de connexion - **4.** Connexion.

La découverte des devices est simple :

```
LocalDevice.getLocalDevice().getDiscoveryAgent().startInquiry(Discovery
Agent.GIAC, listener);
```

Le programme est notifié de façon asynchrone des devices découverts (car ceci peut prendre pas mal de temps) via l'appel à l'objet implémentant DiscoveryListener passé en paramètre. Deux objets sont renvoyés : RemoteDevice qui donne toutes les informations concernant le device et DeviceClass qui donne le type de l'équipement. LEGO a paramétré BlueCore pour renvoyer le code de classe majeure d'équipement 01000 (Toy) (bits 8 à 12) et le code de classe mineure d'équipement 000001 (robot) (bits 2 à 7). Il faut donc filtrer les équipements sur ce critère :

```
public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {
if (cod.getMajorDeviceClass() == 2048 && cod.getMinorDeviceClass() == 4)
        devices.addElement(btDevice);
}
```

La méthode pour rechercher les services est sensiblement la même. Il faut appeler la méthode searchServices, le tableau d'objets UUID passé en paramètre ne contient qu'un élément : 1101 qui correspond au profil SPP.

```
LocalDevice.getLocalDevice().getDiscoveryAgent().searchServices(attributes,
uuids,device,listener);
```

Comme pour la découverte des devices, il faut implémenter une méthode de callback à savoir la méthode servicesDiscovered de l'interface DiscoveryListener :

```
public void servicesDiscovered(int translD, ServiceRecord[] servRecord) {
        / / Should only be one service on a NXT
        if (servRecord.length != 1) return;
        nxtInfo.btResourceString =
   servRecord[0].getConnectionURL(ServiceRecord.NOAUTHENTICATE
_NOENCRYPT, false);
}
```

S'il s'agit bien du NXT, seul un service est disponible. Il ne reste plus qu'à obtenir l'URL de connexion depuis l'objet ServiceRecord en appelant getConnectionURL comme illustré ci-dessus. La communication entre les deux périphériques n'est pas authentifiée ni cryptée. Cette URL est à conserver précieusement car elle pourra être réutilisée pour se connecter à ce même matériel sans refaire la coûteuse phase de recherche. Il ne reste plus qu'à se connecter en utilisant la méthode open de l'objet javax.microedition.io.Connector qui retourne un objet StreamConnection permettant d'obtenir un stream de sortie vers le NXT et un stream d'entrée depuis le NXT.

Récupérer la position des moteurs

Implémentons à présent la méthode permettant de récupérer la position d'un moteur. La commande NXT est GetOutputState qui porte le numéro 6. Cette commande prend en paramètre, sur un octet, le port à lire (0, 1 ou 2 pour lire respectivement les ports A, B ou C). Nous envoyons donc une commande directe demandant réponse (code 0x00) sur le port série. Il faut donc envoyer le tableau d'octets suivants :

Longueur (LSB)	Longueur (MSB)	Type de commande	Numéro de commande	
0x03	0x00	0x00	0x06	Port B

Ce qui donne le code suivant :

```
StreamConnection con = (StreamConnection) Connector.open(btResource
String);
OutputStream os = con.openOutputStream();
InputStream is = con.openInputStream();
```

```
byte[] request = {0x00,0x06,0x01};

int LSB = request.length;
int MSB = request.length >>> 8;
try {
    os.write((byte) LSB);
    os.write((byte) MSB);
    os.write(request);
} catch (IOException e) {
e.printStackTrace();
}
```

Puis il faut lire la réponse octet par octet :

Octet	Valeur
0	0x02 (code pour un message de réponse)
1	0x06 (numéro de la commande recevant une reponse)
...	...
21 à 24	rotation count

La réponse est complexe mais la seule valeur qui nous importe est contenue dans les octets 21 à 24 qui indiquent en *little endian* la rotation du moteur relative à la position lors du dernier *reset* du moteur. Le code est le suivant :

```
byte[] reply = null;
int length = -1;

try {
do {
        length = is.read(); // First byte specifies length of packet.
    } while (length < 0);

int lengthMSB = is.read(); // Most Significant Byte value
length = (0xFF & length) | ((0xFF & lengthMSB) << 8);
reply = new byte[length];
is.read(reply);
int rotationCount = (0xFF & reply[21]) | ((0xFF & reply[22]) << 8)| ((0xFF &
 reply[23]) << 16)| ((0xFF & reply[24]) << 24);
} catch (IOException e) {
    System.err.println(e.getMessage());
    e.printStackTrace();
}
```

Lire les capteurs

Afin de rendre les choses plus amusantes, nous implémentons une nouvelle commande comprise par la brique, getInputValue, qui porte le numéro 7 et qui prend en paramètre le port sur lequel est connecté le capteur (0 à 3). Mais avant d'émettre cette commande il faut d'abord expliciter à la brique le type de capteur connecté sur ce port et comment formater les données reçues. Pour cela, il est nécessaire d'appeler pour chaque port la commande setInputMode qui porte le numéro 5 et qui prend en premier paramètre le type de capteur parmi une énumération (1 pour le capteur de contact, 0x0B pour le capteur ultrasonique par exemple) et en second paramètre le type de valeur retourné parmi une autre énumération (0x20 pour booléen, 0x00 pour de la donnée brute par exemple). Nous allons ainsi paramétrer la brique NXT pour qu'elle reconnaisse le capteur de contact sur le port 2 du NXT (code 0x01) et nous envoie une valeur booléenne pour connaître son état :

Longueur (LSB)	Longueur (MSB)	Type de commande	Numéro de commande	Port 2	Type de capteur	Format des données
0x05	0x00	0x80	0x05	0x01	0x01	0x20

Et nous allons nous servir du capteur ultrasonique sur le port 1 pour décider de l'épaisseur du trait : plus notre main sera proche du capteur, plus le trait sera fin et vice versa, de cette façon nous allons jouer les Jean-Michel Jarre de la peinture :). La commande à envoyer est donc :

Longueur (LSB)	Longueur (MSB)	Type de commande	Numéro de commande	Port 1	Type de capteur	Format des données
0x05	0x00	0x80	0x05	0x00	0x0B	0x00

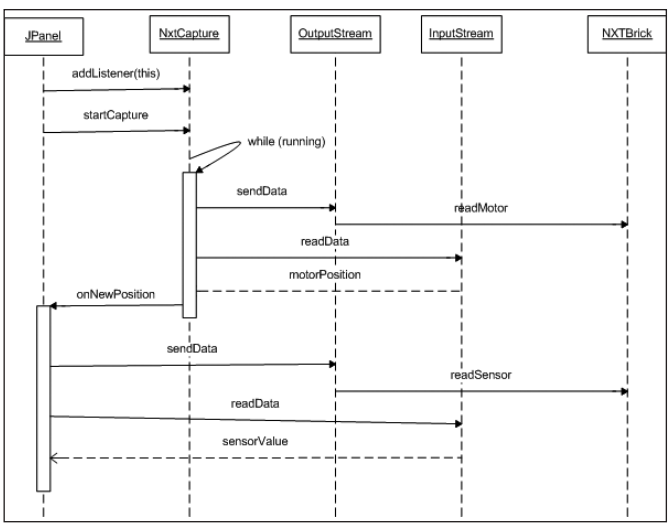
Chaque fois qu'il sera nécessaire de connaître la valeur des capteurs, il faudra alors émettre une commande getInputValue qui retournera dans le cas du capteur de contact :

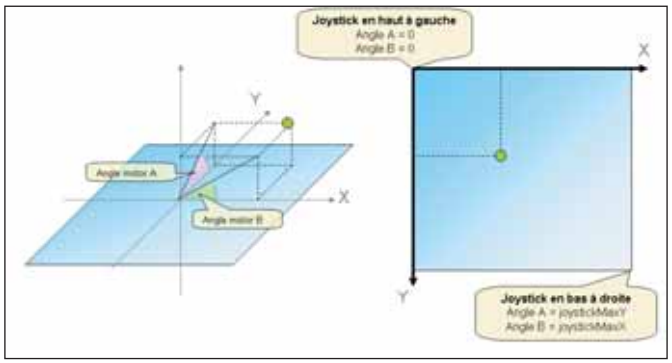
Byte 0	Byte 1	...	Byte 12	Byte 13	...
0x02	0x07		0x01	0x00	

Où le premier octet signifie qu'il s'agit d'une réponse, le suivant répète le numéro de commande pour lequel est la réponse et les octets 12 et 13 fournissent en *little endian* la valeur mise à l'échelle, c'est-à-dire formatée. Ici la valeur est donc 1, ce qui signifie que le capteur est en position enfoncée. Le décodage de la valeur du capteur ultrasonique est similaire bien que plus complexe. Au final, nous parvenons à lire une distance en centimètres entre l'obstacle et le capteur.
Il ne reste plus qu'à faire une application Swing de dessin.

Peindre les coordonnées

D'ordinaire, pour dessiner, on utilise les coordonnées de la souris, relatives à la fenêtre pour tracer une ligne entre le dernier point tracé et le point capturé par l'évènement de déplacement de la souris. Ici, il faut capturer les mouvements du joystick et les passer à l'application Swing. Pour cela, nous utilisons un Thread qui va interroger en permanence la brique NXT pour connaître la position des moteurs et remonter l'information sous forme d'évènements en utilisant un pattern observer. L'état des capteurs sera, quant à lui, demandé quand nécessaire c'est-à-dire à chaque nouvelle position du joystick.





A chaque nouvelle position du joystick, la méthode onNewposition est appelée. Il faut alors traduire la valeur des angles fournie par les moteurs en coordonnées sur le panel.
Il n'est bien sûr pas possible de traduire les angles du moteur vers des coordonnées sans connaître les plages de valeurs renvoyées par le joystick. Pour cela, il faut donc calibrer le joystick en faisant un reset des moteurs, après avoir placé le joystick en haut à gauche et en faisant un relevé de mesure sur les moteurs après avoir placé le joystick en bas à droite. De cette façon, le joystick est calibré. Le reset et le relevé de mesure sont déclenchés par un appui sur le capteur de contact.
Voici ensuite le code du calcul de position :

```
public void onNewPosition(JoystickEvent e) {
    Graphics g = getGraphics();
    int w = image.getWidth();
    int h = image.getHeight();

    int x = (w*e.getAngleB()/joystickMaxX);
    int y = (h*e.getAngleA()/joystickMaxY);

    Point p = new Point(x,y);

    if (touch.isPressed()){
        draw(start, p);
    }

    int dist;
    try {
        dist = distance.getDistance()/2;
    } catch (Throwable npe) {
        return;
    }

    if (dist > 10) dist = 10;
    if (dist < 1) dist = 1;

    changeStroke(dist);

    start = p;
    repaintCursor(p);
}
```

Nous utilisons également, d'une part le capteur de contact comme bouton de souris en ne dessinant que lorsque le bouton est enfoncé. De plus, je fais varier l'épaisseur du trait en fonction de la distance renvoyée par le capteur ultrasonique.

Voici enfin le code du Thread de capture des positions des moteurs :

```
public void run() {
    this.running = true;
    while (running) {
        int A = Motor.A.getRotationCount();
        int B = Motor.B.getRotationCount();

        JoystickEvent event = new JoystickEvent(A, B);
        notifyListeners(event);
    }
}
```

A propos des performances

Au final, le PC interroge en permanence la brique via Bluetooth pour obtenir les valeurs des capteurs. L'inconvénient de cette méthode est que le processeur BlueCore passe en permanence du mode lecture au mode écriture perdant à chaque fois 30 ms. De ce fait, rien que pour lire la position des 2 moteurs, au moins 90 ms sont perdues, ce qui a tendance à donner des relevés de position trop espacés et rendre le dessin un peu naïf. Une solution serait de réaliser un programme sur la brique qui réalise toutes les mesures pour nous et envoie en une seule fois un message bluetooth contenant toutes les informations nécessaires : la position des moteurs, l'état du bouton et l'état du capteur ultrasonique. De ce fait, les changements de sens de communication sont limités.
Si vous souhaitez vous simplifier la tâche concernant la communication série via Bluetooth, je vous conseille d'utiliser la librairie iCommand développée au sein du projet LeJOS mais compatible avec le protocole du Firmware de LEGO (et uniquement celui-ci d'ailleurs). iCommand peut s'appuyer indifféremment sur BlueCove, Bluez ou RXTX pour la communication Bluetooth et supporte quasi complètement les commandes NXT. Cependant, si la librairie iCommand est plutôt bien faite, il lui manque cruellement la possibilité de faire l'envoi et réception de messages standard NXT via les commandes MessageRead et MessageWrite.

Conclusion

J'espère que cet article vous aura mis l'eau à la bouche sur le développement de robots NXT. Les possibilités offertes sont extrêmement nombreuses et peuvent même dépasser le cadre du jeu. Je vous invite à faire un tour sur le site mindtstorms.lego.com et découvrir le nombre incroyable de projets pouvant être réalisés. Et même si le kit souffre de quelques limitations, il n'y a rien là qui puisse gâcher le plaisir :)

Ressources

http://philohome.com/nxtjoystick/joystick.htm
http://mindstorms.lego.com/Overview/NXTreme.aspx
http://tau.ac.il/~stoledo/lego/btperformance.html
http://lejos.sourceforge.net/ - http://www.bluecove.org

■ **Fabrice Dewasmes**
Blog : http://fdewasmes.free.fr
Fabrice.dewasmes@gmail.com

Créez une interface iPod Touch en WPF

Vous avez toujours rêvé d'avoir une interface " à la iPod Touch " d'Apple (cela change de l'iPhone). Eh bien Programmez ! l'a fait ! Nous allons voir comment imiter l'interface graphique de l'ipod touch. C'est en regardant cet objet avec un œil d'informaticien que l'on aperçoit une ergonomie époustouflante et intuitive. Donner un look 'IPod touch' à nos applications Windows, amusant non ?

