Programmez !
www.programmez.com
Mensuel - Janvier 2008 - N°104
LE MAGAZINE DU DÉVELOPPEMENT

Build

Construire vos applications !
Utiliser des usines à build : Ant, Autotools, Maven, MS Build. Créer des Nightly Build

© E. James Small

RIA dans votre mobile avec Java

Développer des applications pour Microsoft Home Server

CYCLE DE VIE LOGICIELS

Améliorer la gestion des projets

Créez le cycle de vie de votre application avec Maven et Team System

Processeur

Bien débuter en programmation multithread

Développement web
Découvrez le surprenant langage HOP

C++
Ultimate++ : l'ultime IDE C++

VB
Migrer vos codes VB avec RealBasic

Langage
Haskell : comprendre la programmation fonctionnelle

Et aussi
- **OpenLaszlo**
- **Les plug-in JGear**
- **PHP Forum**
- **Remoting .Net**

Printed in France - Imprimé en France - BELGIQUE 6,45 € - SUISSE 12 FS - LUXEMBOURG 6,45 € - Canada 8,95 $ CAN DOM Surf 6,90 € - TOM 940 XPF - MAROC 50 DH
M 04319 - 104 - F: 5,95 €

WINDEV | WEBDEV | WINDEV Mobile

DÉVELOPPEZ 10 FOIS PLUS VITE

PLATEFORME PROFESSIONNELLE DE DÉVELOPPEMENT (AGL)

Et vous, connaissez-vous WINDEV 12 ?
Ici présentation de WINDEV à Paris, une des 12 villes du Tour de France WINDEV.

UN CODE UNIQUE :
Windows, .Net, Java, PHP, J2EE, XML, Internet, Ajax, Pocket PC, SmartPhone, Client riche ...

Des applications superbes sans compétence graphique grâce aux gabarits fournis. Ergonomie assurée par le correcteur d'IHM intégré.

Parmi les 500 nouveautés:

- Accès natif à **SAP**
- **Fond de page** PDF
- Débogage à **distance**
- **Compilation** «JITc»
- Gestion des exigences
- Fonctions d'administration Réseau **SNMP**
- Réplication automatique
- Sauvegarde à chaud
- **Héritage** de modèle
- **Nouveau** RAD
- 100 Nouvelles fonctions **Java**
- 39 Nouvelles fonctions **PHP**
- 50 Nouvelles fonctions **Linux**
- Débogueur PHP
- **Web 2.0**/Ajax

500 NOUVEAUTÉS

112 pages de témoignages, sur simple demande

Demandez le dossier technique gratuit (en couleurs, en français), accompagné de 112 pages de témoignages et d'un DVD. **Version Express Gratuite.**

Tél Province **04.67.032.032** Tél Paris **01.48.01.48.88** **info@pcsoft.fr**

www.pcsoft.fr

Fournisseur Officiel de la Préparation Olympique

XII

SOMMAIRE N°104

Donnez votre avis sur ce numéro
www.programmez.com/magazine_satisfaction.php

CD-Rom 104 PROGRAMMEZ !

.NET Framework 3.5
Installer et développer vos applications avec le tout dernier framework .Net ! Découvrez C# 3, VB 9 et Linq !

Microsoft VOLTA 1.0
Concurrent de Google GWT, le projet Volta permet de créer rapidement des applications composites multi-tiers. Nécessite l'installation de Visual Studio 2008

JRuby 1.0.2
Implémentation de Ruby pour les environnements Java

Ruby on Rails 2.0.1
Le tout nouveau framework de développement web open source enfin disponible !

PHP 5.3
La toute dernière version. Inclut de nombreuses nouveautés prévues dans la future v6.0 !

WampServer 2
C'est la nouvelle version de WAMP5, plateforme de développement et serveur web sous Windows. Inclut Apache 2.2.6, MySQL 5.0.45 et PHP 5.2.5

OpenLaszlo 4.0.7
Plateforme de développement pour des applications web. Permet de créer des applications riches (Rich Internet Application). Version Windows et Linux.

Morfik WebOS AppsBuilder
Outil RAD destine à concevoir des applications Web AJAX. Limité à 30 jours

NetBeans 6.0 All
La toute dernière version du puissant IDE Java. Inclut les packages Java SE, Web, Java EE, Mobility, UML, SOA et Ruby ! Version Windows

.NetBeans 6.0 Web & Java EE
IDE Open Source pour Java. Comprend les packs Java SE et Web & Java EE. Version Linux

Perforce Server 2007.3
Dernière version de l'environnement de gestion de projet et de code source de l'éditeur Perforce. Inclut le client graphique Perforce 2007.2.

PSPad
Editeur HTML, CSS, PHP, ASP, Perl, pour le développement de sites Web en mode texte. Inclut un client FTP.

Magic Draw voir page 18

Avec le multithread vos applications pourront exploiter les nouveaux processeurs Intel® Core™2 Duo et Intel® Core™2 Duo Quad (2 et 4 processeurs en 1)

Nouvelles éditions des compilateurs C++ et Fortran 10.1:

En plus des autres Linux, support de Debian et Ubuntu
Support de OpenMP* Microsoft et de OpenMP* GNU
Tuning des processeurs: Penryn avec le switch -xS, Bonnel (enfouit) avec le switch -xL

Formations programmation parallèle.
www.microsigma.fr/EFlash/071203FormationIntel.html

Les compilateurs et bibliothèques sont faits les uns pour les autres:

Les Compilateurs Intel® C++ et Fortran parallélisent automatiquement votre code et l'optimisent en performances pour profiter au mieu des architectures multi-core.

L'Intel® Math Kernel Library met à votre disposition des fonctions mathématiques multi-threadées qui surpassent les performances de codes compilés individuellement ainsi que celles d'autres bibliothèques .

Les Intel® Integrated Performance Primitives (C++ seulement) incorporent des fonctions hautement optimisées qui accélèrent le développement du traitement de média, de cryptographie et du signal.

Les Intel® Threading Building Blocks (C++ seulement) regroupent des routines vérifiées et affinées pour simplifier le développement d'aplications multi-thread robustes et capables de monter en charge.

Formation Programmation Parallèle
Micro Sigma organise deux cours pour vous permettre d'avancer sur les techniques de programmation parallèle et l'utilisation des outils Intel correspondants

intel
Software
Elite Reseller

microsigma

Votre VAR, partenaire à valeur ajoutée officiel

N°Azur 0 810 120 240

Tel 01 30 82 04 54 Fax 01 39 69 93 31
www.microsigma.fr/intel
intel@microsigma.fr

Votre spécialiste Fortran, C++, XML et outils de développement

Au service de ses revendeurs et de ses clients depuis 1984

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, F. Dewasmes, L. Guillois, F. Bordage, J. Chable, J.M. Maman, C. Breysse, J. Lefevre, S. Pol, M. Oubechou, E. Leblond, A. Goncalves, F. Brachiela, F. Schafer, M. Serrano, C. Queinnec, O.Thery, J. Chable, H. Marquis. **Dessin :** Michel Piedoue
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 et 1 encart jeté Solutions Linux pour les abonnés
Illustration de couverture : *Robot conçu par Devon Brooks. Modèle construit par E. James Small. Photo : E. James Small www.smallartworks.ca*

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

2007..., an 0 du nouveau web...

Voilà, une nouvelle année informatique expédiée ! Comme le veut la tradition, il faudrait se plier au bêtisier de l'année ou au " meilleur de " ou encore à la sempiternelle rétrospective… Pour clore 2007, replongeons-nous dans les 12 derniers mois et 2 ou 3 tendances annuelles…
Cela n'aura été une surprise pour personne, le " Web 2.0 " aura été sur tous les claviers et les media n'ont pas arrêté d'en parler et de décliner le réseau social à toutes sauces avec du Dailymotion, des blogs, des facebook et autres Google… À peine si le grand lancement de Windows Vista en février dernier est encore dans les mémoires… Mais dans notre génial Web 2, deux problèmes demeurent : la sécurité et le référencement. Et il faut bien avouer que sur ces deux questions, depuis 18 mois, nous pédalons dans la semoule.

RDA : les nouvelles frontières

Mais finalement, la véritable nouveauté fut l'émergence de la première génération RIA, qui continuera à émerger en 2008… Personnellement, et techniquement parlant, RIA me semble la véritable nouveauté web que l'on attendait. Web 2.0 ne constitue finalement qu'un ensemble de briques assez hétérogènes. L'absence d'un Ajax " standard " nuit aux développeurs et JavaScript demeure leur bête noire. RIA propose une plate-forme mieux définie et architecturée. Et lorsque l'on met bout à bout la communication unifiée, RIA et le futur RDA, on arrive à une plate-forme unifiée et globale réellement excitante pour le développeur qui disposera enfin d'un environnement de conception, de programmation digne de ce nom, de langages dynamiques et statiques et surtout d'une vision d'architecture et d'utilisation lisible. Alors oui, 2008 sera l'an 0 du véritable nouveau web (en attendant le " web 3.0 "), avec la disponibilité de technologies comme AIR, Flex 3, Silverlight 2 ou encore de Mozilla 2, sans oublier les nouveautés Google telles que Google Gears. Et 2007 marque aussi les débuts d'une " fusion " collaborative entre développeur et designer…
RIA souffre encore d' immaturité et surtout du manque d'intégration avec les terminaux mobiles, limitation rédhibitoire. Mais gageons que cela se résoudra dans les 12 prochains mois. Dans le même temps, la RDA (Rich Desktop Application) pointe aussi le bout de son nez. Là aussi, nous n'en sommes qu'à l'aube d'une évolution.
Et si la guerre annoncée Adobe-Microsoft aura bien lieu, un troisième larron va troubler le jeu : Mozilla. Si certains ont souvent mal interprété le projet Prism, cela montre au moins que Mozilla ne veut plus être en marge du mouvement RIA / RDA / Mobilité. Et la nouvelle guerre des navigateurs qui se profile dans les prochains mois, risque de nous surprendre sur bien des points, notamment sur la mobilité, qui devient plus que jamais la cible !

La tornade iPhone

La mobilité 2007 restera marquée par l'iPhone, vous savez ce téléphone d'Apple, mais si, la pomme croquée qui croque tout le monde… Techniquement, fonctionnellement, rien d'exceptionnel, certes. Mais ce qu'on n'a pas toujours compris, c'est que l'iPhone propose une nouvelle vision du téléphone, de son utilisation, de ses applications. Avec une interface, une ergonomie que je qualifie d'extraordinaires, d'où mon éhonté slogan : " iPhone est ton ami ". N'oublions pas non plus le buzz, la rumeur du Gphone. Google nous sort finalement, et à juste raison, un SDK, une plate-forme mobile, Android. Rien que pour cela, 2008 marquera une rupture entre la mobilité d'aujourd'hui, et celle du futur avec l'arrivée, enfin réelle, de l'Open Source, du Libre !

François Tonic - ftonic@programmez.com

PROCHAIN NUMERO

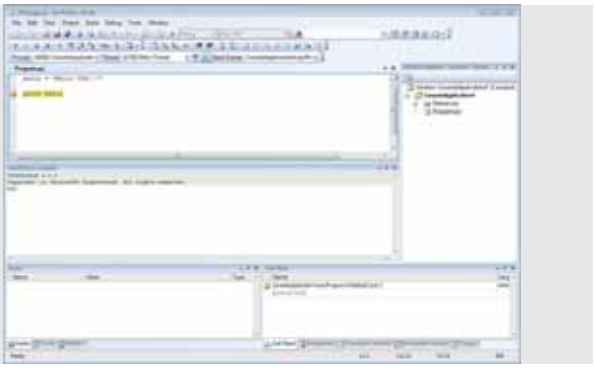
N°105 - Février - Parution : 31 janvier 2008

SGBD
Les nouveautés et tendances 2008

L'offensive Microsoft
L'éditeur renouvelle toute sa gamme : outils de développement, base de données, serveurs...

IDE

Un outil pour IronPython disponible



Sur CodePlex, depuis mi-décembre, les amateurs d'IronPython peuvent coder en bénéficiant d'un IDE, IronPython Studio, basé sur le tout nouveau Visual Studio 2008 Shell. IronPython IDE est un outil gratuit. On bénéficie de toute la puissance de l'environnement Visual Studio 2008 dans ses projets Python. Il nécessite pour fonctionner VS 2008 Shell Isolated Mode Redistributable Package.
Site : http://www.codeplex.com/IronPythonStudio

SERVICE

SourceForge.net : demander les services

Vous êtes fan des projets sourceforge mais vous avez besoin de supports, de formations, d'assistance, bref d'avoir du service ? Sourceforge a lancé un véritable supermarché des services pour les projets open source hébergés sur sa forge. Actuellement, plus de 1200 services sont proposés sur des centaines de projets. Les prix sont indiqués, ainsi que l'éditeur ou la personne proposant le service. En cliquant sur le service recherché, on accède au détail : prix, spécialité, domaines couverts, le niveau de support, la ou les langues, la durée, et les conditions de paiement (chèque ou paypal par exemple). Il s'agit d'un bon moyen de développer les services des logiciels open source et surtout pour les trouver rapidement.