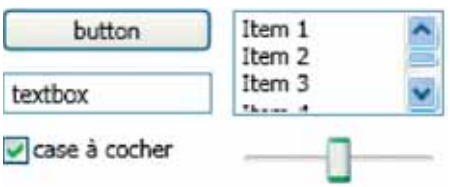
Tout d'abord un petit rappel sur WPF. Il est le successeur de GDI, une brique responsable de l'affichage de nos applications au niveau du système d'exploitation. Il succède mais ne remplace pas, En effet, il y a une rupture, alors que GDI ne prend en charge que le graphisme 2D, WPF gère la 2D, 3D, l'audio, la vidéo, le tout dans une API unifiée pour le développeur. Cette brique est utilisable pour concevoir des applications desktop (clients riches), Web (Xbap), Silverlight (xap). WPF est arrivé fin 2006, avec le Framework .Net 3.0.

WPF est entièrement vectoriel, ce qui permet de 'dessiner' à une échelle et de zoomer sans cet affreux effet escalier que l'on voit lorsque l'on zoome sur une image. Le rendu de WPF se fait au travers de DirectX, mais en tant que développeur il est inutile de connaître DirectX. Une des forces de ce framework est le couplage faible entre le visuel et le comportement de ce visuel. Une autre force de WPF réside dans la réalisation des écrans, puisque ceux-ci sont décrits en XML, dans un format nommé XAML. WPF est LA brique logicielle du framework 3.0 dédiée à l'affichage. La suite de l'article sera plus visuelle ! Maintenant que nous avons un aperçu des possibilités de WPF, voyons ce que l'on veut réaliser :

Si on regarde ces écrans avec un œil de développeur, on discerne plusieurs contrôles.

- Le bouton (plusieurs sous plusieurs formes)
  - Chaque lanceur d'application peut être vu comme un bouton
- L'interrupteur (case à cocher)
- La liste (Listbox contenant l'ensemble des interrupteurs dans notre exemple)
- La liste à défilement horizontal
  - Utilisée pour afficher la liste des applications disponibles
  - Utilisée pour faire défiler les photos
- La zone d'édition (pour la saisie utilisateur)

- La barre de volume (slider)

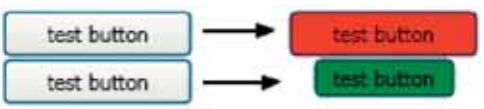


Finalement, les contrôles énumérés ci-dessus sont des contrôles standard que l'on trouve sous toutes les plates-formes. Et donc aussi parmi les contrôles fournis par WPF, en revanche leur look est plutôt Windows

Les styles et templates

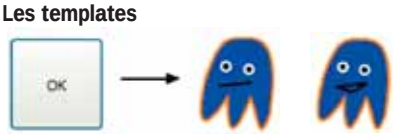
C'est l'une des forces de WPF grâce aux styles et aux templates, il est possible de séparer le visuel du comportement. Les styles sont utilisés pour définir les valeurs des propriétés liées au visuel (couleur de fond, couleur de la police, taille du contrôle, ...), les templates permettent de modifier l'affichage du contrôle (forme et contenu). C'est donc grâce aux styles/templates que nous allons imiter le visuel de l'IPod touch, le tout décrit en XAML. Résumons les styles et templates en images

Les styles




```
<Grid>
    <Grid.Resources>
      <Style x:Key="redbutton" TargetType="{x:Type Button}">
        <Setter  Property="Background" Value="Red"/>
      </Style>
      <Style x:Key="greenbutton" TargetType="{x:Type Button}">
        <Setter  Property="Background" Value="Green"/>
        <Setter Property="Width" Value="75"/>
        <Setter Property="Height" Value="21"/>
      </Style>
    </Grid.Resources>
    <StackPanel>
     <Button Style="{StaticResource redbutton}" Content="test button" Width
="100" Height="25"/>
      <Button Style="{StaticResource greenbutton}" Content="test button" />
    </StackPanel>
</Grid>
```

Les templates

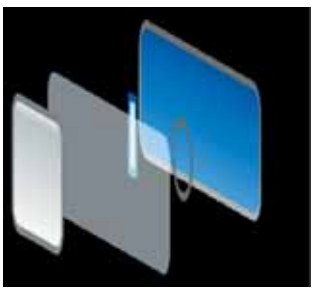


Les templates permettent de changer la forme. et lorsque l'on clique dessus il sourit.

```
<Grid>
  <Grid.Resources>
    <ControlTemplate x:Key="bTemplate" TargetType="{x:Type Button}">
      <Border BorderBrush="Gray" BorderThickness="0">
      <Viewbox VerticalAlignment="Center">
        <Grid>
        <Path x:Name="body" Data="M 240,250 C 200,375 200,250
 175,200 C 100,400 100,250 100,200 C 0,350 0,250 30,130 C
 75,0 100,0 150,0 C200,0 250,0 250,150 Z"
              Fill="Blue" Stroke="Orange" StrokeThickness="10"/>
        <Path Fill="Black" Stroke="White" StrokeThickness="10">
        <Path.Data>
          <GeometryGroup>
           <EllipseGeometry Center="95,95" RadiusX="15" RadiusY="15"/>
          <EllipseGeometry Center="170,105" RadiusX="15" RadiusY="15"/>
          </GeometryGroup>
        </Path.Data>
        </Path>

         <Path Stroke="Black" StrokeThickness="10"  StrokeEndLineCap=
"Round" StrokeStartLineCap="Round">
        <Path.Data>
          <GeometryGroup>
            <LineGeometry StartPoint="75,160" EndPoint="175,150"/>
          </GeometryGroup>
        </Path.Data>

        </Path>
       <Path x:Name="openmouse" Stroke="Black" StrokeThickness="10"
    StrokeEndLineCap="Round"  StrokeStartLineCap="Round"  Visibility=
"Collapsed">
        <Path.Data>
          <GeometryGroup>
            <LineGeometry StartPoint="75,160" EndPoint="125,180"/>
            <LineGeometry StartPoint="125,180" EndPoint="175,150"/>
          </GeometryGroup>
        </Path.Data>

        </Path>
        </Grid>
      </Viewbox>
      </Border>
      <ControlTemplate.Triggers>
        <Trigger Property="IsPressed" Value="true">
          <Setter TargetName="openmouse" Property="Visibility" Value=
"Visible"/>

          <Setter Property="RenderTransform">
            <Setter.Value>
              <ScaleTransform ScaleX="0.96" ScaleY="0.96"   CenterX=
"100" CenterY="100"/>
            </Setter.Value>
          </Setter>
        </Trigger>
      </ControlTemplate.Triggers>
    </ControlTemplate>
  </Grid.Resources>
  <StackPanel Orientation="Horizontal">
    <Button Width="70" Height="70" Content="OK"/>
    <Button Template="{StaticResource bTemplate}" Content="Ok" Height
="70" Width="70" FocusVisualStyle="{x:Null}"/>
  </StackPanel>
  </Grid>
```

Maintenant que nous connaissons le principe de fonctionnement des styles et templates, utilisons-les !
Vous l'aurez compris, nous utiliserons les styles/templates pour donner un look 'Ipod Touch' à nos contrôles standard. Un des contrôles très " parlant " pour l'utilisateur, est la case à cocher. La représentation qui en est faite sur l'Ipod est très intuitive

Pour dessiner ce contrôle on va empiler différentes couches (la technique du peintre)



Description des plans
(du fond vers l'avant)

Plan 1 : Fond du contrôle dont la couleur changera en fonction de son état
Plan 2 : les sigles
Plan 3 : voile qui sera visible lorsque le contrôle sera non modifiable (Disabled)
Plan 4 : le curseur qui masquera alternativement le sigle '0' ou 'I' en fonction de l'état (Checked/ Unchecked)
Nota : la représentation 3D est aussi réalisée avec WPF

Ce qui donne en XAML :

```
<Border x:Name="backBorder" Width="55.35" Height="15.90"
          BorderBrush="{StaticResource backBorderBrushUnchecked}"
           BorderThickness="1" CornerRadius="3" Background="{Static
Resource UnCheckedBrush}"
      HorizontalAlignment="{TemplateBinding HorizontalAlignment}"
      VerticalAlignment="{TemplateBinding VerticalAlignment}"
      >
   <Grid>
      <Grid.ColumnDefinitions>
         <ColumnDefinition />
         <ColumnDefinition />
      </Grid.ColumnDefinitions>
      <!—'O' ->
      <Ellipse .../>
      <!—'I' ->
      <Rectangle ...>
      <!— voile ->
      <Rectangle .../>
      <!— curseur ->
      <Border ...>
   </Grid>
</Border>
```

Et ainsi la case à cocher ne ressemble plus à une case à cocher standard, et elle conserve son comportement originel.
La même technique est utilisée pour réaliser la loupe. Celle-ci apparaît

quand sur l'Ipod touch, vous laissez votre doigt appuyé pendant une seconde sur une zone d'édition. Le principe ici pour zoomer sur un autre contrôle est d'utiliser un VisualBrush :

```
<Ellipse x:Name="loupe"Height="84.2"
Width="85.8" Stroke="Gray" StrokeThickness="1" >
  <Ellipse.Fill>
<VisualBrush x:Name="vbText"
  Visual="{Binding ElementName=tb}"
  Viewbox="105,2,10,10" ViewboxUnits="Absolute"/>
  </Ellipse.Fill>     1
```

On peut faire énormément de choses juste avec les styles et les templates, mais on ne fait pas tout. Prenons l'exemple de la liste.

La plupart du visuel est re-stylé en XAML, mais cela représente un peu plus de code, les éléments suivants on été re-stylés :

- Le scrollviewer (control contenu dans une listbox, permettant de faire défiler les éléments de celle-ci)
- les barres de défilement
  - cela comprend (les boutons haut et bas, et le curseur)
- Le ListboxItem (élément visuel hôte de la donnée à afficher)

Mais il y a une fonctionnalité qui nécessite du code. Chaque élément de la liste est séparé par une ligne, la visibilité de cette ligne dépend donc des données de la liste. En effet, le dernier élément de la liste ne possèdera pas de ligne de séparation.
Cette fonctionnalité est réalisée avec un " converter " :

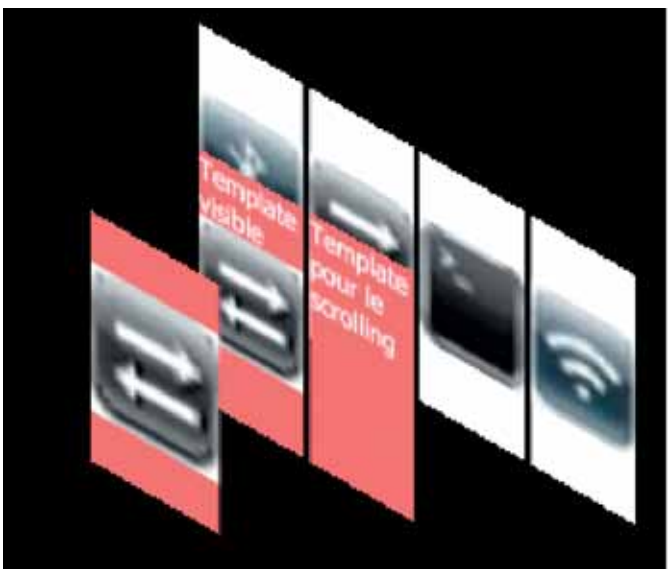
```
<Grid Background="#E1E1E1" Height="1" DockPanel.Dock="Top">
    <Grid.Visibility>
      <MultiBinding Converter="{StaticResource IndexToVisibilityConverter}">
          <Binding Path="."/>
          <Binding RelativeSource="{RelativeSource Mode=FindAncestor,
AncestorType={x:Type ItemsControl}}" Path="." />
      </MultiBinding>
    </Grid.Visibility>
</Grid>
```

Il reste un contrôle qui n'existe pas parmi les contrôles standard de WPF, c'est le PageScroller, et c'est certainement le plus utilisé sur IPod touch/Iphone. Il permet de faire défiler les photos, la liste des applications,… étant un contrôle à part entière, l'utilisation des styles/templates ne suffit pas, la majeure partie est codé en C#. Son architecture de fonctionnement est extrêmement simple :
Du fond vers l'avant

- Les données à afficher (ici une liste d'icônes)
- Template d'affichage
  - Deux instances du même template sont utilisées au moment du scrolling
- Le visuel (affiche le contenu du template visible)

Et pour le scrolling une animation sur les templates, pour les faire défiler dans un sens ou dans l'autre.

```
ThicknessAnimation anim = new ThicknessAnimation();
anim.From = new Thickness(0);
anim.To = new Thickness(-multiply * this.ActualWidth, 0, multiply * this.
ActualWidth, 0);
anim.Duration = new Duration(TimeSpan.FromMilliseconds(msDuration));
anim.AccelerationRatio = 0.6;
```