Agenda

JANVIER

Le jeudi 10 janvier 2008 de 11H à 21H. Palais des Congrès – Porte Maillot – Paris - **Salon Les Jeudis - Emploi Informatique & Ingénierie**
www.lesjeudis.com

Le mardi 15 janvier 2008, CNIT, Paris La Défense. **BEAConvergence Day 2008** « en aparté » (7e édition)
http://www.beaconvergenceday2008.com

Le dimanche 20 janvier 2008
Salon Studyrama des Métiers de l'Informatique
http://www.studyrama.com/salons_details.php3?id_article=14157

29, 30 et 31 janvier 2008 CNIT – Paris-La Défense **Solutions Linux / Open Source 2008**.
www.linuxsolutions.fr

FEVRIER

Le 5 février, Paris 17e , Espace Champerret
Salon emploi des informaticiens ingénieurs IT.
www.kavesta.fr

Paris, Palais des Congrès, Porte Maillot. Du lundi 11 février 2008 au mercredi 13 février 2008
Microsoft TechDays 2008
Le rendez-vous incontournable des développeurs et professionnels de l'informatique
http://www.microsoft.com/france/mstd08

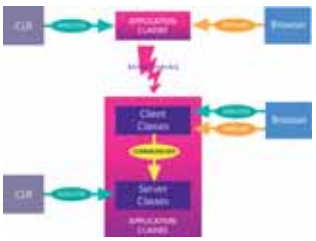
20 et 21 février 2008 - CNIT Paris La Défense **SALON SOLUTIONS INTRANET & TRAVAIL COLLABORATIF**
http://www.groupesolutions.fr

WEB

Sur le CD ROM

Microsoft s'attaque à GWT avec le projet Volta

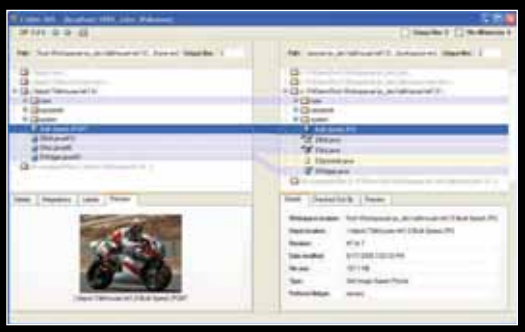
Après quelques mois de rumeurs, cette fois-ci, Microsoft prépare sérieusement sa riposte à la solution Google GWT, avec la première CTP du projet Volta. Cette technologie s'utilise avec Visual Studio 2008 et le framework 3.5. Pour faire simple, Volta est un ensemble d'outils de développement pour créer des applications web multi-tiers utilisant les techniques et outils actuels. Volta fait partie de la stratégie Software + Service de Microsoft. Côté développeur, il s'agit de créer une application cliente .Net avec des portions applicatives tournant sur le serveur. Le client tournant lui, sur le navigateur. Il est possible de cibler le navigateur et une CLR. Nous rentrons ainsi dans l'univers des applications composites. On peut même mixer les deux modes de fonctionnement. Le modèle de développement de Volta est le suivant : par nature, Volta est un recompilateur. Il travaille sur le MSIL de .Net et non sur un langage propre. Cela veut dire que Volta « traduit » le projet dans les langages adéquats. À l'heure actuelle, il se limite à JavaScript et au MSIL. Cela donne une structure, une architecture bien plus souple. Dans cette notion de recompilation, on trouve 3 notions fortes : refactoring, re-ciblage et remodularisation. Volta a pour ambition de simplifier l'application à déployer, à distribuer et ce n'est plus au développeur de le faire. Ainsi, avec une application typique .Net codée dans son langage, avec les librairies nécessaires, en passant par Volta, on évite d'exposer trop de choses. Dans le monde web, on a souvent besoin d'être multi technologies : PHP, Flash, C#, Java, SQL, CSS, etc. Et donc, Volta se propose d'unifier un peu tout cela en réduisant les contraintes de l'application distribuée. C'est possible dans la notion interne au projet d'architecture refactoring et donc de favoriser l'emploi d'une application .net dans un environnement plus vaste. Pour être plus clair, Volta c'est : une indépendance de langage, une chaîne de développement .Net classique, un apprentissage limité, la possibilité d'exposer l'existant, d'aller au-delà du 2-tiers. Bref, avec Volta, on ne sera plus obligé de choisir son architecture, son langage, sa plate-forme d'exécution. Volta offre donc une abstraction d'exécution et de code très intéressante ! C'est pour cela que le projet supporte Internet Explorer et Firefox (pas encore Safari), que le debug se fait sur les navigateurs supportés, que l'on accède aux fonctions et propriétés des navigateurs. Mais alors qu'est-ce qu'une application Volta ? En réalité, Volta propose deux éléments : l'application (application autonome Volta) et le contrôle (groupe d'éléments d'interface réutilisable par d'autres applications). L'application Volta se présente comme une application Winform. Le contrôle Volta propose des éléments personnalisés d'interface, définis en HTML.



Comme le précise Microsoft, Volta est encore expérimental, donc il n'est pas question d'exploiter cette plate-forme dans ses applications. De nombreuses limitations existent : support de VB très incomplet, optimisation du compilateur à réaliser, pas de multithreading, pas de debug multiple, etc. Il est prévu que Volta puisse supporter directement Live. Il faudrait aussi définir les interactions possibles entre Volta et Silverlight.
Site : http://labs.live.com/volta

La fonctionalité Folder Diff,

un atout de productivité du système de GCL Perforce.



Folder Diff de Perforce

Folder Diff est un outil interactif d'affichage en juxtaposition permettant de comparer l'état de deux groupes de fichiers.

À l'aide de Folder Diff, on peut rapidement déterminer les différences entre les fichiers situés dans des dossiers, des branches, des étiquettes ou sur votre disque local. Cette fonction est particulièrement utile lorsque vous devez réaliser des fusions de codes complexes.

De plus, si vous travaillez deconnecter, Folder Diff facilite la synchronisation des données avec celles du serveur Perforce lorsque vous vous connectez de nouveau au réseau.

Folder Diff n'est qu'un des nombreux atouts de productivité offerts par le système de GCL Perforce.

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.

LIBRAIRIE

Trolltech et KDE main dans la main

Cette collaboration concernera en tout premier lieu le projet Phonon, initié par KDE et qui fait partie des profondes modifications de KDE 4. Trolltech a permis de rendre disponible Phonon sur l'ensemble des systèmes majeurs du marché, comme Windows et Mac OS X. Le code modifié ou rajouté est reversé au projet KDE et est sous licence LGPL. Phonon sera aussi intégré dans le futur Qt 4.4 prévu à la fin du premier trimestre 2008. Phonon est une nouvelle librairie multimédia.

IDE

Red Hat dévoile JBoss Developer Studio

L'éditeur Red Hat et sa filiale JBoss viennent de dévoiler un outil de développement basé sur Eclipse : JBoss Developer Studio. Il implémente des outils de conception et d'exécution. JBoss Developer Studio est une solution simple, puissante et conviviale à l'attention des développeurs utilisant JBoss Enterprise Middleware. Cet environnement couvre tout le cycle de vie des applications, de leur développement à leur déploiement, permettant ainsi aux entreprises de s'adapter plus rapidement aux nouvelles conditions du marché. Il inclut les outils Eclipse, la plate-forme JBoss Enterprise Application Platform intégrée, ainsi que Red Hat Enterprise Linux pour le développement et l'accès illimité à Red Hat Network. Developer Studio prend également en charge un certain nombre de technologies, dont Java EE, JBoss Seam et Ajax, Hibernate et Persistence, JBoss jBPM, Struts et Spring IDE. Les développeurs apprécieront la stabilité de cette solution aux outils compatibles, facilement mise à niveau, avec l'avantage d'utiliser le même environnement pour le développement et la production. Il est disponible par abonnement pour Windows et Linux (99 dollars).

SERVICE

Linagora assure MySQL

Décidément, les services commerciaux autour des outils libres / open source se développent rapidement. Linagora, SSII française, a dévoilé mi-décembre une assurance logicielle pour MySQL Enterprise 24/7. Le Groupe LINAGORA et MySQL répondent à cette attente et décident ainsi de renforcer leur partenariat avec la mise en place d'un accord portant sur l'intégration de MySQL Enterprise Unlimited dans l'offre « Open Source Software Assurance » (OSSA) du groupe LINAGORA. Cette nouvelle offre permet aux grands comptes :

- d'utiliser MySQL Enterprise en version illimitée et sécurisée à un prix fixe par an sur 3 ans (nombre infini d'utilisateurs, de processeurs, de serveurs et d'applications) ;
- D'obtenir un support avec engagement de résultat 24/7 en français via un numéro vert gratuit et unique ;
- De maîtriser entièrement le coût total de possession de leurs bases de données.

CONFÉRENCE

La journée MD Day : beau succès !

Le 30 novembre dernier, une journée entière était dédiée au développement piloté, le MD Day. Le succès a été au rendez-vous malgré un thème pas forcément très vendeur. La conférence était co-organisée par cinq des principaux acteurs français de ce marché : Obeo, Lyria, Mia-Software, Objecteering Software et Objet Direct. La journée se découpait en plusieurs sessions techniques avec la modernisation applicative, l'agilité, le prototypage, la SOA, DSM ou encore la méta modélisation, etc. Ce fut aussi une occasion de faire un état du marché. Chaque conférence produit était divisée en deux parties, une étant consacrée à l'offre elle-même et l'autre au témoignage d'un client ayant mis en oeuvre cette technologie. De nombreux experts reconnus étaient présents : Xavier Blanc, maître de conférence à Paris VI, Dominique Vauquier, auteur de la méthode publique Praxème, ou encore François Mérand, responsable de Groupe Architectes au sein de la division DPE chez Microsoft, en parlant notamment des DSL. Bien entendu, Philippe Desfray d'Objecteering Software, un des meilleurs spécialistes français du MDA / UML était présent. Les présentations de cette journée sont disponibles sur www.mdday.fr.

VEILLE

Valtech Days 2008 : déjà en gestation

Après le beau succès de l'édition 2007 de l'automne dernier, Valtech travaille à définir la prochaine édition, avec une nouvelle innovation sur le format de la conférence, comme cette année avec l'espace ouvert (open space).La méthodologie agile restera au cœur de l'approche de Valtech mais l'intérêt pour la RIA est grandissant et devrait occuper une part non négligeable et notamment sur les impacts de la RIA sur son système informatique. Il devrait aussi y avoir un thème sur les ESB Open Source qui connaissent un succès de plus en plus important. Les langages dynamiques ne seront pas non plus oubliés.

STATISTIQUES

PHP 5.2.x : le quart des utilisateurs

Nexen a livré ses statistiques mensuelles sur l'utilisation de PHP dans les sites web. Si la lignée 4.x de PHP continue être très présente, elle se limite désormais à environ 31 %. Tout le reste étant en PHP 5.x. Et la version 5.2.x dépasse aujourd'hui les 27 %, en attendant l'apparition de la 5.3.

UTILITAIRE

ALLCapture en version 3.0

Si vous avez besoin de capturer en vidéo les actions de votre écran, ALLCapture est une des solutions les plus connues. La version 3 est maintenant disponible. Parmi les nouveautés, on trouve de nouvelles fonctions d'enregistrement, des barres de navigations plus nombreuses, l'apparition de l'effet zoom, la durée illimitée d'enregistrement. Disponible en version « standard » et entreprise.

Sur le CD ROM

LANGAGE

Paris on Rails 2007 et Rails 2.0

Depuis le 6 décembre dernier, la version 2.0 du désormais célèbre Ruby on Rails, est disponible en version finale ! Attendue depuis plusieurs mois, elle consolide les fonctions et les acquis des versions 1.x. Un important travail a été réalisé sur les ressources, notamment dans tout ce qui est RESTful. Côté multivue, Rails 2 sépare le format du template du moteur de rendu. Ainsi, on peut donner à son application une interface iPhone… Des améliorations concernent aussi le support de HTML, la sécurité (par exemple : gérer une attaque XSS), la gestion des exceptions mise à niveau, l'apparition d'un nouveau stockage des sessions par un mécanisme de cookies (la session n'est plus gérée dans une base). Un gros travail a été réalisé sur Active Record pour l'alléger et l'optimiser, ainsi la désérialisation XML est supportée. Rails 2 réintroduit le debugger (installable via un simple gem, le format d'installation de Rails). Pour toute migration, il faut d'abord installer la version 1.2.6 avant de passer à la 2.0. Quand à la sortie de la version 2.0 de Ruby, rien n'est encore fixé. Une première version devrait apparaître vers la mi-2008. En France, Rails demeure dans l'ombre de PHP qui est largement dominant, une spécificité française, car la communauté PHP y est très nombreuse. Les organisateurs ont précisé que quasiment 50 % des présents étaient développeurs PHP ! Les développeurs Java arrivent en second avec un bon tiers, et le reste se partage entre .Net, Python, Perl. La montée en puissance des développeurs Java se fait sentir depuis quelques mois, notamment avec la promotion active de Sun autour de Jruby. Sun a d'ailleurs offert à tous les participants un DVD contenant les outils pour bien démarrer la programmation Ruby. On s'attend à une forte présence de Jruby en 2008. Et Sun veut profiter du mouvement pour compléter les possibilités de développement web. L'intégration de Rails dans NetBeans 6 prouve d'ailleurs le soutien actif de l'éditeur. Mais .Net n'est pas en reste et nous avons eu droit à une piqûre de rappel avec le projet IronRuby. Durant la journée, les organisateurs ont voulu alterner : sessions techniques et retour terrain avec Wifirst et surtout Dexia qui utilise énormément Rails pour ses nouveaux projets. Le marché Rails en France se porte bien mais il y a un manque de compétences Ruby / Rails et la demande dépasse assez largement les possibilités, comme le notent Richard Piacentini et Laurent Julliard de Nuxos Group. Site : http://paris.onrails.info/

INTERFACE

KDE 4.0 : finalement pour janvier 2008

KDE 4.0 a subi quelques semaines de retard à cause de plusieurs blocages dans le développement ayant pris plus de temps que prévu. La version finale devrait être disponible tout début janvier (si ce n'est déjà fait quand vous lirez ces lignes). Les nouveautés sont nombreuses. On peut citer Plasma qui doit offrir une nouvelle vision du bureau et de l'interface du système. Plasma tirera parti des cartes graphiques nouvelles générations. L'utilisateur disposera de nouveaux éléments graphiques et concepts comme le panneau, les applets, les extenders. On disposera aussi d'un KDE Developpement Platform pour proposer aux développeurs une plate-forme contenant tous les éléments pour bien développer en KDE. À noter qu'Open SuSe 10.3 intègre déjà une pré-version de KDE 4, la v11 inclura en toute logique KDE 4.1.