Conclusion

La technologie aujourd'hui nous permet vraiment de réaliser des interfaces graphiques en adéquation avec l'utilisateur. En effet WPF s'inscrit plus dans une démarche où l'application est faite pour l'utilisateur. Nous avons vu que le multicouche donne une grande souplesse de conception. WPF est un sujet extrêmement vaste, nous n'avons couvert ici qu'une toute petite partie de la 2D. Les templates et styles sont extrêmement puissants, il est donc impossible de développer pour WPF sans connaître ces notions. Les styles 'IPod touch' sont disponibles à cette adresse : http://www.codeplex.com/WpfIpodControls.
Aujourd'hui ces styles ne sont disponibles que pour WPF en version client riches et XBAP, à terme, ils le seront aussi pour Silverlight.
Pour en savoir plus sur WPF : http://msdn.microsoft.com/en-us/netframework/aa663326.aspx

Pierrick Gourlain
Architecte Technique - MVP Client Application

Mettre en œuvre la DB API de Python

La DB API de Python est une spécification visant à simplifier et unifier les scripts Python accédant à des bases de données. Nous faisons connaissance avec cette API à travers une de ses implémentations pour PostgreSQL: Psycopg

Le monde des bases de données est un monde qui manque d'unité. En dépit de l'existence du standard SQL, les disparités entre les systèmes de bases de données sont grandes. Vient ensuite le problème de la communication avec les systèmes. Il n'est pas rare de trouver plusieurs implémentations de librairies ou de modules d'extensions pour un langage devant travailler avec un système. Les implémentations, différant généralement notablement entre elles, ajoutent encore à la disparité. Dans le cadre du langage Python, la spécification DB API (http://www.python.org/dev/peps/pep-0249/) tente de remédier, au moins en partie, à cela. Le remède aurait pu être plus radical si ces spécifications laissaient moins de liberté aux implémenteurs et si certains de ceux-ci avaient fait preuve d'un peu plus de rigueur. Nous n'entrerons toutefois pas plus avant dans ce débat et nous nous contenterons d'étudier cette API intéressante au moyen d'une implémentation de qualité: Psycopg. Celle-ci est sous licence GPL-2 et peut être librement téléchargée à http://www.initd.org/. Psycopg travaille avec PostgreSQL, SGBDR favori de votre serviteur. Le fond de cet article reste toutefois valable pour tout autre système de base de données disposant d'une interface Python respectant les spécifications de l'API.

1 Installation de Psycopg

Psycopg vient sous la forme d'un module d'extension à Python écrit essentiellement en C. Installer Psycopg sur un système Linux relève de la simple formalité. On décompacte l'archive de Psycopg dans un répertoire, on se positionne dans ce répertoire, on acquiert les droits du super-utilisateur, puis l'on donne la fameuse commande:

```
python setup.py install
```

Tout se passera très bien sous Linux car, a priori, l'utilitaire d'installation trouvera le compilateur C Gcc. Sous Windows, les choses sont un peu moins immédiates. Si vous n'avez pas compilé vous-même votre Python à partir des sources mais que vous l'avez installé à partir de binaires, ce qui est en général le cas, vous devrez disposer sur votre Windows d'un compilateur compatible avec celui qui a été utilisé pour compiler les binaires de la distribution Python, Visual Studio 2003 en l'occurrence. Le compilateur compatible est tout simplement le port de Gcc sous Windows: mingw32. On le téléchargera à http://www.mingw.org/, le plus simple étant de laisser l'outil d'installation en ligne s'occuper de tout. L'installation effectuée et en supposant que celle-ci soit située sous C:\Mingw, on veillera à intégrer deux répertoires dans le PATH: C:\MinGW\bin et C:\MinGW\libexec\gcc\ mingw32\3.4.5. Ensuite, il faut que l'outil d'installation de module sache qu'il doit utiliser Mingw32. Pour cela, le mieux est de créer, s'il n' existe pas déjà, un fichier distutils.cfg dans le répertoire Lib\distutils de votre Python. Dans ce fichier on écrira :

```
[build_ext]
compiler=mingw32
include-dirs=C:\MinGW\lib\gcc\mingw32\3.4.5\include
```

Ce n'est pas tout. Il tombe sous le sens que les fichiers en-têtes et les librairies de PostgreSQL doivent être accessibles au compilateur. Pour cela, le script de construction du module s'appuie sur l'outil de configuration pg_config de PostgreSQL. pg_config doit donc être lui aussi pointé par le PATH de votre système, ce qui revient à dire que le répertoire bin de PostgreSQL doit être intégré au PATH. Quand tout est prêt, vous pouvez saisir la commande python setup.py install. Vous pourrez vérifier ensuite que l'installation est correcte en saisissant dans l'interpréteur interactif de Python :

```
import psycopg2
```

Pour la suite de cet article, nous considérons qu'une base de données nommée Test a été auparavant crée sous PostgreSQL.

2 L'interface du module

Selon les spécifications, l'interface du module doit exposer une fonction connect pour la connexion, des exceptions pour la gestion des erreurs (le lecteur voudra bien se reporter au document des spécifications pour en connaître le détail), ainsi que trois variables informant des caractéristiques de l'implémentation :

Nom	Description
apilevel	Niveau des spécifications implémentées: 1.0 ou 2.0. Si apilevel n'existe pas le niveau est 1.0.
threadsafety	0 Le module ne peut être partagé par des threads. 1 Des threads peuvent partager le module, mais pas les connexions. 2 Des threads peuvent partager le module, les connexions, mais pas les curseurs. 3 Des threads peuvent partager module, connexions et curseurs.
paramstyle	Syntaxe de passage de paramètres dans les requêtes SQL paramétrées, parmi 'qmark', 'numeric', 'named', 'format' et 'pyformat'

Munis de ces informations, nous pouvons nous connecter à notre base Test et voir les caractéristiques de Pyscopg:

```
# -*- coding: Latin_1 -*-

import psycopg2
import sys

print psycopg2.apilevel
print psycopg2.threadsafety
print psycopg2.paramstyle

try:
    cxn = psycopg2.connect(host='localhost', database='Test', \
          user='fred', password='********')
    # ci-dessous: ceci est une extension psycopg
    print "Connexion ouverte. Encodage: ", cxn.encoding
    cxn.close()
```

Pour les .Net *addicted*

Demandez-le à votre marchand de journaux

```
except psycopg2.OperationalError:
    print "Impossible de se connecter à la base de données. Abandon."
    sys.exit(1)
```

Ce script nous affiche fièrement sur la console :

```
2.0
2
pyformat
```

Nous avons donc une implémentation 2.0, un bon support des threads et nous savons que nous devons passer les paramètres sous formes de clés de dictionnaires, un des styles préconisé par les spécifications. Nous regrettons toutefois que celle ci n'imposent pas ici un seul style pour tout le monde. Cette lacune empêche d'avoir du code directement portable d'un SGBDR à l'autre. Les exemples suivants illustreront ce style pyformat. On voit encore que la fonction connect reçoit des arguments mots-clés. Ceux-ci sont clairement spécifiés. On ne peut que regretter au passage que le seul adaptateur pour MySQL, MySQLdb, utilisé fasse fi de ces spécifications et utilise le mot-clé db au lieu de database.

3 L'objet connexion

Celui-ci propose 4 méthodes:

Nom	Description
close()	Ferme la connexion
commit()	Valide la transaction en cours
rollback()	Annule la transaction en cours
cursor()	Obtient un objet curseur (cursor)

Les requêtes SQL sont exécutées via des méthodes de l'objet curseur dont nous parlerons plus loin. Avec l'API tout se passe implicitement au sein d'une transaction. L'implémentation doit si besoin se charger de désactiver le mode auto-commit de certaines bases de données. Le fait que les transactions soient implicites implique que l'on n'utilise jamais les instructions SQL BEGIN ou START TRANSACTION. L'objet curseur se chargera de le faire à la première exécution de requête ou à l'exécution de la première requête suivant un appel à la méthode commit.

4 L'objet curseur

Son rôle est l'exécution des requêtes SQL et la récupération des résultats. Il expose de nombreuses méthodes et propriétés. En voici quelques-unes.

Nom	Description
rowcount	Propriété indiquant le nombre de lignes retournées par une requête.
close()	Libère les ressources relatives au curseur.
callproc(procname[,paramètres])	Exécute une procédure stockée.
execute(operation[,paramètres])	Exécute une requête.
executemany(operation,seq_de_paramètres)	Exécute séquentiellement une requête paramétrée.
fetchone()	récupère la prochaine ligne de résultats.
fetchall()	Récupère toutes les lignes de résultats.

L'utilisation du curseur est directe et intuitive tant que l'on travaille avec des types simples. L'exemple suivant illustre ceci en créant une table à trois colonnes, y insère des données, récupère la totalité de celles-ci puis récupère ensuite la partie répondant à une instruction WHERE :

```
# -*- coding: Latin_1 -*-
import psycopg2
import sys

try:
    cxn = psycopg2.connect(host='localhost', database='Test', \
                           user='fred', password='********')
    print "Connexion ouverte. Encodage: ", cxn.encoding
except psycopg2.OperationalError:
    print """Impossible de se connecter
          à la base de données. Abandon."""
    sys.exit(1)

try:
    curseur = cxn.cursor()
    curseur.execute("""
    CREATE TABLE Produits
    (
          identifiant_produit serial PRIMARY KEY,
          description text,
          prix float
    )  """)
    #cxn.commit() # optionnel ici

    curseur.execute("""INSERT INTO Produits (description, prix)
          VALUES ('Dessert', 4.20)""")
    data = {'nom':"Viande", 'tarif':6.15}
    curseur.execute("""INSERT INTO Produits (description, prix)
    VALUES (%(nom)s, %(tarif)s)""", data)

    data1 = {'nom':"Salade", 'tarif':1.20}
    data2 = {'nom':"Fruits", 'tarif':3.15}
    curseur.executemany("""INSERT INTO Produits (description, prix)
    VALUES (%(nom)s, %(tarif)s)""", [data1, data2])
    cxn.commit()

except psycopg2.ProgrammingError, m:
    print m
    cxn.rollback()

try:
    curseur.execute("SELECT * FROM produits")
    print curseur.description
    print "resultats SELECT"
    rows = curseur.fetchall()
    for row in rows:
          print row

    print "Resultats SELECT & WHERE"
    curseur.execute("""SELECT * FROM produits
          WHERE prix > %(seuil)s""", {'seuil':2.0})
    rows = curseur.fetchall()
    for row in rows:
```

```
        print row
except psycopg2.ProgrammingError, m:
    print m

curseur.close()
cxn.close()
```

Le point intéressant de ce code se trouve dans l'insertion des données de la table. La première ligne est insérée par une requête SQL classique. La deuxième ligne utilise une requête paramétrée. C'est ici que la syntaxe particulière pyformat entre en action. On crée d'abord un dictionnaire. Les clés de ce dictionnaire doivent être alors utilisées dans la chaîne de formatage au format Python de la requête paramétrée. On passe alors cette chaîne en argument à la méthode execute, ainsi que le dictionnaire créé juste avant. L'API se chargera de transformer le tout en requête SQL valide. Enfin les troisième et quatrième lignes de la table sont insérées en une seule fois via un appel à executemany qui exécute séquentiellement une requête paramétrée. Le principe est le même que ce que nous venons de voir. Simplement le deuxième argument de executemany doit être une liste ou un tuple de dictionnaires. Le même principe est encore appliqué pour la constitution de la requête incluant l'instruction WHERE.

5 Travail avec des types complexes

Comment faire avec des types complexes et plus généralement comment mettre en correspondance les types Python avec les types reconnus par le SGBDR ? Une chaîne de caractères Python peut être appelée à devenir CHAR, VARCHAR, TEXT, ou même une donnée binaire dans PostgreSQL. Un nombre peut être appelé à devenir un NUMERIC. Le langage Python ne permet pas d'établir nativement ces subtiles distinctions. De même, comment insérer un type DATE ou TIME à partir de Python ? Pour répondre à ce problème, les spécifications définissent un jeu de classes donc les constructeurs permettront de construire le type directement, par exemple pour un type SQL DATE, ou de préciser le type SQL exprimé par une chaîne Python. Par exemple TEXT ou VARCHAR. Voici quelques-unes de ces classes :

Nom	Description
Date	Construit un type DATE
Time	Construit un type TIME
BINARY	Permet de préciser BLOB, RAW, etc
STRING	Permet de préciser CHAR,VARCHAR,TEXT, etc
NUMBER	Permet de préciser NUMERIC ou un autre format.