Forum PHP 2007 : le web 2.0 à l'honneur

Pour sa 7e édition, le Forum PHP Paris 2007 a mis l'accent sur les architectures distribuées du web 2.0. L'événement a également été l'occasion de préciser la roadmap des versions 5.3 et 6.0.

Malgré les grèves, le Forum PHP a réuni 400 développeurs les 21 et 22 novembre dernier à Paris. Le web 2.0 était au cœur de cette 7e édition, avec deux axes majeurs : les clients riches internet (RIA) et la montée en charge des nouvelles architectures participatives. Côté RIA, Adobe et Microsoft ont montré comment améliorer l'ergonomie des applications PHP grâce à Flex et Silverlight. Les deux éditeurs cherchent clairement à prendre position dans la communauté avant l'arrivée de leur principal concurrent : HTML 5. Supporté par les navigateurs de dernière génération comme Firefox 3, *" HTML 5 fournira nativement les mêmes classes de services que les RIA propriétaires : cache de données local, balises audio et vidéo, etc. "* explique Perrick Penet, membre de l'AFUP et gérant de l'éditeur No Parking. Et comme l'a brillamment démontré Stéphane Deschamps (Orange), les clients riches internet ne facilitent pas l'accessibilité au contenu pour les déficients visuels et les aveugles. HTML 5 pourrait donc proposer une réponse intermédiaire.

Quelle que soit la technologie retenue, l'architecture des sites web 2.0 impose de nouvelles méthodes de programmation côté serveur. *" Les développeurs PHP vont devoir évoluer vers une architecture applicative orientée services, dans le but de fournir une API accessible par divers protocoles de communication (REST, SOAP, XMLRPC) et capable de gérer différents formats de données (XML, JSON, etc..) "* a rappelé Raphaël Rougeron (CCI de Paris) lors de sa session. Pour gérer efficacement les échanges entre un RIA et les serveurs, le framework AMFPHP semble s'imposer. Il simplifie en effet la gestion des échanges client-serveur à l'aide de nombreux protocoles tels que Flash / Flex Remoting, JSON et XML-RPC.

Outre ces nouvelles architectures d'échange de données, le caractère viral et participatif des sites web 2.0 induit une montée en charge souvent bien plus rapide qu'un site web traditionnel. Après un retour d'expérience de WAT.tv (filiale de TF1), Brian Shire, directeur technique de Facebook a expliqué comment utiliser efficacement APC (il participe à son développement) en s'appuyant sur l'exemple de Facebook (55 millions d'utilisateurs). *" Brian Shire n'a pas traversé l'Atlantique pour nos beaux yeux. Avec l'architecture distribuée de Facebook et d'OpenSocial, n'importe quel développeur peut se retrouver avec des dizaines de milliers d'utilisateurs en quelques semaines "* explique Perrick Penet. Les ténors du web 2.0 évangélisent donc à tout va en ce moment. Car la tenue de la charge chez leurs partenaires conditionne aussi leur succès.

PHP 5.3 et 6.0 disponible début 2008

Enfin, le créateur de PHP en personne, Rasmus Lerdorf, a présenté les nouvelles fonctionnalités intéressantes de PHP 5.3 et 6.0

70% des serveurs de production sous PHP 4.x

Selon Damien Séguy (Nexen) qui réalise un suivi mensuel précis des différentes versions en production (28 millions de serveurs testés tous les mois !), PHP 4 représente toujours plus de 70% des serveurs PHP déployés. *" Mais le support de PHP 4 s'achèvera le 31 décembre 2007 "* prévient-il. Comme le montre la progression du nombre d'installations PHP 5.x, les entreprises anticipent cette date butoir depuis un an environ. Pour en savoir plus : www.nexen.net/chiffres_cles/phpversion

WampServer 2.0 disponible

Sur le CD ROM

Initié en 2004, WampServer figure désormais dans le Top 50 des projets Sourceforge. Plus de 3 millions de copies de cette distribution Apache MySQL PHP pour Windows ont déjà été téléchargées. Succédant à WAMP5, WampServer 2 a été complètement revu. *" On peut désormais installer et utiliser n'importe quelle version de PHP, Apache ou MySQL en un clic "* explique Romain Bourdon (Anaska) à l'origine du projet. Chaque développeur peut donc reproduire fidèlement son serveur de production sur sa machine locale en quelques minutes. WampServer 2 est aussi très intéressant pour migrer de PHP 4 vers PHP 5 puisqu'il permet de tester son application PHP 4 sous PHP 5 en un clic. WampServer 2 supporte PHP 4.12 à 5.2.5, Apache 1.3.35 à 2.2.6, et MySQL 4.1.2 à 5.0.45

LEONARDI

L'ihm

EN TOUTE SIMPLICITÉ !

ATELIER "QUICK START" LEONARDI 24 ET 25 JAN. 500€ HT REMBOURSABLES.

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*"...

TRES BONNE ANNEE 2008 A TOUS LES LECTEURS DE PROGRAMMEZ!

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

Malgré les grèves, plus de 400 personnes ont assisté au Forum PHP 2007.

pour la création d'un site web de nouvelle génération. La version finale de PHP 5.3 sortira au premier trimestre 2008 à quelques semaines d'intervalle avec la première bêta publique de PHP 6.0 (prévue pour 2009).

La réécriture complète de PHP 6.0 pour supporter Unicode ayant pris plus de temps qu'annoncé, de nombreuses fonctionnalités prévues pour PHP 6.0 seront intégrées dans PHP 5.3. Parmi les évolutions intéressantes pour le web 2.0 on peut citer les extensions XMLReader et XMLWriter présentes depuis PHP 5.1 mais qui seront intégrées nativement à PHP 5.3 et le support d'OpenID dès PHP 5.3.

Les autres évolutions touchent au langage lui-même (nouveaux itérateurs, late static binding, et __callStatic notamment), ainsi qu'à l'organisation du code avec l'apparition très attendue des namespaces (espaces de noms). En regroupant les classes par domaine, les namespaces permettent d'obtenir un code plus propre (plus besoin de préfixer les classes) et donc plus facilement réutilisable.

Voir les présentations du Forum PHP 2007

Tous les résumés de la conférence sont accessibles à cette adresse : http://www.afup.org/pages/forumphp2007/resumes.php

■ **Frédéric Bordage**

Demain : déjà des hypothèses pour PHP 7

A quoi ressemblera PHP 7 ? A mesure que l'architecture du web 2.0 devient une réalité, quelques scénarios plausibles se dessinent.

Ancien président de l'AFUP et fin connaisseur de PHP 6, Perrick Penet (perrick@noparking.net) nous donne son avis.

Perrick Penet : *La distribution des réseaux sociaux éclatés sur de multiples sites (*mes amis *sur Facebook,* mes contacts professionnels *sur LinkedIn,* mes musiques préférées *sur MySpace, etc.) pose des problèmes d'architecture complètement nouveaux car nous assistons à la naissance d'un gigantesque système de fichiers distribués à l'échelle de la terre entière.*

Heureusement, un projet Open Source particulièrement prometteur – Hadoop – pourrait rendre les choses beaucoup plus aisées pour le commun des développeurs. Il s'agit d'un système de fichiers distribués capable de gérer des données dont l'unité ne serait plus ni le Go (une clé USB), ni le To (un serveur de stockage), mais Po (1000 To) !

Ce projet s'inspire de MapReduce, une technique de programmation qui permet d'effectuer des calculs sur un très grand nombre de données (souvent plus d'un To), initialement développé par Google pour gérer les gigantesques données nécessaires à leur moteur de recherche.

Attendons-nous à voir des fonctions PHP pour en profiter. Qui sait ? La présence des créateurs d'Hadoop et de PHP dans la même société – Yahoo! – facilitera peut être les choses…

PHP 5.3 et 6 : deux versions majeures

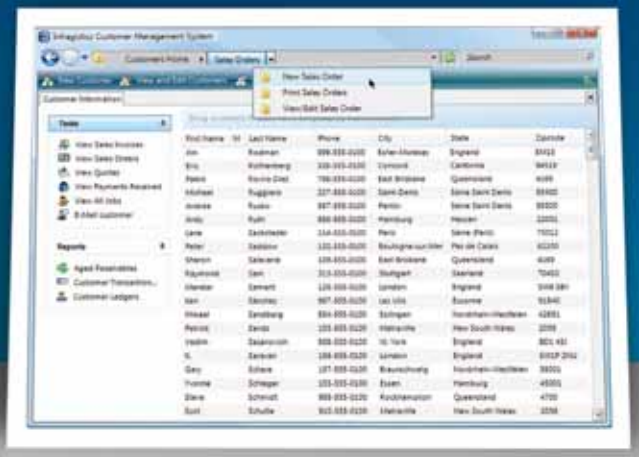
Sur le CD ROM

PHP 6 mettra plus de temps que prévu à voir le jour. Certaines fonctionnalités dont l'intégration était initialement prévue dans PHP 6 seront finalement " backportées " dans PHP 5.3, soit nativement, soit sous la forme de patch. PHP 5.3 version finale et PHP 6 alpha sont attendues pour le premier trimestre 2008. Tour d'horizon des principales nouveautés des deux prochaines versions majeures.

Nouveauté	Description	PHP 5.3	PHP 6.0
Namespace	Les espaces de noms permettent d'organiser son code logiquement afin de faciliter sa réutilisation et d'améliorer le nom des fonctions et des objets qui n'ont plus à être préfixés.	X (patch)	x
XMLWriter et XMLReader	Comme son nom l'indique, XMLWriter facilite la production d'un code XML. XMLReader propose de son côté un parser SAX. Ces deux extensions sont intégrées nativement à PHP 5.3.	X	x
OpenID	Le support du protocole d'identification unique OpenID sera intégré dans la 5.3 via une librairie PECL. OpenID permet de partager des données d'identification entre différents sites web tiers.	x	X
Librairie objet	Le modèle de programmation objet apparu dans PHP 5 continue à être amélioré. Les itérateurs permettront notamment de construire des boucles for each plus précises grâce à l'ajout de structures par défaut plus évoluées.	x	x
Late static binding	Permet de référencer une classe appelée dans un contexte d'héritage static	x	x
__callStatic		x	x
MysqInd	Drivers MySQL natifs pour PHP s'appuyant directement sur le Zend Engine donc plus rapides que la librairie C et nécessitant moins de mémoire. L'intégration permettra aussi de faciliter le debug et de créer de nombreuses fonctions intéressantes comme un cache de requête côté client.	x	x
Unicode	PHP gère pour l'instant les chaînes de caractères comme des blob ce qui complexifie inutilement leur manipulation, notamment la gestion native de plusieurs alphabets étendus (français, hébreu, chinois, etc.) au sein d'une même application.		x
Cache APC	Le cache d'opcode APC restera une librairie PECL sous PHP 5.3. Il ne sera intégré au noyau de PHP qu'avec la version 6. A la demande de Zend qui propose des solutions commerciales concurrentes, il sera désactivé par défaut.		X

" Je mène **une direction stratégique**
à travers une initiative technologique. "

- Directeur technique



NetAdvantage® for Windows® Forms

Une expérience multi-plateformes consistante pour vos utilisateurs.

NetAdvantage

Amplifiez votre passion en créant des interfaces utilisateur exceptionnelles avec NetAdvantage

Donnez du pouvoir à vos utilisateurs – Délivrez des interfaces utilisateur riches et productives à vos clients, pour Windows Forms, ASP.NET, WPF ou JSF

Profitez d'architectures réutilisables – Uniformisez vos méthodes de développement avec des frameworks et outils consistants

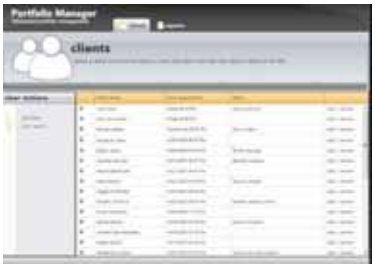
Garantissez un look & feel cohérent – Grâce à notre composant Application Styling™ permettant de peaufiner vos propres applications (des styles conçus professionnellement ou créez votre selon vos chartes graphiques).

Une assistance technique globale – Communiquez avec des techniciens à Londres, New York, Tokyo et Bangalore pour un support complet de nos produits par téléphone, email et messagerie instantanée 24h/24

Maximisez vos résultats – TestAdvantage automatise le test de l'interface utilisateur de vos applications créées avec NetAdvantage. Recevez une assistance claire, une formation complète et des services de consultation sur mesure

NetAdvantage® for ASP.NET

NetAdvantage® for WPF



NetAdvantage® for JSF

Infragistics®
Powering The Presentation Layer

Pour de plus amples informations:
infragistics.com/dotnet
sales-europe@infragistics.com

N° Vert 0800 667 307

IBM Business Partner

Urbanisation des applications : des dll à la SOA

SOA est une approche d'urbanisation logicielle permettant de construire une architecture de communication entre applicatifs hétérogènes. Cet article présente une "approche" SOA dans le sens où il n'est pas question ici de faire communiquer des applicatifs hétérogènes mais d'urbaniser les traitements en services homogènes constituant les couches Métier et Data Access des applications.

Les services en question prennent la forme de Services Windows accessibles via .NET Remoting et présentant des Interfaces. Seuls les services implémentent celles-ci, les IHM se contentant d'utiliser ces Interfaces et demandant aux services de leur fournir les implémentations adéquates.

Un nouveau développement, une nouvelle architecture, une nouvelle fonctionnalité, tous répondent généralement à un besoin ou un problème. Voilà pourquoi je commencerai par exposer les raisons pour lesquelles j'ai pensé à cette architecture.

Gérer la dll

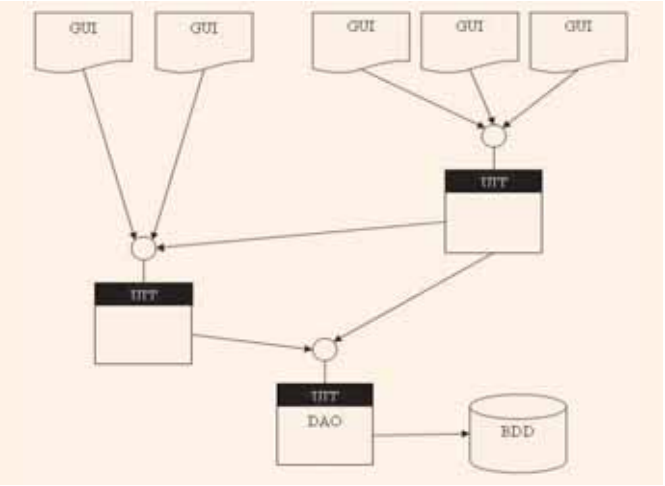
Avec son GAC (Global Assembly Cache) ou son déploiement facilité, le .NET promettait de libérer le développeur de « l'Enfer des DLL », terme communément admis pour caractériser le casse-tête à résoudre lors des déploiements et mises à jour, des exécutables et des .dll associés à leurs produits. En effet il est désormais inutile d'enregistrer ces très chères bibliothèques. Grâce au .NET vous pouvez, au choix, les déployer dans le GAC ou directement avec l'exécutable.

Dans le cas le plus simple, la .dll est copiée dans le même répertoire que l'application qui l'utilise, il suffit de mettre à jour le fichier et tout se passera bien. Nous en sommes arrivés à la panacée du déploiement facilité. Si j'ai un bug, je modifie ma bibliothèque, je la compile et je la copie, l'IHM qui l'utilise n'y verra que du feu… à partir du moment où les interfaces publiques des classes présentent au moins les mêmes propriétés et méthodes publiques. Mais si j'ai plusieurs applicatifs qui utilisent la même .dll je me vois obligé de copier celle-ci auprès de chacun d'entre eux. Vous me direz que ce n'est pas si grave, je vous rejoins en précisant : tant que le nombre d'applicatifs reste faible et qu'ils ne sont pas trop dispersés sur le SI du client, car sinon l'enfer recommence.

La grande solution est donc de déployer les .dll dans un endroit où les applicatifs sauront où les chercher, j'ai nommé le GAC. C'est très pratique il faut le reconnaître. Mais le GAC permet de faire cohabiter des .dll identiques de version différente. Dans ce cas, les applicatifs dont on n'aura pas modifié les références à cette .dll, continueront d'utiliser l'ancienne version. On en revient au cas ci-dessus.

Bref, en définitive, j'ai besoin d'avoir ma .dll dans un endroit unique sans avoir à intervenir sur les liens entre celle-ci et les applications qui l'utilisent à chaque fois que je la modifie.

Dans un projet d'Intranet développé en ASP.NET sur lequel j'ai participé, j'avais développé une application winform destinée à recevoir des ordres d'impression via Remoting.NET. Le jour vint où l'on me demanda de transformer la partie « réception et impression » des commandes en Service Windows de manière à ce que l'on ne soit pas obligé d'ouvrir



une session sur la machine hôte pour que le service d'impression soit démarré. Par la suite, d'autres applicatifs eurent à utiliser ce même service pour leurs commandes d'impression.

Réfléchissant au problème énoncé ci-dessus, j'ai réalisé que j'avais la solution sous les yeux. Si je devais débugger ou ajouter un nouveau type d'impression, j'intervenais sur le Service Windows uniquement sans avoir à m'inquiéter des autres applications.

Dans le même temps je commençais à m'intéresser aux architectures SOA. Mettre en place une telle architecture ne s'improvise pas, c'est un travail d'analyse au niveau du S.I. et de son urbanisation. A mon niveau il n'était question que de rendre les .dll suffisamment découplées des applications utilisatrices pour faciliter la maintenance. SOA est porteuse des principes propres à un tel découplage. Voilà pourquoi j'ai adapté celle-ci à mon problème.

Les unités de traitement

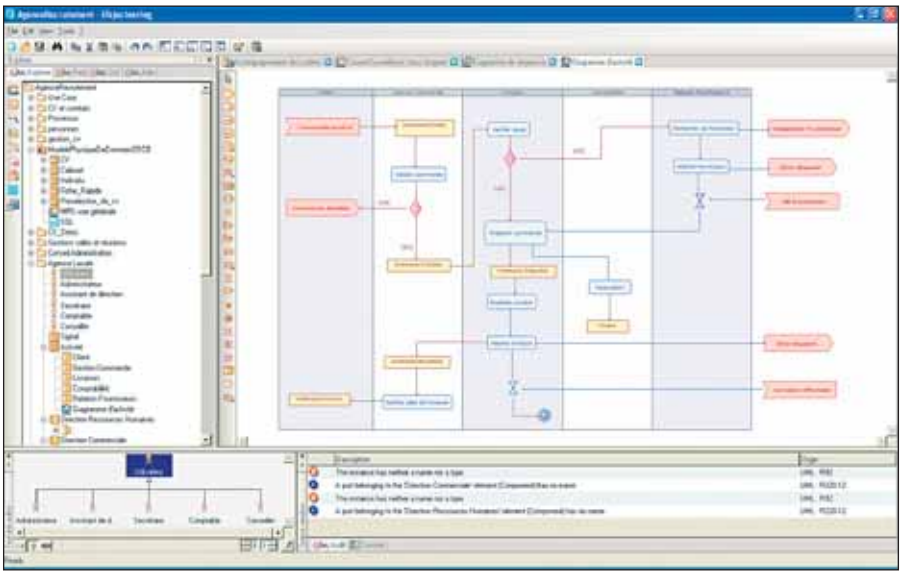
Le principe est donc de transformer des bibliothèques de classes .dll en, ce que j'appellerai, des Unités Individuelles de Traitement. Chaque UIT est donc porteuse de fonctionnalités qu'elles vont exposer vers l'extérieur via des Interfaces d'Entrée. Nous avons donc un principe de Boîte Noire. Comme les UIT doivent aussi travailler entre-elles elles disposent toutes d'une Interface de Sortie qui mappe les Interfaces d'Entrée des autres UIT qu'elles utilisent. L'objectif est double : renforcer la cohérence interne et limiter le couplage avec l'extérieur. Dans un prochain article j'aborderai cette architecture interne qui est composée d'interfaces mais aussi de générateurs d'événements, de piles d'appel, d'outils divers et variés. On se retrouve donc avec une architecture com-

Objecteering 6.1
Le développement guidé par le modèle

NOUVELLE VERSION

Objecteering 6.1 optimise MDA et UML2 pour générer un code d'un haut niveau d'expertise : il maximise la productivité et la qualité des développements en Java, C++ ou C#.

Comment tirer parti au mieux de la modélisation UML à des fins de production automatisée d'un code de qualité, maintenu en cohérence avec le modèle ? Comment guider les développeurs dans leur modélisation et optimiser la production de code pour des architectures orientées services (SOA) s'appuyant sur des frameworks complexes ? L'approche MDA qui consiste à exploiter le modèle par des mécanismes de transformation répond précisément à ces problématiques en assurant également la traçabilité entre le code généré, le modèle dont il est issu et les exigences qui le justifient. Avec Objecteering 6.1, Objecteering Software met à disposition des développeurs une nouvelle génération d'outils de développement guidés par le modèle, en s'appuyant sur les dernières avancées de MDA et de UML2.



Objecteering 6.1 : Diagramme d'activité.

L'expression des besoins intégrée à la modélisation UML

Réussir l'expression des besoins de votre application est le point de départ fondamental pour le succès du développement. Objecteering 6.1 intègre la gestion des exigences avec la modélisation UML. Vous démarrez dès la phase de définition des exigences, et poursuivez sans rupture jusqu'aux modèles d'analyse et conception. Vous pouvez ainsi obtenir un modèle des exigences complet qui vous permettra d'aborder les étapes d'élaboration de votre application sur des bases solides et justifiées par les besoins.

Le maintien en cohérence exigences / modèle / diagrammes / code / documentation

La génération automatique de code pour les cibles Java/J2EE, C# .Net, C++, Corba, Fortran ou SQL, supportant des frameworks tels que Spring, JSF, Struts et Hibernate ou des frameworks spécifiques comme les architectures SOA apporte des gains substantiels en qualité et en productivité. La génération de documentation permet de fournir à chaque type d'acteurs des documents pour les exigences, le modèle et le code. Le référentiel unique est garant de la cohérence, de la traçabilité et de la non redondance des informations.

Une réelle assistance à la construction des modèles

Avec UML2.1 le standard de l'OMG est devenu un langage riche et complet pour couvrir le besoin en modélisation d'entreprise et de systèmes techniques.
Il est de ce fait aisé de commettre des erreurs de modélisation, et d'obtenir des modèles inconsistants.
Objecteering 6 est doté d'un éditeur graphique UML sensitif qui assiste l'utilisateur à construire des modèles corrects dès le début.
L'audit de modèle qui vérifie en temps réel 282 règles sémantiques permet en outre d'assurer la cohérence du modèle dans sa globalité, y compris dans le cadre d'un travail collaboratif sur un modèle partagé.

Nouveau ! Le support BPMN et l'architecture d'entreprise

Objecteering 6.1 intègre le support de BPMN pour modéliser les processus métier en liaison avec le modèle UML. Avec l'extension Objecteering EA, l'architecture d'entreprise, la cartographie des processus, l'urbanisation du SI sont intégrés en continuité dans la modélisation UML, pour adresser l'ensemble des participants à la définition du SI de l'entreprise.

Nouveau ! Une API Java de paramétrage MDA

Avec Objecteering MDA Modeler, vous modélisez vos extensions UML, vous définissez vos transformations de modèle, vos générateurs de code et vous implémentez en Java des comportements dédiés. L'API Java très riche permet de naviguer dans le modèle, de le transformer et de piloter Objecteering 6.1, son IHM, ses diagrammes pour que l'atelier soit dédié à votre environnement, qu'il génère un code spécifique et guide les modélisateurs selon votre méthodologie. Des wizards dédiés permettent de minimiser la programmation Java, et réduisent l'apprentissage du mode de paramétrage de l'outil. La puissance de MDA combinée à la richesse de Java et aux capacités de Objecteering 6.1 décupleront la productivité et la qualité de vos développements.

Objecteering SOFTWARE

Venez découvrir

Objecteering 6.1

Nouvelle Version !