Lorsqu'on lit les spécifications, l'utilisation pratique de ces classes n'est pas d'une clarté limpide. Voici donc un dernier exemple qui devrait clarifier les choses. Le code crée une nouvelle table et y insère des données de types SQL TEXT, NUMERIC, DATE et TIME.

```
# -*- coding: Latin_1 -*-
import psycopg2
import sys

try:
    cxn = psycopg2.connect(host='localhost', database='Test', \
        user='fred', password='********')
    print "Connexion ouverte. Encodage: ", cxn.encoding
except psycopg2.OperationalError:
    print """Impossible de se connecter
        à la base de données. Abandon."""
    sys.exit(1)

try:
    curseur = cxn.cursor()
    curseur.execute("""
    CREATE TABLE LesTypes
    (
    id serial PRIMARY KEY,
    baratin text,
    valeur numeric,
    jour date,
    heure time
    ) """)

    data = {'t':"Programmez!",
        'v':6.15,
        'j':psycopg2.Date(2008,2,20),
        'h':psycopg2.Time(10, 59, 30)
        }
    curseur.execute("""INSERT INTO LesTypes (baratin, valeur,
     jour, heure) VALUES (%(t)s, %(v)s, %(j)s, %(h)s)""", data)

    data = {'t':psycopg2.STRING("Abonnez vous", "TEXT")
        'v':psycopg2.NUMBER("15.0", "NUMERIC"),
        'j':psycopg2.Date(2008,2,21),
        'h':psycopg2.Time(11, 59, 30)
    }
    curseur.execute("""INSERT INTO LesTypes (baratin, valeur,
    jour, heure) VALUES (%(t)s, %(v)s, %(j)s, %(h)s)""", data)
    cxn.commit()

except psycopg2.ProgrammingError, m:
    print m
    cxn.rollback()

try:
    curseur.execute("SELECT * FROM LesTypes")
    print curseur.description
    print "resultats SELECT"
    rows = curseur.fetchall()
    print rows
    for row in rows:
        for item in row:
            print item

except psycopg2.ProgrammingError, m:
    print m

curseur.close()
cxn.close()
```

Frédéric Mazué
fmazue@programmez.com

Utiliser Spring, Hibernate et JMS : attention aux transactions !

Lorsque l'on est amené à faire une vraie application devant mettre en œuvre à la fois de la publication/consommation de messages et des mises à jour en base de données, et qu'en plus on aimerait faire tout ça très orienté POJO en utilisant Spring, mieux vaut se montrer prudent quant à l'utilisation des transactions.

En fait, le problème est essentiellement dû à un manque de connaissances des développeurs sur les transactions. Donc avant toute chose, un peu de théorie.

1 XA, 2PC, distributed transactions, JTA, JTS et les autres

Tout d'abord qu'est-ce qu'une transaction ? Une transaction permet de réaliser plusieurs opérations dans une seule unité de traitement. La transaction n'est valide que si toutes les opérations se sont déroulées correctement : c'est le *commit.* Si l'une des opérations vient à échouer, toutes les autres sont annulées, laissant le système dans son état initial : c'est le *rollback.* Jusque là, la plupart des développeurs sont au point avec ces notions.

La transaction telle que décrite précédemment est souvent utilisée pour ne gérer que les opérations effectuées sur une seule ressource (Base de données, broker JMS, ...). Dans ce cas, le gestionnaire de la ressource (ou resource manager) est l'unique sous-système en charge de la cohérence des données. On parle dans ce cas de transactions locales. (Fig.1)

Si par contre, les opérations qui doivent être effectuées sont gérées par divers gestionnaires de ressources, on parlera alors de transactions globales. Un bon exemple de cela est la nécessité d'avoir une transaction qui s'occupe de mettre à jour des données situées sur deux bases de données différentes ou encore une transaction devant mettre à jour une base de données et dans le même temps envoyer un message JMS. Pour que cela puisse se faire, il a fallu que plusieurs ISVs constituent un consortium (Open Group) pour définir une "spécification" (X/Open Distributed transaction processing) permettant de définir comment synchroniser les différents gestionnaires de ressources. Cette spécification définit une interface XA que doit implémenter chaque gestionnaire de ressources afin de pouvoir être enrôlé dans une transaction globale. De cette façon, il sera possible à un gestionnaire de transaction de coordonner l'ensemble des gestionnaires de ressources. Cette coordination a lieu en deux temps : un premier appel est lancé à tous les gestionnaires de ressource afin qu'ils se tiennent prêts, leur permettant ainsi de faire le locking nécessaire à l'isolation de la transaction ; un second appel est ensuite émis afin de réaliser effectivement les opérations. Ces deux appels constituent le "two phase commit" (2PC). (Fig.2) En java, il existe deux API parfois confondues permettant la mise en œuvre de cette mécanique :

• **JTA** est une API qui permet d'une part à une application de communiquer avec le gestionnaire de transaction afin de permettre de lui passer

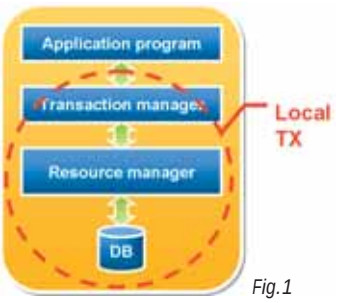


Fig.1

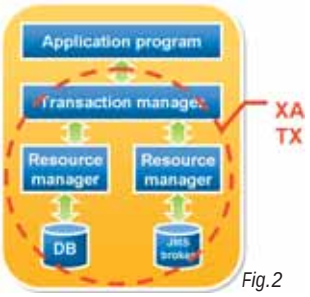


Fig.2

les ordres de démarrage, arrêt ou annulation de transaction et d'autre part au gestionnaire de transaction de communiquer avec les gestionnaires de ressources. De plus, JTA définit un mapping du protocole standard XA. JTA est une API de haut niveau servant de liant à l'ensemble des participants d'une transaction (trait rose sur le schéma ci-contre).

• **JTS** est une API que doit implémenter un gestionnaire de transaction s'il veut supporter JTA et pouvoir participer dans des transactions globales éventuellement distribuées. (Fig.3)

Ainsi, on peut dire que JTA est la partie émergée de l'iceberg et JTS la partie immergée.

Les **transactions distribuées** sont une espèce particulière de transaction globale pour laquelle tous les gestionnaires de ressources ne se trouvent par forcément dans le même espace mémoire et donc pour lesquels il est nécessaire de transférer sur le réseau les commandes de synchronisation XA. Ainsi, il est important de ne pas confondre XA et distribué : il est tout à fait possible de faire du XA sans faire du distribué. Le serveur d'application Jboss, par exemple, supporte les transactions XA mais pas les transactions distribuées.

2 La problématique

Pour des applications critiques devant assumer une charge assez conséquente, il est fréquent d'avoir recours à JMS. Mais bien souvent, l'envoi ou la réception d'un message s'inscrit dans un processus demandant une mise à jour de données. Mais que se passe-t-il si pour une raison ou une autre la mise à jour de la donnée échoue ? Le message a été consommé et potentiellement aucun traitement n'y aura été apporté. De la même façon : que se passe-t-il si une mise à jour de données est effectuée et, suite à cette mise à jour, nous souhaitons envoyer un message mais que l'envoi de celui-ci échoue ? La mise à jour est faite en base mais d'autres sous-systèmes n'en ont pas été notifiés et le reste du traitement n'a pas lieu. Très inconfortable. La seule solution dans ce cas est d'enrôler les deux ressources dans une transac-

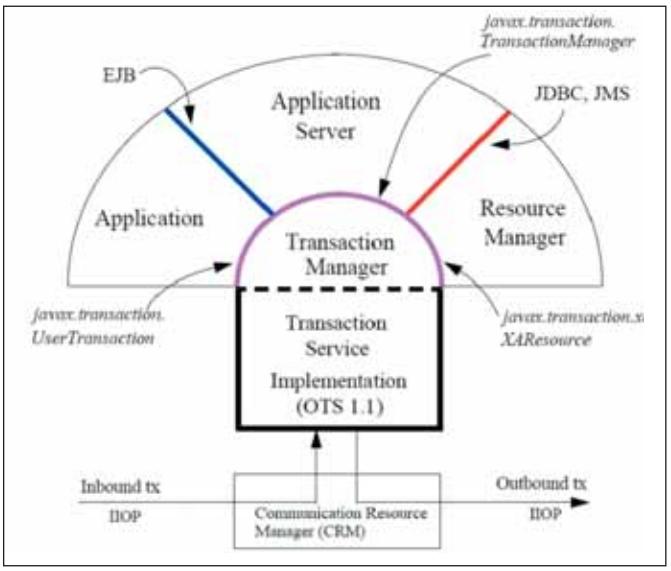


Fig.3

tion. Si vous avez bien suivi la première partie de l'article, une transaction locale suffit-elle ? Non bien sûr, il nous faut un gestionnaire de transaction, capable de faire du XA !
Mais de surcroît, comme vous êtes un programmeur à la pointe et que vous profitez toujours des dernières avancées, vous vous évitez du travail en utilisant Hibernate et comme vous êtes, pour couronner le tout, quelqu'un qui préfère utiliser des POJOs, vous utilisez Spring. Voyons comment faire fonctionner tout ceci sur un serveur d'application JBoss.

3 Posons les bases

Créons tout d'abord un EJB stateless en utilisant les classes utilitaires de Spring afin de renforcer l'utilisation de POJOs derrière une façade de EJB. Cet EJB va nous servir à démarquer les transactions en utilisant ses capacités CMT (Container Managed Transactions). Voici un extrait de l'ejb-jar.xml montrant la configuration transactionnelle de l'EJB :

```
<enterprise-beans>
   <session >
      <ejb-name>SampleEjbBean</ejb-name>
[…]
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
   </session>
</enterprise-beans>
[…]
<assembly-descriptor>
   <container-transaction >
      <method >
         <ejb-name>SampleEjbBean</ejb-name>
         <method-intf>Remote</method-intf>
         <method-name>sampleTransactedMethod</method-name>
         <method-params>
            <method-param>boolean</method-param>
         </method-params>
      </method>
      <trans-attribute>Required</trans-attribute>
   </container-transaction>
</assembly-descriptor>
```

Au sein d'une méthode de cet EJB nous allons émettre un message JMS, puis faire un ajout en base via Hibernate. Un booléen passé en paramètre nous permettra de déclencher un rollback de la transaction simplement en jetant une EJBException.

```
public void sampleTransactedMethod(boolean fail) {
   if (jmsTemplate != null) {
      jmsTemplate.send(new MessageCreator() {
         public Message createMessage(Session session)
         throws JMSException {
            TextMessage message = session.createTextMessage("hello
 queue world");
         return message;
         }
      });
   } else {
      throw new EJBException();
   }

   delegate.sampleTransactedMethod(fail);

   if (fail) {
      throw new EJBException();
   }
}
```

Le corps de la méthode du delegate est particulièrement simple :

```
public void sampleTransactedMethod(boolean fail) {
   Person person = new Person();
   person.setFirstname("Fabrice");
   person.setLastName("Dewasmes");
   dao.save(person);
}
```

L'objet Person est un objet persisté par Hibernate. L'objet de cet article n'étant pas d'expliquer comment utiliser Hibernate, je vous laisse le soin de consulter les nombreux tutoriaux disponibles sur Internet. Il est juste bon de mentionner ici que nous utilisons la classe HibernateDaoSupport de Spring afin de simplifier au maximum le code et la configuration.

4 Le cœur du problème

Jusque là nous n'avons rien fait d'extraordinaire. Voyons à présent la véritable partie intéressante. Afin que tout fonctionne comme prévu, il est nécessaire que la création de l'objet Person et l'émission du message JMS soient enrôlés dans la même transaction globale. Pour cela, la seule façon d'y parvenir est d'utiliser un gestionnaire de transaction JTA. Voyons d'abord pour Hibernate. L'implémentation du pattern DAO a besoin d'une session hibernate que nous allons obtenir comme ceci :

```
<bean id="PersonDaoHibernateImpl" class="jms.dao.PersonDaoHiber
nateImpl">
   <property name="sessionFactory" ref="sessionFactory" />
</bean>
```

Le bean sessionFactory va, quant à lui, faire le liant. Il s'appuie d'une part sur la dataSource, ce qui est normal, mais d'autre part sur un gestionnaire de transaction. Ce gestionnaire de transaction peut être dans Spring, soit HibernateTransactionManager, soit JtaTransactionManager. Sur le web, vous trouverez la plupart du temps des exemples avec le premier mais ATTENTION : celui-ci est un gestionnaire de transactions LOCALES et par conséquent inapproprié pour nous. Dans la mesure où nous faisons tourner le code dans un serveur d'applications J2EE il nous est possible d'utiliser le gestionnaire de transaction JTA du serveur. Bien souvent, il est possible de l'utiliser avec un simple lookup JNDI. Dans JBoss le gestionnaire de transaction est disponible sous java:/TransactionManager. Il va donc nous falloir utiliser la configuration suivante :

```
<bean id="transactionManager"
   class="org.springframework.transaction.jta.JtaTransactionManager">
   <property name="userTransactionName" value="UserTransaction" />
   <property name="transactionManagerName" value="java:/Trans
actionManager" />
</bean>

<bean id="sessionFactory"
      class="org.springframework.orm.hibernate3.LocalSessionFactory
Bean">
   <property name="configLocation" value="classpath:hibernate.cfg.
xml" />
   <property name="dataSource" ref="dataSource" />
   <property name="schemaUpdate" value="false" />
   <property name="jtaTransactionManager">
      <bean factory-bean="transactionManager"
         factory-method="getTransactionManager" />
   </property>
</bean>
```

De ce fait, la sessionFactory Hibernate va gérer les transactions en utilisant le gestionnaire de transaction JTA du serveur, ce qui tombe bien puisque nous allons utiliser ce même gestionnaire pour réaliser les transactions JMS. Pour terminer, nous allons utiliser les fonctionnalités AOP de Spring pour permettre une gestion déclarative des transactions sur le DAO :

```
<tx:advice id="txAdvice" transaction-manager="transactionManager">
   <tx:attributes>
      <tx:method name="*" />
   </tx:attributes>
</tx:advice>

<aop:config>
   <aop:pointcut id="daoOperation"
      expression="execution(* jms.*DaoHibernateImpl.*(..))" />
   <aop:advisor advice-ref="txAdvice"
      pointcut-ref="daoOperation" />
</aop:config>
```

Ici, nous interceptons les appels aux méthodes du DAO pour tisser un advice utilisant le gestionnaire de transactions de JBoss pour propager les transactions sur les méthodes du DAO (par défaut le tag <tx:method> utilise comme attribut de transaction PROPAGATION_REQUIRED). Pour JMS, les choses sont plus simples. La spécification JMS ne force pas les fournisseurs à supporter les transactions XA mais s'il le fait, alors il doit exposer son support JTA à travers des ressources spéciales et notamment une XAConnectionFactory. Fort bien, JBoss expose justement une connection factory s'appelant XAConnectionFactory : ne vous en servez pas ! Car elle est utilisée en interne par l'implémentation JTA de JBoss. Au lieu de cela, il faut utiliser la connection factory bindée sous le nom java:/JmsXA (ce qui, soit dit en passant, est TRES peu documenté). Voilà donc la configuration nous permettant de créer un JmsTemplate Spring qui enrôlera l'envoi de messages auprès du gestionnaire de transactions JTA :

```
<jee:jndi-lookup id="appJmsDestination" jndi-name="queue/A" />
<jee:jndi-lookup id="queueConnectionFactory" jndi-name="java:/JmsXA" />

<bean id="appSenderTemplate"
   class="org.springframework.jms.core.JmsTemplate">
   <property name="connectionFactory" ref="queueConnectionFactory" />
   <property name="defaultDestination" ref="appJmsDestination" />
   <property name="messageTimestampEnabled" value="false" />
   <property name="messageIdEnabled" value="false" />
</bean>
```

Il ne nous reste plus qu'à tester tout ceci. Créez un petit MDB qui va simplement afficher le contenu du message JMS d'une part et d'autre part créez un classe cliente qui va appeler notre méthode sampleTransactedMethod avec comme paramètre false puis true en vérifiant le contenu de la base de données :

```
public static void main(String[] args) {
ClassPathXmlApplicationContext context = new ClassPathXmlApplication
Context("test-applicationContext.xml");
SampleBizInterface test = (SampleBizInterface) context.getBean("my
Component");
System.out.println("Before test : " + test.getAllPersons().size() + "persons in DB");
System.out.println("calling transacted method without rolling back");
test.sampleTransactedMethod(false);
System.out.println( test.getAllPersons().size() + "persons in DB");
}
```

Constatez que dans le cas où la transaction n'est pas annulée, l'insertion en base a bien lieu et le message est bien lu par le MDB. En revanche, si la transaction est annulée, on ne constate ni ajout en base ni consommation de messages. Un autre test 'amusant' à faire est d'utiliser HibernateTransactionManager comme gestionnaire de transaction pour Hibernate. Il suffit pour cela de modifier un peu la configuration Spring et relancer le test et vous verrez comme les transactions sont devenues indépendantes.
Afin de tester facilement tout cela vous pouvez télécharger les sources de l'exemple sur le site http://fdewasmes.free.fr.

Fabrice Dewasmes
Java & Open Source Department Manager
PragmaConsult - www.pragmaconsult.lu

L'INFORMATION du DÉCIDEUR

Choisir, déployer, exploiter les softwares

Abonnez-vous au seul magazine offrant aux responsables informatiques une information et des témoignages focalisés sur le logiciel en entreprise.

Dans chaque numéro,
les tendances, les dossiers, les interviews, les témoignages, les avis d'expert dans tous les domaines du logiciel professionnel :

- *Les SSII, des métiers et du recrutement ;*
- *L'administration, les réseaux ;*
- *La sécurité, la sauvegarde ;*
- *La gestion des projets, les méthodes, le développement ;*
- *Les progiciels, ERP, BI et SGBD…*

L'actualité au quotidien :
• Sécurité • Projets et développement
• Administration
• Progiciels

Les Cas Clients

Prochainement : **Vidéos** *(Actualité et Cas Clients)*

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 : 25€ *au lieu de 30€, prix au numéro* *(Tarif France métropolitaine)* **- Autres destinations :** *CEE et Suisse : 30€ - Algérie, Maroc, Tunisie : 33€ , Canada : 39,50€ - Dom : 38€ Tom : 50€*

6 numéros. Prochaines parutions : N°4 Octobre - N°5 Novembre/Décembre - N°6 Janvier/Février 2009 - N°7 Mars/Avril - N°8 Mai/juin - N°9 Juillet/Août/Septembre

☐ 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

SL003

Grails : l'autre manière de faire du Java

Ruby on Rails est un framework pour le développement d'applications Web qui implémente le pattern Modèle-Vue-Contrôleur. Sa particularité repose sur l'approche agile qu'il apporte aux développeurs. RoR entre ainsi dans la catégorie dont il est la figure de proue : les langages RAD.

Le framework vient en concurrence avec les technologies déjà présentes dans le milieu professionnel et plus particulièrement Java EE. Cette dernière étant jugée trop lourde, les concepteurs ont décidé de reprendre les meilleures idées de Rails pour bâtir un framework agile pour la plate-forme Java qui reste malgré tout incontournable en entreprise.

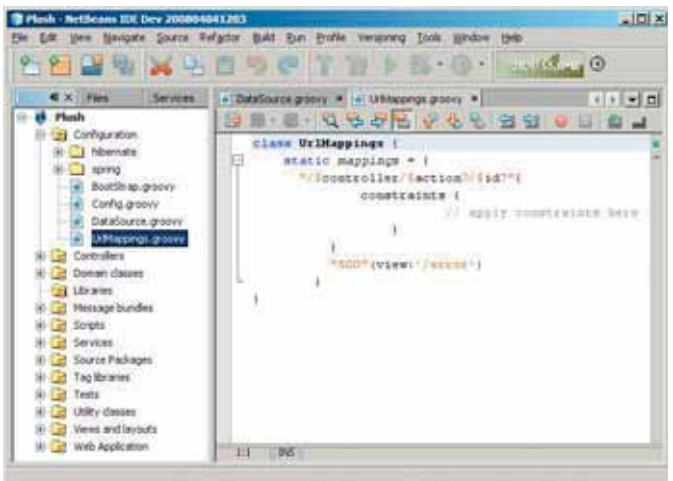
Architecture et fonctionnalités

Lorsque les créateurs de Grails ont abordé le développement du framework, ceux-ci avaient un certain nombre d'objectifs en tête. Tout d'abord, Grails devait améliorer la productivité de la plate-forme Java dans un contexte entreprise. Cela ne devait pas pour autant empêcher la réutilisation du code Java ainsi que des compétences des équipes de développeurs, notamment sur les outils tels que Spring et Hibernate. C'est pour cette raison que le framework propose une interface simple permettant une intégration complète de Grails avec les frameworks J2EE. Grails se définit notamment par un support puissant de la persistance, ainsi qu' une construction des vues simplifiée, notamment par l'utilisation de codes groovy (pages GSP). Le support d'Ajax est natif avec Grails et de nombreuses possibilités de personnalisation et d'extensions sont possibles. C'est dans cet esprit que Grails intègre de nombreuses librairies, dont OpenRico, Prototype et la librairie de Yahoo! Ce qui offre de quoi développer des applications Web 2.0 avec agilité !
Au delà du framework, ses créateurs ont convenu dès le départ de la nécessité d'une documentation bien fournie ainsi que d'un ensemble d'applications de démonstrations mettant en œuvre le framework. Tout comme Ruby On Rails, ici, nous disposons d'un environnement complet incluant un serveur web ainsi qu'un mécanisme de rechargement automatique des ressources. Grails est donc simple à apprendre et augmente la productivité.

Le Langage Groovy

Bien qu'étant un langage de script, Groovy repose sur une syntaxe proche de Java. Son apprentissage est donc très rapide et un développeur Java n'aura pas besoin de fournir un travail important pour s'y adapter. Cependant, Groovy est bien plus concis que Java et l'approche développement est différente et plus près des langages agiles. Le code suivant (en Java) peut ainsi être bien plus court en Groovy :

```
import java.util.*;

public class AStrings {
 public static void main(String args[]) {
   List aList = new ArrayList();
   aList.add("Jim");
   aList.add("Joan");
   aList.add("James");

   List aWords = new ArrayList();
   for (String item : aList) {
     if (item.contains("a")) {
       aWords.add(item);
     }
   }
   for (String item : aWords) {
     System.out.println(item);
   }
 }
}
```



En Groovy :

```
aList = ["Jim", "Joan", "James"]
aWords = aList.findAll { it.contains("a") }
aWords.each { println it }
```

Groovy est particulièrement novateur et propose quelques fonctionnalités qui devraient apparaître dans Java 7. On peut citer notamment l'intégration des langages à balises, dont XML et XHTML. Cela permet de travailler directement sur un document sans avoir recours à une API.

```
import groovy.xml.MarkupBuilder
    def myXMLDoc = new MarkupBuilder()
    myXMLDoc.workbook {
      worksheet(caption:"Employees") {
        row(fname="John", lname="McDoe")
        row(fname="Nancy", lname="Davolio")
      }
```

```
    worksheet(caption:"Products") {
      row(name="Veeblefeetzer", id="sku34510")
      row(name="Prune Unit Zappa", id="sku3a550")
    }
  }
  println myXMLDoc
```

Grails : un framework

Grails est en réalité une couche qui vient se superposer sur la plate-forme Java. Cela signifie qu'il est simple d'intégrer des librairies Java, ainsi que des frameworks et du code existant. La principale fonctionnalité apportée par Grails est la possibilité d'intégrer de manière transparente des classes mappées avec Hibernate. Cela renforce l'aspect non intrusif du framework : aucune recompilation ni reconfiguration n'est nécessaire. Une conséquence de cette technique est la possibilité d'opérer un réel échafaudage pour la couche de persistance Java via les actions CRUD.

Environnement de développement

La majorité des outils proposent le support et l'intégration du langage Groovy. Netbeans dispose ainsi d'un plug-in qui permet d'utiliser pleinement la puissance de l'IDE au service du codage d'applications agiles. Le plug-in apporte un wizard pour la création de projets Groovy et Grails ainsi que la complétion du code.

Afin de pouvoir installer le plug-in, vous devrez vous procurer Netbeans dans sa version de développement (nightly build). Cette version est une bêta qui intègre de nouvelles fonctionnalités. Dans notre cas, c'est le support de Grails qui nous intéresse. Le plug-in s'installe par le menu Tools > Plugin. Vous n'avez plus qu'à faire une recherche de Groovy dans la liste des plug-in disponibles et lancer l'installation. Les utilisateurs de Linux devront vérifier qu'ils disposent bien du JDK de Sun. En effet, Groovy ne fonctionnera pas avec l'implémentation GNU de Java (GCJ). Pour le vérifier, utilisez la commande suivante :

```
java -version
```

Dans le cas où il ne s'agit pas de la version de Sun. Vous n'avez plus qu'à télécharger le SDK et à l'installer.

Petite mise en pratique

Commençons par créer notre projet. Pour cela, nous allons simplement utiliser le wizard mis à notre disposition. Il suffit de choisir un nom et un emplacement au projet. Evitez les chemins contenant des espaces ou autres bizarreries. Groovy ne supporte pas les chemins UNC. Comme avec Ruby On Rails, l'arborescence va être générée avec un ensemble de fichiers de configuration (Spring, datasources…) ou autres feuilles de styles et pages d'erreurs. C'est ce que l'on appelle le scaffolding ou échafaudage en bon français. A partir de là, il est possible de lancer l'application. Pour cela, il suffit de faire un clic droit sur le projet, puis Grails > Run application. Vous ne pourrez pas utiliser le lanceur d'application de Netbeans avec ce plug-in. Il ne sera donc pas possible de debugger vos applications avec les outils, pourtant très utiles, de profiling.