Télécharger sur

www.objecteering.com

Pour plus d'information contactez-nous au

01 30 12 16 60

posée d'applications Front ou Back Office et des UIT. La communication se fait à distance via Remoting.NET.
Voici un exemple que vous pourrez aisément tester chez vous. Il s'agit d'un service qui va stocker une valeur et la retourner au client et/ou l'incrémenter sur demande. En suivant la logique énoncée ci-dessus on a donc à développer des Interfaces, l'Implémentation de celles-ci, un Serveur et un Client. Créer une nouvelle solution sous Visual Studio et quatre projets nommés comme ci-dessous.
Dans le projet Interfaces, créez un fichier IAfficheur.cs :

```
namespace Interfaces
{
    public interface IAfficheur
    {
        void Increment();
        int GetValue();
    }
}
```

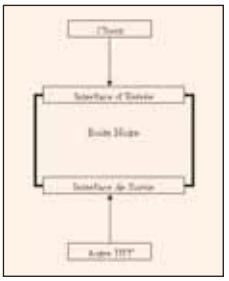
Dans le projet Implementations, créez un fichier Afficheur.cs :

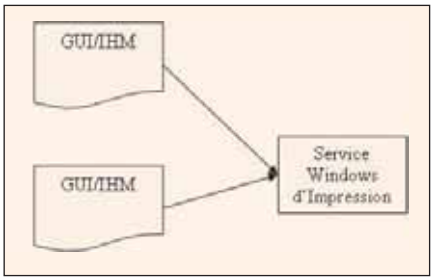
```
using Interfaces;

namespace Implementation
{
    public class Afficheur : MarshalByRefObject, IAfficheur
    {
        int _valeur;

        public Afficheur(){
            _valeur = 0;
        }

        public void Increment(){
            _valeur++;
        }
        public int GetValue(){
            return _valeur;
        }
    }
}
```

Remarquez trois choses : le Using Interfaces, l'héritage de MarshalByRefObject et l'implémentation de IAfficheur.
Maintenant créons le serveur, un programme console dont le fichier Program.cs sera le suivant :

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
using Implementation;

namespace Serveur
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                TcpChannel channel = new TcpChannel(1069);
                ChannelServices.RegisterChannel(channel,true);
                RemotingConfiguration.RegisterWellKnownServiceType(
                    typeof(Afficheur),
                    "Afficheur",
                    WellKnownObjectMode.Singleton);

                Console.WriteLine("Le serveur a démarré avec succès");
                Console.ReadLine();
            }
            catch
            {
                Console.WriteLine("Erreur lors du démarrage du serveur");
                Console.ReadLine();
            }
        }
    }
}
```

Vous pouvez constater qu'il utilise la classe Afficheur et qu'il la rend accessible à distance. Bien que le using ne fasse apparaître que la bibliothèque Implementation, il sera aussi nécessaire de référencer la bibliothèque Interfaces. Enfin occupons nous du client, un projet console lui aussi, dont le fichier Program.cs sera le suivant : voir le code source du projet (CD et/ou site web).





Vous pouvez constater que le client ne référence que la bibliothèque Interfaces. Il se connecte au serveur via remoting et lui demande un objet de type IAfficheur. Il recevra en fait un objet de type Afficheur implémentant IAfficheur mais ne s'en rendra pas compte.
Imaginons maintenant que de nombreux mois après le déploiement de cette solution, il faille modifier la façon dont l'incrément est calculé. En gros, on change la commande « _valeur++; » en « _valeur = _valeur + 1; ». Il suffit de recompiler le projet Implémentation, de remplacer la bibliothèque obsolète par celle-ci auprès de l'exécutable du serveur et, le tour est joué. Non seulement cela fonctionne bien mais en plus nous ne serons à aucun moment intervenus sur les applicatifs utilisant cette UIT.
Ceci était une petite introduction à cette architecture inspirée de SOA et fondée sur le Remoting.NET. Prochainement j'approfondirai les choses en détaillant les architectures d'une UIT et certaines UIT spéciales chargées des redirections, du monitoring et du Log.

Christophe Breysse

NETFECTIVE
Creative Technology

Editeur du logiciel BLU AGE™ et propriétaire de la marque Opteams™

Le groupe NETFECTIVE TECHNOLOGY, spécialiste des technologies Objets, UML® et MDA®, propose un catalogue complet de formations et de séminaires.
La totalité des sessions proposées est disponible en mode inter et intra entreprise.
Retrouvez-nous dans nos locaux à Suresnes (92) ou Bordeaux (33).

Plus d'informations sur www.netfective.com
ou tout simplement contactez-nous au 01 56 05 88 00 / 01 56 05 60 91.

Formations Concepts Objet et UML

AOO-1j **Approche Orientée Objet** (1 jour)
7 janvier 2008; 4 février 2008; 7 mai 2008; 9 juin 2008; 29 septembre 2008...

AC-UML-4j **Analyse et conception avec UML 2.x, OCL 2.x & MDD** (4 jours)
8 janvier 2008; 5 février 2008; 13 mai 2008; 10 juin 2008; 30 septembre 2008...

MD-3j **Application d'UML 2.x avec MagicDraw™** (3 jours)
14 janvier 2008; 11 février 2008; 16 juin 2008; 13 octobre 2008...

Formations Java, Java EE

JAVA-4j **Programmation avec le langage JAVA** (4 jours)
28 janvier 2008; 25 mars 2008; 1er décembre 2008...

J2EE-4j **Développement d'applications web pour la plateforme Java EE** (4 jours)
26 février 2008; 21 avril 2008; 15 décembre 2008...

STRUTS-3j **Développement d'applications web avec Struts** (4 jours)
3 mars 2008; 28 mai 2008; 15 septembre 2008...

JSF-3j **Développement d'applications web avec JSF** (3 jours)
6 mars 2008; 2 juin 2008; 18 septembre 2008...

HIBER-3j **Gestion de la persistance avec Hibernate** (3 jours)
11 mars 2008; 30 juin 2008; 23 septembre 2008...

SPRING-3j **Développement d'applications avec Spring** (3 jours)
17 mars 2008; 15 juillet 2008; 20 octobre 2008...

Formation XML

XML-CM-3j **Technologies XML : Conception et mise en oeuvre** (3 jours)
28 avril 2008, 1er septembre 2008...

Formations BLU AGE™

BLU-INT-1j **Présentation synthétique des outils & méthodes MDD™ de BLU AGE™ (1 jour)**
18 janvier 2008; 15 février 2008; 21 mars 2008; 19 mai 2008; 20 juin 2008; 4 juillet 2008; 5 et 26 septembre 2008; 23 octobre 2008...

BLU-MDD-5j **Une approche pragmatique du MDA : MDD™ avec BLU AGE™** (5 jours)
21 janvier 2008; 18 février 2008; 31 mars 2008; 21 mai 2008; 23 juin 2008; 8 septembre 2008, 6 octobre 2008; 24 novembre 2008...

BLU-SFC-5j **BLU AGE™ Software factory configuration** (5 jours)
7 avril 2008; 21 juillet 2008...

BLU-EMD-5j **EMDD avec BLU AGE™ - Comment construire des applications complexes avec BLU AGE™** (5 jours)
14 avril 2008; 7 juillet 2008; 25 août 2008; 17 novembre 2008; 8 décembre 2008...

Formations 2008

BLU AGE™
J2EE .NET APPLICATION GENERATOR

BLU AGE™, le logiciel innovant issu du département de R&D de NETFECTIVE TECHNOLOGY, permet la génération automatique d'applications Java EE et/ou .NET sans recours aux techniques et ressources de développement.

'You design, We generate'
'Vous dessinez, Nous générons'

Découvrez BLU AGE™
Démonstration en ligne de génération d'application.
Agenda : 8, 15 & 29 janvier. 5, 12 & 19 février 2008 (français)
10 & 24 janvier. 7 & 21 février 2008 (anglais).

Inscrivez-vous gratuitement aux prochains webinars sur www.bluage.com.

Sur le CD ROM

MagicDraw 14.0 et BLU AGE Edition 2008

Modélisation et génération d'application Java EE , .Net

Vous trouverez dans le CD-ROM accompagnant ce numéro de Programmez ! un modèle UML décrivant les fonctionnalités d'une application de démonstration nommée : 'Magic Library'. Cette application permet de gérer les emprunts de livres dans une bibliothèque.

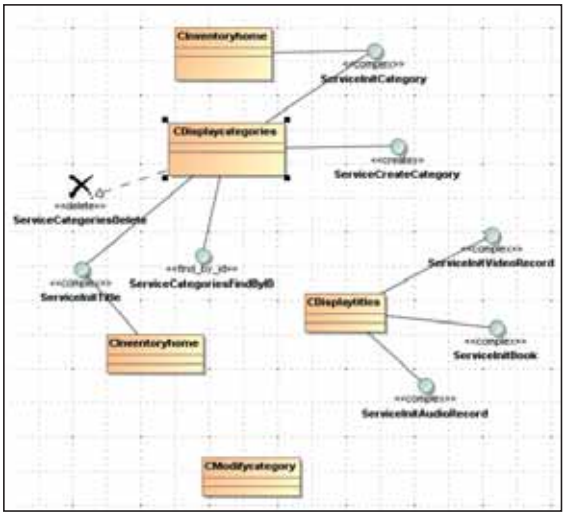
Cette application est générée automatiquement par BLU AGE, et est déployée sur un serveur d'application accessible à l'adresse suivante : http://www.bluage.com/magic.html.

Pour éditer le modèle UML de cette application, vous trouverez dans ce même CD-ROM l'outil de modélisation UML : MagicDraw 14.0, édité par No Magic. Cet outil permet de visualiser l'ensemble des diagrammes UML réalisés pour décrire les fonctionnalités de l'application 'Magic Library' et nécessaire pour la génération de la totalité de l'application.

Présentation de MagicDraw

MagicDraw est un outil de modélisation UML des systèmes orientés objets. Il est destiné aux analystes et concepteurs et supporte le travail collaboratif.

En outre, il possède une interface intuitive et facile à prendre en main avec un chargement rapide des diagrammes. Il se distingue par l'implémentation du standard UML2.1 et le respect des normes de modélisation. Ainsi, l'export des modèles UML répond aux recommandations de l'OMG avec l'implémentation de la norme XMI. De plus, grâce à une version 'TeamWork Server', il permet à plusieurs concepteurs de partager un même modèle avec un référentiel de sources pour sauvegarder toutes les modifications et l'historique des changements.



Présentation de BLU AGE

BLU AGE est une suite logicielle qui permet de générer automatiquement et intégralement des applications web sur les environnements Java EE et .Net. Elle présente la particularité d'intégrer la totalité d'un projet applicatif, depuis la conception et la méthodologie associée, jusqu'à la génération automatique de l'ensemble du code source et l'intégration des applications générées au sein de l'environnement technique cible.

L'outil se base sur les fondamentaux MDA (Model Driven Architecture) édictés par l'Object Management Group. Cette approche permet de se focaliser sur la construction fonctionnelle indépendamment de l'environnement technologique. Le modèle UML est transformé grâce aux BSP (BLU AGE Shared Plug-in) pour générer automatiquement le package final. Ce dernier inclut le code source généré, la documentation complète de l'application et le script de création de la base de données.

La génération se base sur les input suivants :

- Modélisation détaillée sous forme de diagrammes UML,
- Maquette statique sous forme de fichiers XHTML,
- Et le descriptif détaillant l'architecture technique cible et l'environnement de déploiement.

La modélisation UML est conforme au méta-modèle de BLUAGE, qui correspond à un sous ensemble de l'UML 2.1 enrichi au travers de stéréotypes et valeurs balisées. Le principe de transformation se base sur la spécification QVT (Query View Transformation) préconisée par les designs patterns de l'OMG. Le modèle UML comprend des diagrammes de cas d'utilisation qui permettent de décrire les packages fonctionnels et les profils utilisateurs associés. Des diagrammes d'activités sont utilisés pour détailler l'exécution de chaque cas d'utilisation, les entités et les objets métiers sont spécifiés dans des diagrammes de classe. Quant aux services, ils sont modélisés dans des diagrammes de classe qui permettent de référencer les services réalisés par chaque processus fonctionnel. D'autres diagrammes UML sont utilisés au besoin suivant la complexité métier des applications générées. Les composants du modèle UML facilitent l'accès à une approche purement SOA dirigée par les modèles et orientée vers les services. Cette approche facilite l'intégration des applications générées dans les systèmes d'informations à travers, notamment, des Web Services dont la description se fait dans le modèle UML.

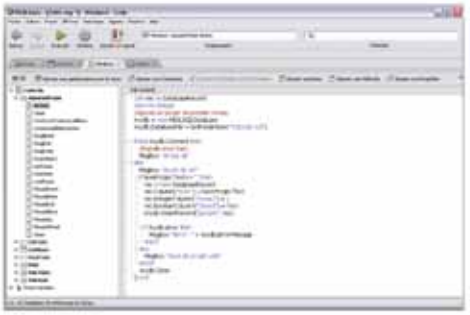
Pour répondre au besoin d'agilité, BLU AGE intègre un outil Web de gestion de projet : BLU AGE Config-Runner, basé sur un mode projet itératif. Cet outil accessible par les postes clients, permet de créer des projets de génération, de définir les ressources projets, de créer des itérations et de mettre à disposition le code source généré.

Imad BERNOUSSI
Directeur du marketing technique
http://www.bluage.com

REALbasic 2007

Cross-Platform that Really Works.*

Développez facilement des applications graphiques ou consoles pour Windows, Mac et Linux à partir d'un source unique



Débuggez à distance : Testez votre application sur Mac à partir de votre PC sous Windows ou Linux et vice versa

Accédez facilement à tous les SGBD du marché : MySQL, MS SQL, PostgreSQL, REAL SQL Server, sources ODBC, ….

Développez des application multilingues : REALbasic est totalement Unicode pour gérer tous types d'alphabet et des applications multilingues.

Compilez des applications natives pour Windows, Mac et Linux en quelques clicks

Déployez facilement : REALbasic ne nécessite ni machine virtuelle, ni DLL et génère des exécutables autonomes pour les trois plates-formes.

Développeurs Visual Basic, portez vos applications sous REALbasic très rapidement sur trois plate-formes dans un environnement familier

Environnement avancé : multi thread, XML, http, TCP/IP, SOAP, orienté objet, Server sockets, SSL et beaucoup plus encore

Support technique gratuit : Support technique par email pour plus d'efficacité

-25%

Offre spéciale Programmez

Offre Spéciale réservée aux lecteurs de programmez !

Pour bénéficier de cette offre, rendez-vous directement sur notre site : www.realsoftware.com/programmez.

Pour tout renseignement complémentaire, appelez le 08 70 40 94 92 ou contactez nous par email : commercial@realsoftware.fr

25% de remise sur REALbasic PRO Edition soit l'environnement de développement complet pour seulement 300 Euros H.T., déploiement illimité et support par email compris

Téléchargez la version d'évaluation gratuite sur :

www.realsoftware.com/download

*l'outil cross plate-forme qui fonctionne vraiment

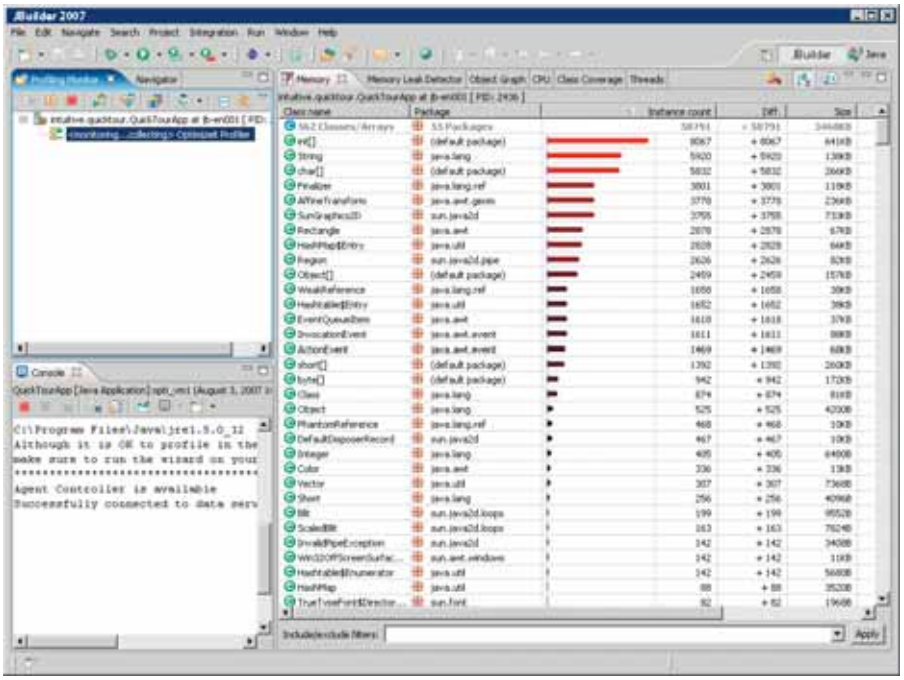
Les plug-in JGear : composants de productivité

JGear est une suite de quatre plug-in destinés à l'environnement de développement Eclipse. Celui-ci est de loin le plus utilisé dans le milieu professionnel des SSII. L'éditeur espère ainsi percer sur un marché où peu d'outils existent.

JGear est une famille de produits conçus pour améliorer la productivité de l'environnement de développement Eclipse. Elle fournit un ensemble d'outils puissants permettant de gagner du temps sur les phases de codage des projets. Il est possible de les installer séparément puisque chaque produit est indépendant. Vous sélectionnerez ceux-ci en fonction des besoins de votre projet.

Etant donné qu'Eclipse fournit un SDK open source permettant de le modifier à volonté, de nombreux éditeurs ont bâti leur offre sur Eclipse. Borland propose notamment JBuilder et d'autres éditeurs tels que BEA et IBM disposent également de solutions. Les plug-in JGear sont fonctionnels avec l'ensemble de ces environnements.

L'ensemble de la gamme est documenté par des présentations vidéo qui permettent d'apprécier rapidement l'intérêt et l'efficacité de tel ou tel module. Pour ceux qui souhaitent aller plus loin, des démonstrations sont disponibles après simple inscription sur le site Internet de CodeGear. L'installation se fera par utilisation du système de mise à jour d'Eclipse. L'ensemble des produits est localisé, vous pourrez donc profiter d'une utilisation et d'une documentation dans la langue de Molière.



JGear LiveSource