Construisons une application

Afin de mettre en œuvre Grails, nous allons créer une simple application qui permet d'accéder aux actions CRUD sur un modèle de données quelconque. Pour l'exemple, nous allons nous contenter d'une classe Player disposant d'attributs classiques : nom, prénom etc. Créez cette classe dans le dossier Domain classes. On remarque que l'implémentation de la persistance est totalement transparente.

```
class Player {
    Long id
    Long version

    String name
    String firstname
    Date birthdate
    String email
}
```

Elément principal de la logique de l'application, le contrôleur permettra d'arbitrer les branchements. Il a pour rôle d'appeler la vue correspondante à chaque URL ainsi que les traitements nécessaires au bon déroulement de l'application. Ce code est entièrement généré par l'utilisation du plug-in, il suffit de faire un clic droit sur la classe puis de cliquer sur generate all. Les méthodes suivantes seront générées automatiquement : index, list, show, delete, edit, update, create et save.

```
class PlayerController {

    def index = { redirect(action:list,params:params) }

    // the delete, save and update actions only accept POST requests
    def allowedMethods = [delete:'POST', save:'POST', update:'POST']

    def list = {
        if(!params.max) params.max = 10
        [ playerList: Player.list( params ) ]
    }

    def show = {
        def player = Player.get( params.id )

        if(!player) {
            flash.message = "Player not found with id ${params.id}"
            redirect(action:list)
        }
        else { return [ player : player ] }
    }

    def delete = {
        def player = Player.get( params.id )
        if(player) {
            player.delete()
            flash.message = "Player ${params.id} deleted"
            redirect(action:list)
        }
        else {
            flash.message = "Player not found with id ${params.id}"
            redirect(action:list)
        }
    }
```

```
    def edit = {
        def player = Player.get( params.id )

        if(!player) {
            flash.message = "Player not found with id ${params.id}"
            redirect(action:list)
        }
        else {
            return [ player : player ]
        }
    }

    def update = {
    /* implementation */
    }

    def create = {
 /* implementation */
    }

    def save = {
 /* implementation */
    }

}
```

Le contrôleur permet d'ordonner l'affichage d'une vue. Ces vues sont définies par un layout. Il s'agit d'une page GSP particulière qui définit l'agencement des vues. Le contenu de chaque vue est inséré dynamiquement dans ce layout par Grails au niveau de la balise <g:layoutBody />. Cela permet de factoriser, le style et l'agencement des vues et de disposer ainsi d'une application à l'interface graphique uniforme. Le développeur peut définir autant de layout qu'il le souhaite.

```
<html>
    <head>
        <title><g:layoutTitle default="Grails" /></title>
        <link rel="stylesheet" href="${createLinkTo(dir:'css',file:'main.css')}" />
        <link rel="shortcut icon" href="${createLinkTo(dir:'images',file:'favicon
.ico')}" type="image/x-icon" />
        <g:layoutHead />
        <g:javascript library="application" />
    </head>
    <body>
        <div id="spinner" class="spinner" style="display:none;">
            <img src="${createLinkTo(dir:'images',file:'spinner.gif')}" alt="Spinner" />
        </div>
        <div class="logo"><img src="${createLinkTo(dir:'images',file:'grails
_logo.jpg')}" alt="Grails" /></div>
        <g:layoutBody />
    </body>
</html>
```

Une vue est une page GSP ne contenant pas d'en-tête, son code sera inséré dans le layout. Ici, il s'agit d'une partie de la vue List. On remarque l'utilisation de tag librairies spécifiques. Du code Groovy peut également être écrit directement dans la vue.

```
<table>
                <thead>
                    <tr>
                        <g:sortableColumn property="id" title="Id" />
                        <g:sortableColumn property="birthdate" title="Birthdate" />
                        <g:sortableColumn property="firstname" title="Firstname" />
                        <g:sortableColumn property="name" title="Name" />
                    </tr>
                </thead>
                <tbody>
                <g:each in="${playerList}" status="i" var="player">
                    <tr class="${(i % 2) == 0 ? 'odd' : 'even'}">
                        <td><g:link action="show" id="${player.id}">${pla
yer.id?.encodeAsHTML()}</g:link></td>
                        <td>${player.birthdate?.encodeAsHTML()}</td>
                        <td>${player.firstname?.encodeAsHTML()}</td>
                        <td>${player.name?.encodeAsHTML()}</td>
                    </tr>
                </g:each>
                </tbody>
            </table>
```

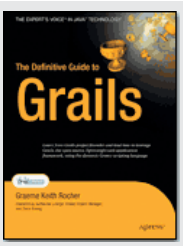
Conclusion

Grails est déjà bien mature et son évolution s'oriente vers une intégration avancée aux environnements Java EE et notamment via le support des portails, de l'API de mail ainsi que du module de gestion de contenu (Java Content Repository). Les outils du développeur seront améliorés par le support de Maven 2 ainsi que des tests continus. Cependant, la base du framework va elle aussi évoluer afin d'améliorer le support de la persistance via le standard JPA. D'autres évolutions comme le support de la librairie de tags JSP et de migration de base de données sont prévues. L'utilisation du framework va très probablement se généraliser dans les mois à venir. L'architecture à base de plug-in laisse donc imaginer des architectures bien plus riches que le MVC des premières versions de Grails.

■ Loïc Guillois

+ Tutoriel sur www.programmez.com/tutoriel.php

Grails et les tests



Grails supporte deux types de tests : les tests unitaires et les tests d'intégration. Cela permet d'apporter un niveau de contrôle et de qualité nécessaires aux applications d'entreprise. Il est même possible de mettre en œuvre la méthodologie de développement piloté par les tests. Il s'agit d'une approche préconisant d'écrire les tests unitaires avant les implémentations. La TDD permet de renforcer la qualité et la maintenabilité des applications et s'intègre dans de bonnes conditions aux approches agiles notamment par la méthodologie eXtrem Programming.

DEVELOPPEZ VOTRE SAVOIR-FAIRE

ABONNEMENT .NET

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 !

PROGRAMMEZ

- **Abonnement 1 an au magazine : 45 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*
- **1 an au magazine + 5 numéros HS .NET : 63 €** *Tarif France métropolitaine*
- **Abonnement Intégral : 1 an au magazine + archives sur Internet et PDF : 57 €** *Tarif France métropolitaine*
- **Abonnement PDF / 1 an : 30 €** *- Tarif unique*
  Inscription et paiement exclusivement en ligne
  www.programmez.com
- **Abonnement Etudiant : 1 an au magazine : 39 €** (au lieu de 65,45 € tarif au numéro) *Offre France métropolitaine*

PROGRAMMEZ HORS SERIE .NET *Seulement*

- **Abonnement 1 an aux Hors Série : 5 numéros : 20 €** (au lieu de 25 € tarif au numéro) *Tarif France métropolitaine*

Abonnez-vous :

Le magazine mensuel

11 numéros par an : **45 €***

soit **3** Numéros GRATUITS

**Tarifs France métropolitaine*

Abonnez-vous :

Les Hors-Série .Net

5 numéros par an : **20 €***

soit **1** Numéro GRATUIT

**Tarifs France métropolitaine*

+ Abonnement INTÉGRAL

NOUVEAU

ACCÈS ILLIMITÉ aux ARCHIVES du MAGAZINE pour 1€ par mois ! *Prix de lancement*

Cette option est réservée aux abonnés pour 1 an au magazine, quel que soit le type d'abonnement (Éco, Numérique, Etudiant). Le prix de leur abonnement normal est majoré de 12 € (prix de lancement, identique pour toutes zones géographiques). Pendant la durée de leur abonnement, ils ont ainsi accès, en supplément, à tous les anciens numéros et articles /dossiers parus.

OUI, *je m'abonne*

Vous pouvez aussi vous abonner en ligne et trouver tous les tarifs www.programmez.com

PROGRAMMEZ

- ☐ **Abonnement 1 an au magazine : 45 €** (au lieu de 65,45 € tarif au numéro) *Tarif France métropolitaine*
- ☐ **1 an au magazine + 5 numéros HS .NET : 63 €** Seulement (au lieu de 90,45 € tarif au numéro) *Tarif France métropolitaine*
- ☐ **Abonnement Intégral : 1 an au magazine + archives sur Internet et PDF : 57 €** *Tarif France métropolitaine*
- ☐ **Abonnement Etudiant : 1 an au magazine : 39 €** (au lieu de 65,45 € tarif au numéro) *Offre France métropolitaine*

PROGRAMMEZ HORS SERIE .NET NOUVELLE OFFRE

- ☐ **Abonnement 1 an aux Hors Série : 5 numéros : 20 €** *(au lieu de 25 € tarif au numéro) Tarif France métropolitaine*

☐ M. ☐ Mme ☐ Mlle Entreprise : ..... Fonction : .....

Nom : ..... Prénom : .....

Adresse : .....

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

Tél : ..... 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

Programmez! LE MAGAZINE DU DÉVELOPPEMENT

Offre limitée, valable jusqu'au 30 août 2008

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 110

Programmation Qt

2e partie

Nous continuons, dans ce numéro, le développement de l'application QuickDesktopNotes en créant la fenêtre de configuration ainsi que le parser XML/RSS.

L'analyseur XML/RSS

Cette classe doit pouvoir analyser le fichier RSS dans lequel l'application va sauvegarder la liste des notes. Ce fichier se conforme à la norme RSS 2.0. Si vous n'êtes pas familier de cette norme, vous pouvez vous rendre sur un des nombreux sites la décrivant ou sur Wikipedia qui contient suffisamment d'informations pour la compréhension de cet article. Petit point important : le fichier RSS est conforme à la norme mais notre " handler " n'extrait que les informations dont nous avons expressément besoin ! Voyons dans un premier temps le fichier d'en-tête du parser (nommé PXmlHandler) :

```
#ifndef PXMLHANDLER_H
#define PXMLHANDLER_H

#include <QXmlDefaultHandler>
#include <QList>
#include <QAction>

class QString;


class PXmlHandler : public QXmlDefaultHandler
{
	public:
		PXmlHandler( QList<QAction *> *newList);
		bool startElement( const QString &namespaceURI, const QString
&localName, const QString &qName, const QXmlAttributes &attributes );
		bool endElement( const QString &namespaceURI, const QString
&localName, const QString &qName );
		bool characters(const QString &str);
		bool fatalError(const QXmlParseException &exception);
	private:
		QList<QAction *> *actionList;
		QString currentText;
		QAction *currentAction;
		bool haveItem;
};

#endif
```

Tout d'abord, nous constatons que PXmlHandler hérite publiquement de QXmlDefaultHandler. Celui-ci implémente toutes les fonctionnalités de QXmlReader et QXmlSimpleReader. De plus, cette classe hérite à la fois que QXmlContentHandler et de QxmlErrorHandler, ce qui en fait un couteau suisse du XML ! L'intérêt majeur de QXmlDefaultHandler est qu'elle permet de ne se concentrer que sur l'analyse propre à notre application. Comme vous allez vous en rendre compte, cette classe prend en charge l'essentiel de l'analyse et nous n'écrirons que le code propre à l'utilisation des données dont nous avons besoin, contenues dans le document XML. D'ou sa dénomination de " handler " : elle ne s'occupe que de l'utilisation des données et non pas de l'analyse à proprement parler. Nous redéfinissons les 4 fonctions nécessaires à l'analyse d'un document XML via SAX2 : startElement(), endElement(), characters() et fatalError(). Ces 4 fonctions retournent un booléen qui, s'il est positionné à false, interrompt l'analyse du document. Le lecteur pourra se référer à l'assistant pour une introduction à SAX2 s'il n'est pas familier de cette méthode d'analyse de flux XML.

Enfin, nous déclarons 4 variables privées. Tout d'abord, *actionList. C'est une QList de QAction qui représente l'historique des notes. Celui-ci étant sans limite fixe, une liste s'impose. La QString currentText est un accumulateur de texte que nous utilisons pour concaténer le contenu d'une balise. Le pointeur currentAction et le booléen haveItem sont des facilités que nous nous offrons et dont nous allons reparler. Avant d'entamer l'analyse du code, voici un exemple d'une note sauvegardée au format RSS :

```
<item>
	<title>the current note...</title>
	<link>file://[...]/desktop_notes.rss</link>
	<description>is lonely !</description>
	<pubDate>sam. mai 3 12:41:04 2008</pubDate>
	<guid></guid>
</item>
```

L'élément <title> contient bien entendu le titre de la note et l'élément <description> en détient le contenu. Voyons le code de cette classe. Dans un premier temps il est nécessaire d'inclure le fichier d'en-tête ainsi que la définition de QMessageBox, qui va nous permettre d'afficher une boîte de dialogue en cas d'erreur fatale.

```
#include "pxmlhandler.h"
#include <QMessageBox>
```

Après quoi, nous définissons le constructeur du parser. Celui-ci prend en argument un pointeur vers une QList contenant elle même des pointeurs vers des QAction. Cette liste sera ensuite utilisée par la classe PTray pour créer l'historique des notes. Ceci permet de fournir le mécanisme de notes multiples (qui sont toutes éditables).

```
PXmlHandler::PXmlHandler( QList<QAction *> *newList){
	actionList = newList;
	currentText = "";
	haveItem = false;
}
```

Le constructeur stocke le pointeur sur la liste dans une variable membre (actionList) et initialise l'accumulateur de texte. Il s'occupe aussi d'ini-

tialiser haveItem à false. Ce booléen nous permet de savoir, pendant l'analyse, que nous sommes bien à l'intérieur d'une balise <item>. En parlant d'analyse, la première fonction d'analyse SAX est la méthode startElement(). Celle-ci prend 4 arguments : l'URI de l'espace de nom si celui-ci a été déclaré, le nom local dans le cas où un espace de nom est déclaré, le nom complet (avec le préfixe) et enfin la liste des attributs (sous forme de QXmlAttributes). Cette méthode est appelée automatiquement par le lecteur (qui sera dans le cas de la classe PTray un QXmlSimpleReader). Pour plus d'informations sur la façon d'analyser des fichiers XML avec Qt, on se reportera à la documentation du module QXml.

```
bool PXmlHandler::startElement( const QString &namespaceURI, constQString
&localName, const QString &qName, const QXmlAttributes &attributes ){
        if( qName == "item" ){
                currentAction = new QAction( QString("none"),0 );
                haveItem=true;
        }
        else
            if(qName == "title" || qName == "description")
                currentText = "";
        return true;
}
```

Lorsque startElement() est appelée, nous savons que l'analyseur a rencontré une balise ouvrante et nous ne nous occupons donc que des cas qui nous intéressent, à savoir :

- créer une nouvelle entrée dans l'historique (i.e créer une nouvelle QAction) à chaque fois que nous voyons apparaître une nouvelle note (i.e un nouvel élément <item>) ;
- modifier le contexte en positionnant haveItem à true afin que les autres méthodes sachent que nous analysons le contenu d'une note ;
- si la balise ouvrante est un élément <title> ou <description> réinitialiser l'accumulateur de texte.

Toute balise ouvrante doit avoir sa balise fermante pour que le document soit valide. Lorsqu'un tel élément est rencontré par l'analyseur, il appelle la méthode endElement(). Celle-ci prend les mêmes arguments que startElement() en excluant bien entendu les attributs.

```
bool PXmlHandler::endElement( const QString &namespaceURI, const Q
String &localName, const QString &qName )
{
    if(haveItem){
        if(qName == "title")
            currentAction->setText(currentText);
        else
            if(qName == "description")
                currentAction->setToolTip(currentText);
            else
                if(qName == "item"){
                    actionList->append(currentAction);
                    haveItem=false;
                }
    }
    return true;
}
```

Le code est, là aussi, assez simple. Nous commençons par tester le contexte, si nous sommes présentement à l'intérieur d'un élément <item>, nous vérifions le nom de l'élément en cours de fermeture. Si c'est une balise <title> nous affectons le contenu de l'accumulateur de texte au texte visible de la QAction (que nous considérons donc être son titre). Si c'est une balise <description>, nous affectons le contenu de l'accumulateur au " tooltip " de l'action (que nous considérons être le contenu d'une note). Enfin, si l'élément est un <item>, nous ajoutons l'action courante à la fin de la liste et nous modifions le contexte pour signaler que nous ne sommes plus à l'intérieur d'un élément <item>.
Vous constatez que l'accumulateur de texte a un rôle vital. C'est la fonction characters() qui s'occupe de le remplir. Cette dernière prend pour argument le texte rencontré par l'analyseur (sous forme de QString). Nous nous contentons de concaténer cette chaîne de caractères à l'accumulateur :

```
bool PXmlHandler::characters(const QString &str){
        currentText += str ;
        return true;
}
```

De fait, la réinitialisation de l'accumulateur étant effectuée à chaque fois qu'une nouvelle balise (intéressante) ouvrante est rencontrée, nous n'avons rien besoin de tester.
La dernière fonction à implémenter est, bien entendu, celle qui gère les erreurs fatales. Originalement nommée fatalError(), cette dernière prend un QXmlParseException en paramètre.

```
bool PXmlHandler::fatalError(const QXmlParseException &exception){
        QMessageBox::warning(0,QObject::tr("QuickDesktopNotes XML/
RSS parser"),
                QObject::tr("parse error at line %1, column %2:\n%3")
                .arg( exception.lineNumber() )
                .arg( exception.columnNumber() )
                .arg( exception.message() )

        );
        return false;
}
```

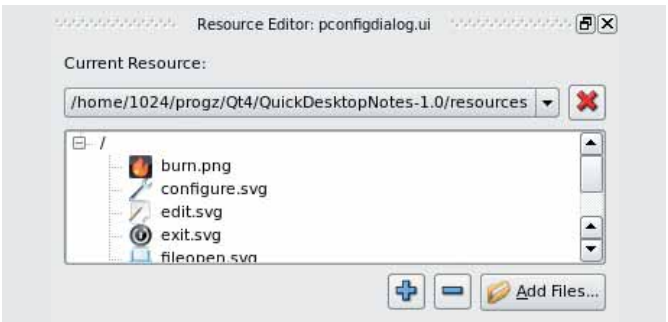
Si une erreur est détectée nous affichons une fenêtre d'avertissement qui donne plus de détails. Remarquez que nous appelons la fonction tr() pour l'affichage de nos chaînes de caractères. Celle ci retourne la chaîne passée en argument traduite (si un traducteur est installé) sous forme de QString. Ceci permet aussi l'introduction d'une fonctionnalité appréciable de QString : les arguments de chaînes. En plaçant, des mots spéciaux %<chiffre>, il est possible de placer des variables dans une QString. Il suffit d'appeler la fonction arg() de cet objet pour que la variable soit remplacée par la valeur donnée en paramètre de arg(). Cette dernière retournant à son tour une QString, il est possible de chaîner les appels.
Nous avons fini avec cette classe, il ne reste plus qu'à modifier le fichier projet. Tout d'abord, il faut ajouter notre classe à la ligne HEADERS et SOURCES mais il faut aussi ajouter la ligne suivante :

```
QT += xml
```

Afin de préciser au compilateur de lier le module QtXml au binaire. Regénérez votre Makefile, puis testez la compilation de cette classe. Nous l'utiliserons lors du prochain article dans la classe PTray.

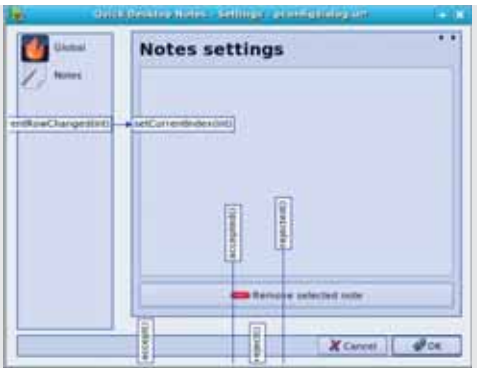
La boîte de dialogue de configuration

Pour créer la boîte de configuration il va falloir ressortir notre bon designer. Suivant le même procédé que pour la classe PNoteEditor, nous allons d'abord dessiner notre interface dans le designer avant d'implémenter les fonctionnalités qui nous intéressent.
Fonctionnellement parlant, nous voulons pouvoir gérer les notes sauvegardées (et supprimer celles dont nous n'avons plus d'utilité). Nous désirons aussi pouvoir spécifier l'emplacement du fichier RSS ainsi que l'icône à utiliser dans le dock.
Nous nommerons cette fenêtre de configuration PConfigDialog et c'est un " dialog with buttons bottom " que vous devez créer. Dans un premier temps, créez un fichier de ressource dans lequel vous référencerez toutes les images. Il sera compilé avec l'application et permet d'embarquer le nécessaire (images, texte, etc.).



Ajoutez maintenant un QListWidget et un QStackedWidget. Dans le QListWidget ajoutez 2 items : " Global " et " Notes ", affectez leur les icônes qui vous plaisent puis sauvegardez ! Le QStackedWidget permet de mettre en place une pile de widgets visibles à tour de rôle. Sur la première page ajoutez : 3 Qlabel, 1 QLineEdit (nommez le saveAsPath), 1 QComboBox (dont le nom sera iconSelector), et 2 QToolButton (que l'on baptisera saveAsButton et openIconButton) en vrac.
Positionnez ensuite un QLabel, saveAsPath et saveAsButton sur la même ligne, sélectionnez les tous les 3 et cliquez sur " Lay Out Horizontally ". Faites de même avec un autre QLabel, iconSelector et openIconButton. Sélectionnez ensuite les 2 groupes que vous venez de créer et cliquez sur " Lay Out Vertically ". Ajoutez ensuite un " vertical spacer " et positionnez le tout dans un layout de type grille. Voici ce que vous devriez obtenir :

Cliquez sur la petite flèche en haut du QStackedWidget pour changer de page. Cette seconde page est bien plus simple : ajouter un Qlabel, un QListWidget (notesList) et un QpushButton (removeNoteButton), puis disposez cette page dans une grille.
Il ne nous reste plus qu'à connecter le signal currentRowChanged(int) du QListWidget latéral au slot setCurrentIndex(int) du QStackedWidget. Ceci se fait graphiquement, cliquez sur le bouton " Edit Signals/Slots ", sélectionnez la QlistWidget et glissez vers le QStackedWidget, choisissez les bon signaux/slots dans la boîte qui apparaît, puis validez votre choix. C'est tout ! Désormais, même dans la " preview " votre fenêtre réagit correctement.



Remarquez au passage que les boutons " Ok " et " Cancel " sont déjà connectés aux slots ad-hoc. Sauvegardez le tout sous le nom pconfigdialog.ui et faites chauffer votre éditeur favori. Le code de cette classe étant un peu volumineux nous ne l'analyserons pas entièrement, nous nous concentrerons sur l'essentiel. Le code source de l'application étant fourni par ailleurs. Tout d'abord, nous devons inclure quelques en-têtes :

```
#include "pconfigdialog.h"
#include <QDir>
#include <QFileDialog>
#include <QSettings>
```

Le constructeur ne présente pas de difficulté, il est à noter qu'il appelle la méthode readSettings(). Détaillons plutôt le slot setNoteList(). Il sera appelé dans la classe PTray, à la création de la fenêtre de configuration.

```
void PConfigDialog::setNoteList(QActionGroup *group){
    actionsGroup = group;
    QAction *ita;
    foreach (ita, actionsGroup->actions()){
        noteList->addItem(ita->text());
        QListWidgetItem *item = noteList->item( noteList->count()-1 );
        item->setToolTip(ita->toolTip());
    }
}
```

Ce slot prend en argument un QActionGroup qui représente la liste des notes chargées. Il itère ensuite sur la liste des actions (grâce à la macro foreach(<variable>, <conteneur>) de Qt) et pour chaque note, ajoute un item dans la liste des notes. Enfin, il récupère le dernier item ajouté pour affecter son tooltip avec le contenu de la note.
Lorsque le bouton de suppression d'une note est cliqué, le slot on_remo-

veNoteButton_clicked() est appelé. Celui-ci bénéficie de la connexion par nom de fonction. Dans un premier temps, le slot récupère l'item qui est sélectionné, si il y a effectivement un item sélectionné, il itère sur la liste des actions. Dans le cas ou il trouve une action identique (même titre, même contenu), il la supprime de la liste et émet le signal (déclaré dans le header) noteRemoved(QAction*) puis sort de la boucle. Le signal est émis pour que la classe PTray tienne à jour sa propre liste d'actions. C'est généralement une très mauvaise idée de modifier directement la structure de données d'une fenêtre depuis une autre. L'émission d'un signal est bien plus élégante et moins génératrice de bugs... Ceci dit, il aurait été possible de le faire ici car nous détenons le pointeur de la liste d'actions de PTray dans la variable membre actionsGroup.

```
void PConfigDialog::on_removeNoteButton_clicked(){
    QListWidgetItem *item = noteList->currentItem();
    if(item){
        QAction *ita;
        foreach (ita, actionsGroup->actions()){
            if( ita->text() == item->text() && ita->toolTip() == item->toolTip() ){
                emit(noteRemoved(ita));
                noteList->takeItem( noteList->row(item) );
                break;
            }
        }
    }
}
```

Notez que la macro foreach() se casse comme une boucle classique et que la comparaison de QString potentiellement grande ne pose aucun problème. Une fois ceci mis en place, il faut écrire le code des 2 boutons de sélection de fichier. Les 2 slots on_saveAsButton_clicked() et on_openIconButton_clicked() étant très similaires, nous ne détaillerons que le second. Dans un premier temps, nous demandons à l'utilisateur de sélectionner le fichier d'icône qu'il veut utiliser (grâce à QfileDialog::getOpenFileName()). Celle-ci prend en argument un QWidget de référence, un titre, un chemin dans lequel se positionner, ainsi qu'un filtre de fichiers. Si l'utilisateur a sélectionné un fichier, nous l'ajoutons à la première position du menu déroulant, puis nous sélectionnons l'icône fraîchement ajoutée.

```
void PConfigDialog::on_openIconButton_clicked(){
      QString iconPath = QFileDialog::getOpenFileName(this,tr("Quick
 Desktop Notes"), QDir::currentPath() , tr("Images (*.png *.xpm *.jpg *.svg)") );
    if(!iconPath.isEmpty()){
        iconSelector->insertItem(0, QIcon(iconPath),iconPath );
        iconSelector->setCurrentIndex(0);
    }
}
```

La redéfinition du slot accept() ne posant aucun problème, nous la laissons de côté pour nous concentrer sur les 2 fonctions privées writeSettings() et readSettings(). Ces dernières reposent sur la classe QSettings qui permet de sauvegarder des configurations indépendamment de la plate-forme (via le registre sous Windows et des fichiers ini sous Unix). Le constructeur basique de cet objet prend en argument un nom d'organisation ainsi qu'un nom de programme. Son utilisation est ensuite totalement triviale : on affecte des valeurs en utilisant la méthode setValue(<nom de valeur>, <valeur>) et on retire les valeurs en utilisant value(<nom de valeur>, <valeur par défaut>). La valeur par défaut est facultative. Le constructeur appelle readSettings() et accept() appelle writeSettings().

```
void PConfigDialog::writeSettings()
{
     QSettings settings("Programmez", "QuickDesktopNotes");
     settings.setValue("notesFileName",saveAsPath->text());
    settings.setValue("iconFileName",iconSelector->itemText((iconSelector->
currentIndex())) );
}

void PConfigDialog::readSettings()
{
    QSettings settings("Programmez", "QuickDesktopNotes");
     saveAsPath->setText( settings.value("notesFileName","desktop_notes.
rss").toString() );
     QString iconPath = settings.value("iconFileName",":/images/burn.
png").toString();
    if(!iconPath.startsWith(":/")){
        iconSelector->insertItem(0, QIcon(iconPath),iconPath );
        iconSelector->setCurrentIndex(0);
    }
    else{
        iconSelector->setCurrentIndex( iconSelector->findText(iconPath) );
    }
}
```

Vous constatez que si le nom du fichier icône ne provient pas du fichier de ressource, nous l'ajoutons au menu déroulant.
Il ne reste plus qu'à ajouter cette nouvelle classe au fichier projet afin de la compiler.