Accélérer la productivité des développeurs en fournissant des outils mettant en œuvre l'approche RAD, c'est clairement l'objectif de l'outil JGear LiveSource. Celui ci permet la création de composants applicatifs via une interface graphique et un ensemble de wizards. L'outil supporte notamment les EJB 2.1 et 3 ainsi que les web services. Les serveurs d'applications s'intègrent parfaitement : les configurations spécifiques de chaque serveur sont prises en charge, notamment pour la conception de composants EJB. Que vous utilisiez JBoss, WebSphere ou encore WebLogic, LiveSource saura le reconnaître. Il permet de travailler dans deux modes. La solution la plus efficace se fonde sur la conception du diagramme de classe des composants. La barre d'outils fournit ainsi le nécessaire pour concevoir l'architecture de l'application par simple glisser déposer. Le code est généré par la suite de manière transparente pour le développeur. Celui ci peut accéder aux propriétés des composants par un simple tableau de propriétés. Pour une meilleure granularité, il est possible de travailler au niveau du code source. Les modifications se répercuteront sur le modèle. Les développeurs et les architectes peuvent ainsi faire évoluer le modèle au fil du développement tout en restant synchrone avec le code.

La prise en charge des diagrammes de classes, de séquences et d'états est un vrai plus dans le cadre de développements de composants et d'architectures SOA. Les services web sont parfaitement intégrés, ainsi chaque composant peut être publié en quelques clics sous forme de services web SOAP.

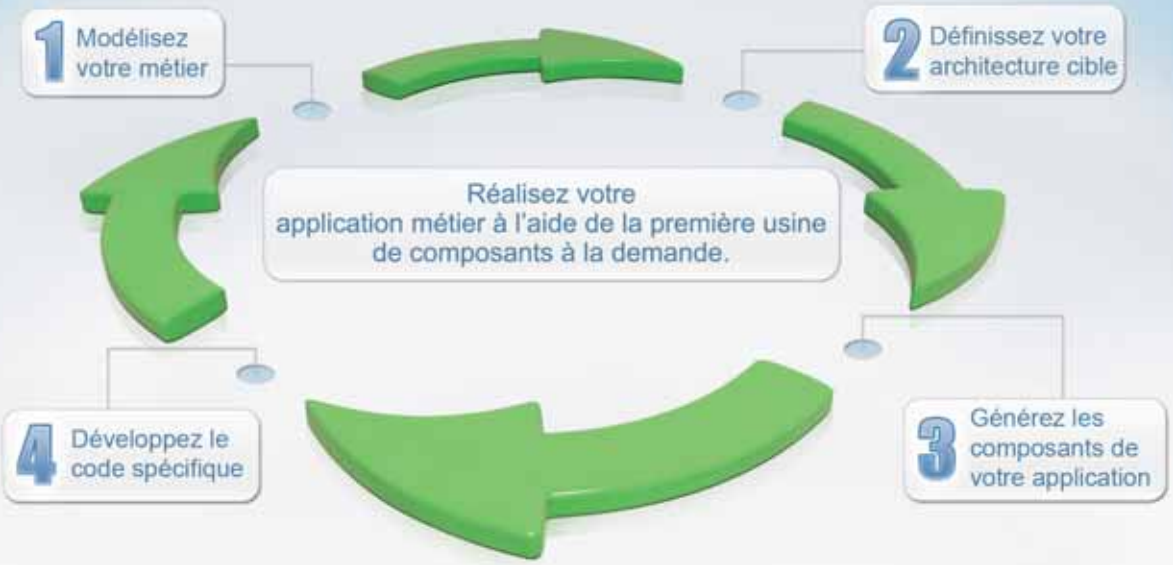
JGear Performance

JGear Performance fournit un débogueur et un profiler évolué permettant de diagnostiquer précisément les comportements des applications et de ses composants. Son utilisation permet la correction des bugs de manière plus efficace et d'améliorer le code source pour en favoriser les performances et la qualité.

Ses fonctionnalités sont larges et couvrent les besoins des développeurs qui travaillent sur un projet complexe et exigeant. On peut citer notamment la visualisation de l'utilisation CPU et mémoire ainsi que la détection des fuites mémoires. L'outil permet également d'assurer

Code Fluent

Accélérez vos développements .NET



CodeFluent.com

La fabrique en ligne est totalement gratuite pour les modèles métier comprenant jusque 30 entités.

nouveau !

Inscrivez-vous dès aujourd'hui sur : http://www.codefluent.com

0€

CodeFluent

Version complète Développeur donnant accès aux codes sources des composants générés.

Prix public : ~~2490€ht~~

Prix spécial lecteurs de "Programmez" : **1990€ht**

CodeFluent est une usine logicielle qui automatise la création des composants de votre application métier .NET selon les meilleures pratiques des experts de la plate-forme Microsoft.

Soft Fluent

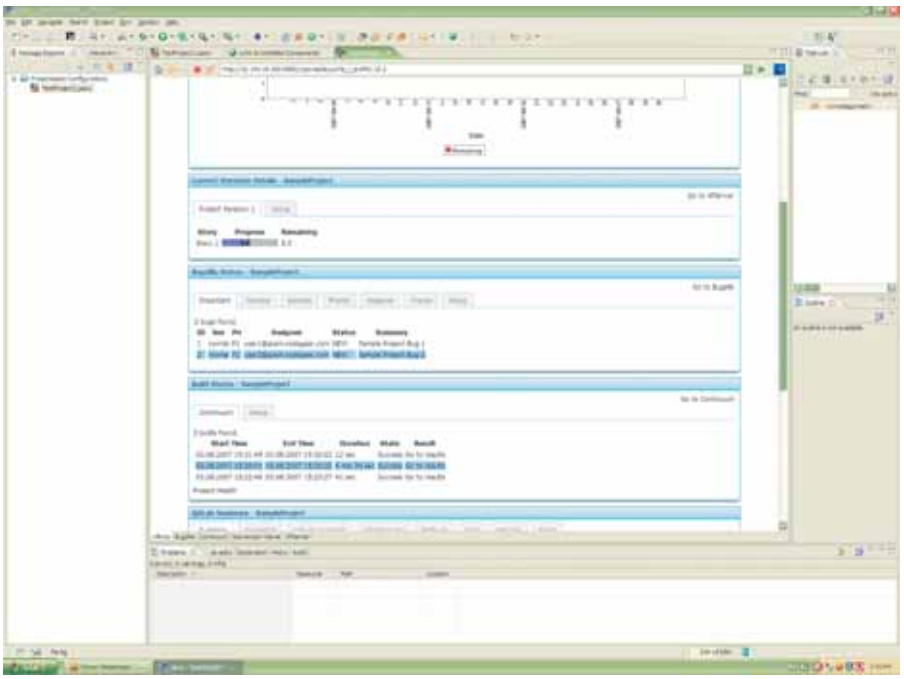
SoftFluent est une société spécialisée dans le développement sous plate-forme Microsoft. Elle compte déjà de nombreux clients prestigieux tels qu'ILOG, VCS Timeless, TF1 ou Microsoft France et Europe. Contactez-nous sur info@softfluent.com

Softfluent recrute !

Venez assouvir votre passion du développement logiciel au sein d'une équipe dynamique !

SoftFluent 46, rue Auguste Blanqui 94250 GENTILLY - 01 46 16 05 02

Microsoft | PROGRAMME IDÉES



le suivi du cycle de vie des objets et notamment de l'exécution du garbage collector. Cette fonctionnalité est très utile pour suivre l'évolution des threads et des composants. JGear Performance est capable d'analyser les applications JEE en détectant les goulots d'étranglements en terme de performance : JDBC, EJB, RMI ou encore JMS.

Afin d'analyser un composant ou une application au sein de l'environnement Eclipse, il est nécessaire d'avoir accès au code source. Cependant il est possible de réaliser des sessions en mode attaché. Dans ce mode, il est possible d'analyser des applications de tailles conséquentes comme un serveur d'applications, sans pour autant bénéficier de toutes les fonctionnalités d'analyse.

Afin d'analyser l'utilisation mémoire, le profiler affichera de nombreuses informations en temps réel : nombre d'instances, espace total, nombre d'objets libérés par le garbage collector pour chaque classe. A partir de ces données, vous pourrez remonter aux classes qui les ont instanciées et ainsi repérer jusqu'à la ligne de code défaillante. De nombreuses vues sont proposées : tableau, graphe ou encore arbre. L'interface est intuitive et permet de rapidement trouver l'information que l'on cherche. L'analyseur de thread est tout aussi complet. Il permet une analyse poussée des comportements des threads : charge cpu, objets instanciés, état, problèmes de synchronisation…

Un module d'audit permet d'aller encore plus loin dans la vérification des erreurs au niveau du code source par rapport au JDK de Sun. Il permet par exemple de détecter les transtypages répétés qui font perdre de la performance. Pour l'ensemble des erreurs reportées, l'outil est capable d'effectuer du refactoring qui corrigera le problème. Il est possible de créer manuellement des indicateurs qui permettront de valider ou non le code source, et donc les composants. Par exemple, si l'architecte du projet souhaite que le code source contienne 20% de commentaire au minimum et que le code ne contienne aucun attribut public, il pourra le faire et l'outil remontera alors les erreurs dans un rapport.

Team Client et Team Server

Team Server est l'outil qui offre la prise en charge de différents serveurs permettant de favoriser le travail d'équipe. La majorité des solutions proviennent du milieu open source : reporting de bug avec Bugzilla, gestion de configuration avec Subversion, ou encore serveur d'intégration avec Continuum. Ce plug-in offre essentiellement une visibilité sur une console d'administration qui centralise l'ensemble des outils.

La version cliente, JGear Team Client, est quant à elle destinée aux développeurs. Elle leur permet de suivre le projet sous différents angles : code source, rapports de bug, planning… L'ensemble de ces services est accessible via un portail intégré au sein d'Eclipse. Team Client s'interface avec l'outil Mylyn ou encore la solution StarTeam de Borland.

Conclusion

La solution de CodeGear est certainement l'une des meilleurs en ce qui concerne les projets de développements JEE. Elle apporte une réponse aux chefs de projets qui souhaitent rationaliser les développements en industrialisant l'ensemble des processus de conception et de réalisation. Un outil dédié aux tests aurait été apprécié. Ces outils favorisent fortement le développement par composants ainsi que l'utilisation des méthodes agiles.

Sites internet

www.codegear.com : Site officiel de l'éditeur
www.eclipse.org : Site communautaire du projet Eclipse

Loic Guillois

Plates-formes supportées

Attention, bien que les plug-in soient compatibles avec Eclipse et toutes ses variantes (JBuilder, BEA Workshop, MyEclipse…). Les systèmes d'exploitation officiellement supportés sont au nombre de deux : Windows XP et Windows Server. Pour l'instant, il n'est donc pas envisageable de les utiliser sous Linux, MacOS X ou encore Windows Vista.

Netbeans 6

Sur le CD ROM

La solution gratuite proposée par Sun est disponible dans sa version 6. Celle-ci apporte un lot de nouveautés considérable et parfaitement intégré avec les dernières technologies que sont les EJB 3 et JSF. Il propose différents modules permettant notamment de créer des projets SOA ou des applications d'entreprises. Netbeans dispose également d'un profiler qui n'a rien à envier à JGear.

Le rendez-vous incontournable des développeurs et professionnels de l'informatique !

Microsoft TechDays2008

L'innovation avance avec vous

Retrouvez toutes les innovations informatiques présentes et à venir réunies dans un événement unique et plus que jamais incontournable.

Édition Spéciale Lancement

Windows Server 2008

Microsoft SQL Server 2008

Microsoft Visual Studio 2008

11, 12, 13 février 2008

au Palais des Congrès de Paris

Circuit recrutement Microsoft et partenaires !

À vous de sélectionner votre parcours :

22 parcours techniques

220 et plus sessions sur tous les produits, toutes les solutions Microsoft

... Datacenter | Sécurité | Développement web | Décisionnel | Windows Vista | SharePoint...

msdn Microsoft TechNet

Inscrivez-vous, c'est gratuit !
www.microsoft.com/france/mstd08

Composants 4D v11 SQL

Concilier innovation et réduction des délais de développement

Seuls les projets de petite taille peuvent rester monolithiques. Lorsqu'un projet commence à devenir quelque peu complexe, il est généralement préférable de le structurer de façon modulaire en se basant sur la conception de composants spécifiques dits composants "métiers" ou/et sur l'intégration de composants existants prêts à l'emploi.

La programmation orientée composant (POC) consiste à utiliser une approche modulaire au niveau de l'architecture d'un projet informatique, ce qui permet d'assurer au logiciel une meilleure lisibilité et une meilleure maintenance. Les développeurs, au lieu de créer un exécutable monolithique, se servent de briques réutilisables. La POC est particulièrement pratique pour le travail en équipe et permet d'industrialiser la création de logiciels. 4D v11 SQL adopte tous les standards du marché et propose aux développeurs d'applications professionnelles une nouvelle génération de composants : faciles à développer et à installer, multi-plates-formes et utilisables immédiatement en monoposte, client/server et RIA.

Installation/Désinstallation par un simple Glisser/Déposer

Pour installer un composant dans une base, il suffit de le copier dans le dossier "Components" de la base, placé à côté du fichier de structure. Il est possible d'utiliser un raccourci (Windows) ou un alias (Mac OS). Pour le désinstaller, il suffit de le supprimer du dossier. Le dossier "Components" peut contenir tout fichier ou dossier personnalisé nécessaire au fonctionnement du composant (ressources xliff pour la localisation, images, ...).

Développer des composants 4D multi-plates-formes

- **Compilé**

Une base 4D exécutée en mode non-compilé peut utiliser indifféremment des composants non compilés ou compilés. Une base 4D exécutée en mode compilé utilisera des composants compilés.

- **Windows / Mac OS**

Un composant compilé ou non compilé développé sous Mac OS peut être installé dans un environnement Windows et inversement.

- **Stand alone / Client-Server / RIA**

Les composants utilisés pour des applications monopostes peuvent être distribués en Client/Server et être exécutés par une centaine de postes clients distants. Il suffit pour cela d'installer le composant sur le poste hébergeant 4D Server.

Un composant orienté Web 2.0 peut également être installé afin d'obtenir un client utilisable dans un navigateur avec la richesse d'AJAX.

Définir le niveau de granularité de votre composant

La granularité des composants se rapporte à l'éventail de fonctionnalités exposées par le composant :

- Les composants à granularité fine peuvent fournir une utilité limitée quant aux processus métier, mais serviront à la réutilisation des briques de développement les plus utilisées afin de personnaliser le mode développement.

Composants 4D/SQL

- Les composants à forte granularité peuvent être structurés de manière intelligente en respectant certaines règles d'architecture et de nommage pour satisfaire des besoins métiers spécifiques. Par exemple : composant comptabilité, composant eMailing, etc.

Pour régler votre niveau de granularité, 4D vous permet de définir le ou les objets qui seront partagés par vos composants et par l'application hôte. Ci-dessous, un exemple de partage des méthodes (Fig. A).

Ce paramétrage est effectué via les propriétés de la méthode (Fig. B).

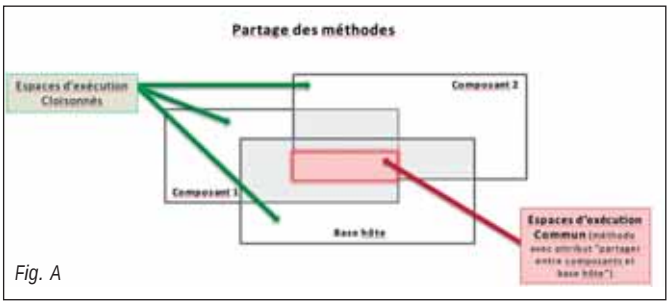


Fig. A

Les composants 4D peuvent être en langage 4D, SQL, ... voire même en C/C++

Fig. B

Les composants bénéficient de toutes les possibilités liées à une base 4D : IDE intégré, serveur SQL, serveur Web,...Les composants peuvent ainsi être écrits par des développeurs 4D ou des développeurs SQL.

Mais 4D donne également la possibilité à des développeurs C/C++ de créer des composants 4D et de travailler en bonne intelligence avec d'autres développeurs provenant de technologies différentes.

Pour simplifier leur travail, un Wizard et une API 4D ont été conçus spécialement pour eux. Ce type de composant, appelé plus communément Plug-in, s'installe aussi facilement que les composants.

■ **Jacques Lefevre**

Les bénéfices de l'utilisation des Composants 4D en 8 points :

1. Spécialisation : l'équipe de développement peut être divisée en sous-groupes, chacun se spécialisant dans le développement d'un composant 4D. L'application pourra être constituée d'un certain nombre de composants.

2. Sous-traitance : le développement d'un composant 4D peut être externalisé, à condition d'en avoir bien défini les spécifications au préalable.

3. Maintenance : la modification d'un composant 4D ne nécessite pas la recompilation du projet complet, ce qui rend ainsi la maintenance et l'évolution de l'application plus aisée.