Conclusion

Bien qu'un peu dense cet article nous a permis d'introduire un grand nombre de notions intéressantes. Vous pouvez modifier le main.cpp que nous avons créé la dernière fois pour tester votre nouvelle fenêtre de configuration. A titre d'exercice vous pouvez aussi tenter d'implémenter par vous-même les méthodes manquantes de la classe PConfigDialog ! Suite et fin au prochain numéro !

Arnaud Dupuis
Consultant - Uperto (Open Source Business Unit) - Devoteam group

À la découverte de l'architecture SCA

Cet article propose de découvrir SCA (Service Component Architecture) en utilisant le runtime Apache Tuscany et l'éditeur graphique SCA du projet Eclipse STP développé par le français Obeo (http://www.obeo.fr).

SCA marie l'approche par composants et l'approche par services pour construire une architecture SOA. Ces architectures sont actuellement popularisées par des méthodologies, des grands principes d'isolation entre applications, et quelques frameworks, dont le célèbre Spring, mais aucune norme technique ne s'était jusqu'alors imposée pour la partie implémentation technique. Afin de ne pas réinventer la roue et de s'insérer dans les middlewares existants (EJB, Corba, SOAP, ...), SCA n'impose pas de technologies pour l'implémentation des composants, la définition des interfaces des services et le protocole d'accès aux services. Une application SCA est décrite par des fichiers XML décrivant un assemblage de composants au sein de composites où chaque composant définit des services fournis et des services requis. Des implémentations sont ensuite associées à chaque composant et des interfaces aux services et références. Il suffit enfin de relier les services requis d'un composant aux services rendus d'un autre, pour que les deux applications communiquent, en automatisant les éventuelles conversions de protocole et de technologie middleware. Différentes implémentations Open Source des spécifications SCA existent, Tuscany, FraSCAti (issu du projet français ANR SCOrWare), Fabric 3 et des outils pour SCA sont disponibles au sein du top-level project Eclipse SOA Tools Platform.

Installation de Tuscany et de Eclipse STP/SCA

Télécharger Tuscany 1.1 (http://incubator.apache.org/tuscany/sca-java-releases.html) et décompresser l'archive. Installez ensuite les outils SCA du projet Eclipse STP disponibles sur la page web STP (http://www.eclipse.org/stp/sca/index.php). Une fois Eclipse démarré, créez une librairie des jar Tuscany. Dans la barre de menu, sélectionnez Window > Preference, puis dans la boîte de dialogue, sélectionnez Java > Build Path > User Librairies. Cliquez sur le bouton Next. Entrez TUSCANY comme nom de librairie. Cliquez sur Add Jars et ajoutez tous les jar Tuscany qui se trouvent dans le répertoire lib de Tuscany.

Votre première application SCA

La figure 1 montre l'application SCA que nous allons construire à l'aide de l'éditeur graphique GMF. Le composite Restaurant est formé de 3 composants :

- RestaurantServiceComponent donne la liste des menus et calcule l'addition pour un menu,
- VatServiceComponent calcule la TVA,
- TipServiceComponent calcule le pourboire.

Créez un nouveau projet Java nommé Restaurant et ajoutez la librairie TUSCANY dans le build path.

Définition de l'assemblage SCA

Pour créer de manière graphique l'assemblage SCA :

- Clic droit sur le répertoire src de votre projet > New > Other...
- Sélectionnez SCA Composite Diagram > Next,
- Entrez Restaurant comme nom de fichier > Finish.

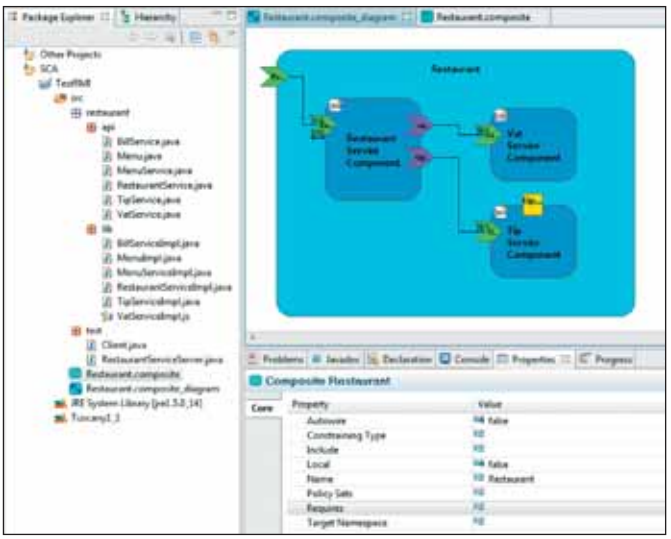


Figure 1: Première application SCA

Deux fichiers sont créés : *Restaurant.composite* le fichier XML décrivant l'assemblage SCA et *Restaurant.composite_diagram,* contenant les données graphiques. Le designer SCA est ouvert automatiquement. Ouvrez la fenêtre propriétés (clic droit sur le diagramme > sélectionnez Show properties view). Utilisez la palette se trouvant sur la droite de l'éditeur pour créer 3 composants (outil Component) nommés : RestaurantServiceComponent, VatServiceComponent et TipServiceComponent. Définissez ensuite les services offerts par ces composants (outil Service dans la palette) nommés : RestaurantService (composant RestaurantServiceComponent), VatService (composant VatServiceComponent) et TipService (composant TipServiceComponent). De même, ajoutez les références nommées vatService et tipService (composant RestaurantServiceComponent). Connectez (outil Wire) ensuite la référence vatService avec le service VatService et tipService avec le service TipService. La dernière étape consiste à rendre visible depuis l'extérieur du composite le service offert par le composant RestaurantServiceComponent. Pour cela, faites un clic droit sur le service, puis sélectionnez dans le menu l'action Promote.

Sauvegardez votre assemblage qui doit ressembler à la figure 1.

Définition des interfaces et des implémentations

Dans le répertoire source de votre projet, créez le package restaurant et les sous-packages api et lib. Dans le package api définissez les interfaces suivantes :

```
public interface RestaurantService{
  Menu[] getMenus();
  double getBill(Menu menu);
}
```

```
public interface VatService{
  double getPriceWithVat(double price);
}

public interface TipService{
  double getPriceWithTip(double price);
}
```

Définissez également l'objet de transfert suivant :

```
public interface Menu extends Serializable{
  String printMenu();
}
```

Définissez ensuite les implémentations dans le package restaurant.lib :

```
@Service(RestaurantService.class)
public class RestaurantServiceImpl implements RestaurantService{
  private Menu[] menus;
  private double[] prices;
  private VatService vs;
  private TipService ts;

  @Init
  public void init(){
    menus=new Menu[]{
      new MenuImpl(0,"Grilled hamburger with French fries"),
      new MenuImpl(1,"Roasted chicken with vegetables"),
      new MenuImpl(2,"Duck breast in an orange sauce"),
      new MenuImpl(3,"Duck foie gras & mango chutney")};
    prices=new double[] {10,15,35,50};
  }

  @Reference
  public void setTipService(TipService ts){
    this.ts=ts;
  }
  @Reference
  public void setVatService(VatService vs){
    this.vs=vs;
  }

  public double getBill(Menu menu){
    double menuPrice=prices[((MenuImpl) menu).getId()];
    double pricewithTaxRate=vs.getPriceWithVat(menuPrice);
    double priceWithTipRate=ts.getPriceWithTip(pricewithTaxRate);
    return priceWithTipRate;
  }
  public Menu[] getMenus(){return menus;}
}

public class MenuImpl implements Menu{
  private int id;
  private String details;
  MenuImpl(int idC, String detailsC){
    id = idC; details = detailsC;
  }
```

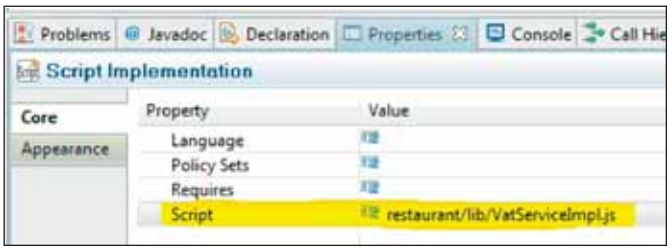


Fig.2

Fig.3

```
  public String printMenu(){return details;}
  int getId(){return id;}
}

@Service(TipService.class)
public class TipServiceImpl implements TipService{
  @Property
  public double tipRate;
  public TipServiceImpl(){tipRate=10;}
  public double getPriceWithTip(double price){return price*tipRate/100+price;}
}
```

Dans le répertoire src/restaurant/lib créez un fichier nommé VatServiceImpl.js avec le code JavaScript suivant :

```
var vatRate=19.6
function getPriceWithVat(price) {
  return price*vatRate/100+price;
}
```

Vous pouvez maintenant compléter l'assemblage SCA en associant les interfaces aux services et les implémentations aux composants. Pour cela, glissez/déposez les interfaces Java (classes Java) que vous avez écrites sur les services (composants) correspondants. Le code javascript ne peut être glissé/déposé, utilisez la palette pour créer un nouvel élément Script sur le composant VatServiceComponent. Dans la vue propriétés, mettez la valeur restaurant/lib/VatServiceImpl.js pour l'attribut Script (figure 2).

Afin de rendre accessible l'application par RMI, ajoutez sur le service du composite un binding RMI (outil RMIBinding). Dans la vue propriété du binding mettez la valeur localhost pour l'attribut host, 8099 pour le port et TestRMI pour le nom du service.

Validez l'assemblage en sélectionnant dans le menu Diagram > Validate. Les erreurs apparaissent dans la vue Problems.

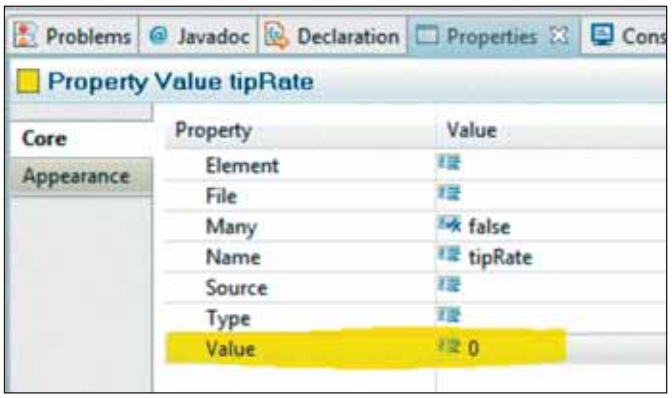
Test

Pour démarrer l'application SCA créez la classe suivante :

```
import org.apache.tuscany.sca.host.embedded.SCADomain;
public class RestaurantServiceServer{
  public static void main(String[] args) throws Exception{
    System.out.println("Starting of the SCA Restaurant Application");
    SCADomain scaDomain=SCADomain.newInstance("Restaurant.composite");
    System.out.println("Press Enter to Exit");
    System.in.read();
    scaDomain.close();
    System.out.println("Exited");
    System.exit(0);
  }
}
```

Créez ensuite le code du client :

```
public class Client{
  public static void main(String[] args) throws Exception{
    RestaurantService restaurantService=(RestaurantService) Naming
        .lookup("//localhost:8099/TestRMI");
    Menu[] menus=restaurantService.getMenus();
    System.out.println("--- Menu ---");
    int i = 0;
    for (Menu m : menus){
      System.out.println(++i + " - " + m.printMenu());
    }
    System.out.println("----------");
      BufferedReader br = new BufferedReader(new InputStreamReader
(System.in));
    int choice=-1;
    while (choice==-1){
      System.out.print("? ");
      String c = br.readLine();
      try{
        choice=new Integer(c).intValue();
        if (choice<1||choice>menus.length){choice = -1;}
      } catch (NumberFormatException e){}
    }
    System.out.println();
```




```
    Menu menu=menus[choice-1];
    System.out.println("Your choice: " + menu.printMenu());
    System.out.println("Price: " + restaurantService.getBill(menu));
  }
}
```

Démarrez l'application SCA puis le Client afin de tester l'application (figure 3).

Redéfinition d'une propriété

Vous pouvez facilement redéfinir la propriété du composant TipService-Component afin de fixer un taux différent. Ajoutez une propriété au composant (outil Property) ayant pour nom tipRate. Assignez lui la valeur 0 (figure 4). Relancez le test. Le prix pour le menu 4 est maintenant 59,8. Pour aller plus loin, vous pouvez changer avec le modeleur SCA le protocole de communication et ajouter une interface WebService de description d'une interface.

Conclusion

Cet article vous a donné un bref aperçu de SCA et des outils open source disponibles pour le développement SCA. Pour toute information complémentaire ou questions relatives aux outils SCA n'hésitez pas à poser vos questions sur le newsgroup STP (news://news.eclipse.org/eclipse.stp).

Stéphane Drapeau
Leader du projet Eclipse STP SCA
stephane.drapeau@obeo.fr

+ Tutoriel sur
www.programmez.com/tutoriel.php