4. Test unitaire : une telle architecture facilite l'écriture et l'exécution de tests unitaires plutôt que fonctionnels.

5. Faciliter le déploiement : dans le cas d'une mise à jour, d'un correctif de sécurité ou d'une implémentation mineure alors que le logiciel à déjà été livré au client, il suffit de fournir le composant 4D modifié sans devoir à nouveau livrer l'intégralité de l'application.

6. Productivité : réutiliser un composant 4D permet un gain de productivité non négligeable car le temps de développement est diminué, et ce d'autant plus si le composant est souvent réutilisé.

7. Adaptation : les composants permettent de personnaliser également le mode développement de 4D en y ajoutant ses propres outils.

8. Composants métiers : composants modernes orientés services avec la possibilité de créer des composants métiers.

Pour en savoir plus :
site : www.4D.fr - 01 40 87 92 00

Composants C/C++

Pervasive PSQL Summit v10
De Btrieve à Pervasive

L'éditeur Pervasive fêtait ses 25 ans en 2007 et vient de lancer la V10 "Summit" de sa base de données embarquée. Gilbert Van Cutsem, vice-président de l'entreprise, retrace l'histoire d'un éditeur atypique.

Dans la vallée des géants, Btrieve je me souviens ...

Dans le monde technologique actuel, qui dit Base de Données pense "Grand Compte" et les noms d'Oracle, Microsoft voire MySQL sont les premiers qui vous viennent à l'esprit. Le jour où ces géants se sont mis à cibler le monde de la petite et moyenne entreprise les éditions "lite" ou "express" ont vu le jour. Il est néanmoins important de noter qu'avant la naissance de SQL Server et avant même la conception de Windows il y avait Btrieve !

Dès le début de ses 25 années d'existence, Pervasive Software a toujours été l'exception à la règle. Mars 2007 marque le 25e anniversaire de Btrieve, tout comme la création du logiciel aujourd'hui connu sous le nom de PSQL. Ce qui jadis débuta comme développement et commercialisation d'un simple moteur (isam) de gestion d'enregistrements est devenu de nos jours une société cotée en bourse (nasdaq:pvsw), disposant d'un historique (vingt-six trimestres) de forte profitabilité, d'une position de trésorerie plus que confortable et d'une présence commerciale mondiale.

Une communauté de 5000 éditeurs

Bien que Btrieve - considéré comme un "vilain petit canard" par certains - ne se soit pas vraiment transformé en "cygne gracieux" comme ses créateurs l'avaient espéré, le logiciel s'est tout de même forgé une solide réputation parmi une communauté de quelques 5 000 éditeurs dans le monde entier. Il est devenu le cheval de trait - voire de bataille - de bon nombre d'éditeurs de logiciels pour la PME qui n'hésitent pas à utiliser les yeux fermés cet outil qui continue à leur apporter la robustesse tant recherchée.

En effet, répondre aux besoins des éditeurs a toujours été l'objectif principal de Pervasive en apportant une solution de base de données embarquée ne nécessitant aucune administration.

Nouvelle version

Pervasive PSQL Summit v10, la version la plus récente de ce logiciel phare, perpétue cet esprit et cette tradition au sein du marché "bases de données".

"Summit", simplifie aussi bien le développement que le déploiement et minimise les défis liés à l'adoption de nouvelles technologies. Ceci inclut les nouveaux standards tels que Windows Vista, Windows Server 2008 (Longhorn), l'architecture 64 bits,.NET 3.x (Orcas, LINQ) et bien sûr la technologie des processeurs multi-core.

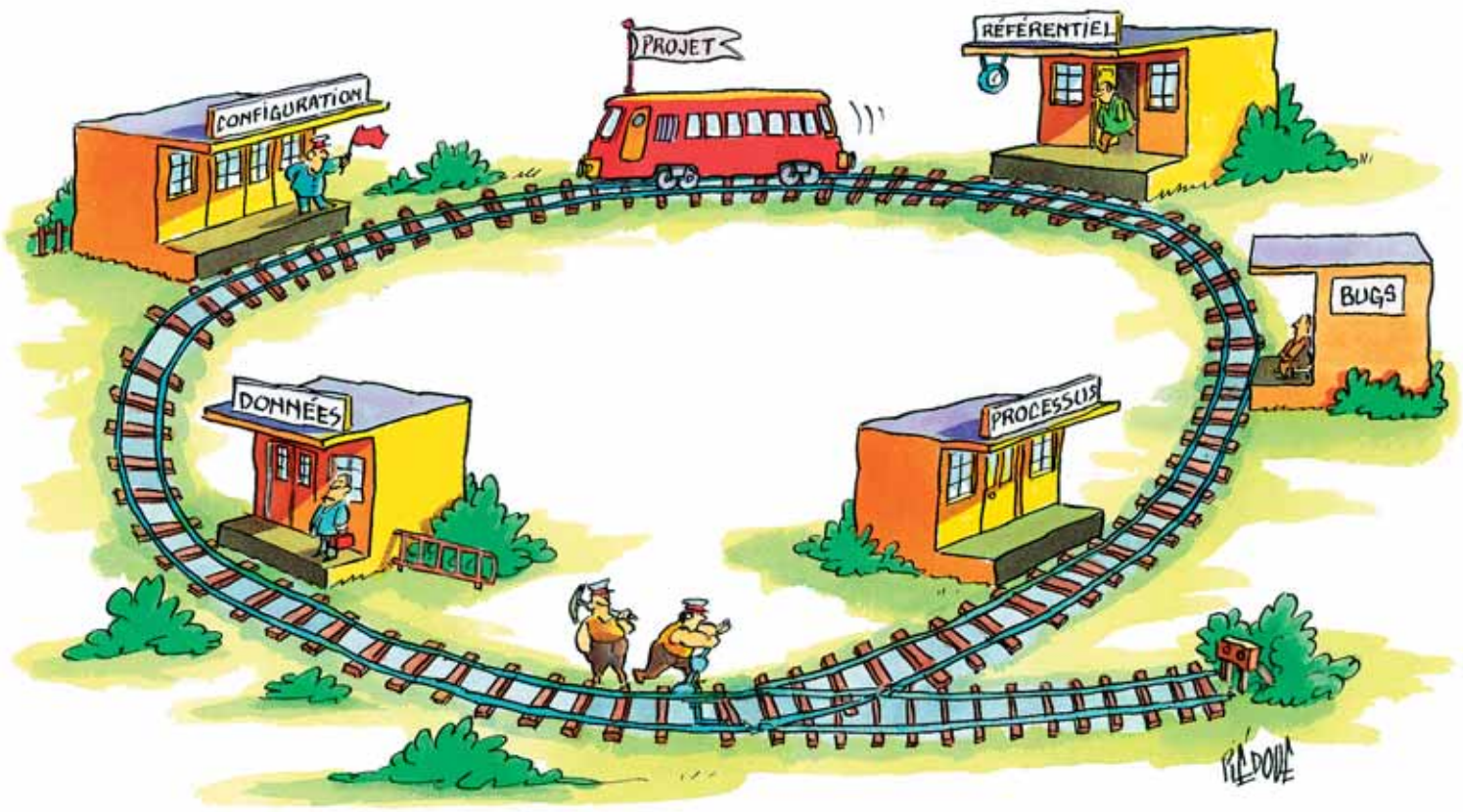
Pervasive PSQL Summit v10 continue - après maintes années de perfectionnement - d'être la base de données la mieux adaptée aux besoins de la PME (et de ses éditeurs) et aux préoccupations clés de ce marché :

- Performances applicatives reconnues comme hors pair
- Mise en œuvre à long terme ne nécessitant aucune administration.
- Auto-réglage et optimisation, installation simple (silencieuse, invisible).
- Mises à jour (Upgrade) faciles à implémenter, compatibilité totale et entière (amont/aval).

En continuant à délivrer un moteur de base de données embarqué fiable, Pervasive garantit la compatibilité totale avec le passé tout en focalisant sur l'innovation et l'avenir.

Gilbert Van Cutsem
Senior Vice President and
General Manager Database Division
www.pervasivetimemachine.com

Le cycle de vie logiciels



Aujourd'hui, avec la multiplication de l'offre logicielle et des domaines couverts, la gestion du cycle de vie devient de plus en plus complexe, surtout si on y ajoute les développements et équipes distribuées. Dans le meilleur des mondes, l'ALM (Application Lifecycle Management) est un cercle vertueux des exigences au déploiement, à la maintenance. Bien entendu, dans la réalité, les choses se passent un peu différemment.

ALM, le terme fait parfois peur, ou bien laisse totalement indifférent. Si son usage oblige à s'adapter, à modifier ses habitudes, à mieux structurer ses projets, son travail, l'ALM a aussi de bons côtés. Elle permet de mettre en place de bonnes pratiques, d'exprimer les exigences, de prévoir les tests en amont, etc. Mais elle exige d'être adoptée par tous les acteurs : utilisateurs, testeurs, développeurs, chefs de projets, directions informatiques.

Alors pourquoi la gestion du cycle de vie continue-t-elle à intimider ? Trop lourde ? Trop complexe ? Trop chère ? Difficile de nier ces réalités. Cependant, l'ALM d'aujourd'hui sait s'adapter à vos besoins, à vos équipes. Vous n'êtes pas obligés de tout acheter en même temps ou de tout mettre en œuvre. Car l'ALM débute avec une simple gestion de version comme CVS ou Subversion, une usine à build. L'autre souci est de savoir si mon outillage actuel peut continuer à fonctionner. Sur ce point, si des progrès ont été réalisés, l'interopérabilité reste un problème, même si les éditeurs ALM supportent de plus en plus l'existant comme Borland avec son Open ALM. D'autre part, le cycle se retrouve de plus en plus directement dans l'environnement de développement. On le voit avec Visual Studio Team System ou encore Eclipse avec le projet Eclipse Process Framework.

Aujourd'hui, la base, l'indispensable devrait-on dire, c'est au moins de mettre en plus une gestion des sources et de versions, avec du CVS, du Subversion, etc. C'est le cycle minimum. Pour Bruno de Combiens (Borland France), selon un sondage Borland, environ la moitié des entreprises auraient un outil de type CVS : *" on est encore structuré en silos, il n'y a pas une vision globale. Dans le développement, beaucoup de choses restent à faire, même si les fondations sont posées. Cela dépend beaucoup de la maturité des utilisateurs. "* Ce dossier cycle de vie et sa gestion n'est en réalité que la première partie d'une suite de dossiers abordant la qualité logicielle, la sécurité du code, l'usine à logiciels, les tests ou encore la méthodologie et l'architecture logicielle. Car n'oubliez pas que le cycle de vie n'est qu'un maillon d'un vaste ensemble. Ce cycle fédère en quelque sorte toutes les notions nommées plus haut. Dans ce numéro, nous reviendrons sur le cycle de vie : sa définition, son rôle, les outils et surtout les différents cycles que l'on rencontre aujourd'hui. On abordera aussi le cycle de vie vu par Microsoft avec la solution Team System.

■ **François Tonic**

Améliorer la gestion des projets

Mieux gérer les projets : c'est un voeu pieux mais comment faire ? La réponse est simple, en gestion de projets comme ailleurs, ne cherchons pas à réinventer la poudre. Des générations d'informaticiens, dans le monde entier, ont été et sont encore confrontés à cette épineuse question. Les esprits les plus prolifiques ont cherché (et cherchent encore !) à modéliser les étapes de la fabrication d'un "produit" informatique, et y ont associé des recommandations à suivre pour être le plus efficace possible.

Cette modélisation, désignée par le terme de "cycle de vie logiciel", correspond à des méthodes ou bonnes pratiques, pour réussir au mieux ses projets. Encore faut-il savoir s'orienter dans la jungle des acronymes : Cycles en V, en cascade, en spirale, RAD, RUP, UP, SCRUM, XP, DSDM, méthodes agiles… Des termes obscurs qui désignent finalement des méthodes pragmatiques et applicables si la volonté est là. Le choix doit être réalisé en fonction de l'environnement technique, organisationnel, humain, et des phases d'intervention sur le projet. Une fois le modèle déterminé, encore faut-il passer de la théorie à la pratique. Savoir adapter un modèle sans le dénaturer : tout un programme !

Le CVL qu'est ce que c'est ?

Tout d'abord quelques rappels : le cycle de vie d'un logiciel est la description d'un processus modélisant les étapes de l'élaboration d'un "produit" informatique, ceci depuis l'expression du besoin jusqu'à sa fin de vie. L'objectif d'un tel découpage est de permettre :

- la définition de jalons intermédiaires permettant de valider la conformité du développement par rapport aux besoins exprimés,
- l'adéquation des méthodes mises en œuvre dans le processus de développement.

Le but de ce découpage est de maîtriser au mieux les risques, les délais et les coûts et obtenir une qualité conforme aux exigences.

Le CVL définit notamment le nom de chaque étape (conception, développements…), les pré-requis, les résultats attendus, les livrables à produire. Cette modélisation a une double vocation : tout d'abord la mise en place d'une terminologie précise permet aux différents interlocuteurs d'un projet de se comprendre. Elle rend également possible la comparaison entre différents projets, et donc l'évaluation d'un projet au regard de références.

La modélisation en cycle de vie permet également d'avoir une vue d'ensemble du projet, et donc de prévoir en amont du cycle la préparation des étapes en aval.

Communiquer, évaluer, synthétiser : voilà déjà pourquoi la référence au cycle de vie est essentielle pour un projet. Voyons maintenant quels sont les modèles les plus répandus et leurs principes sous-jacents.

Fig.1 - Waterfall model (ou cycle en cascade)

Cycles en cascade ou itératif : les grandes familles

Du cycle en cascade au cycle V : le B-A BA

Le cycle en cascade, qui est le plus simple et le premier des CVL historiquement, part du principe que toute étape doit être achevée avant de passer à l'étape suivante (dans le bâtiment, il n'est pas possible de construire les murs avant les fondations). Cela signifie, typiquement, que les spécifications doivent être validées avant le démarrage des développements (Fig.1).

Ce modèle n'est pas spécifique à l'industrie informatique et repose sur l'idée que les conséquences d'une modification en amont du cycle ont un impact majeur sur les coûts en aval. En cas de modification, il y a retour aux étapes précédentes, voire au tout début du cycle.

Le processus de développement utilisant un cycle en cascade exécute des phases qui ont pour avantages :

- de produire des livrables définis au préalable
- de se terminer à une date précise
- de ne se terminer que lorsque les livrables sont jugés satisfaisants lors d'une étape de validation

Ces éléments permettent d'avoir une bonne maîtrise du projet mais imposent en retour des contraintes relativement fortes.

Si on réfléchit aux spécificités de l'industrie informatique, l'une des grandes difficultés est de produire, au final, un livrable qui est conforme au besoin des utilisateurs. Or sur une grande majorité de projets, nos clients ont beaucoup de mal à définir leurs besoins avec la précision exigée par l'informatique. D'autant qu'un logiciel est plus abstrait pour les utilisateurs que les produits "matériels" fabriqués par l'industrie. Le cycle en cascade ne donne pas le moyen d'éviter cet " effet tunnel " qui fait qu'en fin de projet, les utilisateurs peuvent avoir le sentiment de recevoir un produit somme toute assez éloigné de ce qu'ils avaient commandé. Afin d'améliorer cet état de chose, le cycle en cascade a été décliné dès les années 80 sous la forme du cycle en V, qui est aujourd'hui un standard de l'industrie informatique (Fig.2).

Afin de réduire l'effet tunnel, le cycle V prévoit qu'à chaque étape en amont des développements (branche analyse) corresponde une phase de test de même niveau et qui peut (et doit) être préparée en même temps : à la spécification des besoins correspond la recette, à la conception générale correspond la recette d'intégration… C'est-à-dire qu'en face de tout livrable de conception doit correspondre un dossier de validation.

Ce cycle oblige donc à tenir compte des cas d'utilisation (et des exigences) du produit fini dès la conception, et ceci en ayant une approche par les résultats attendus. Ce système a plusieurs avantages :

- Il permet la parallélisation des chantiers de conception et de recette qui pourront éventuellement être menés par des équipes indépendantes.
- Il permet lors de la préparation de la recette d'alimenter l'analyse ce qui oblige les

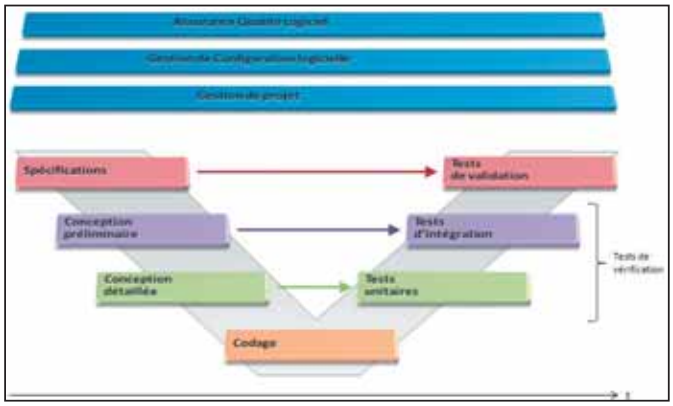


Fig.2 - Cycle en V

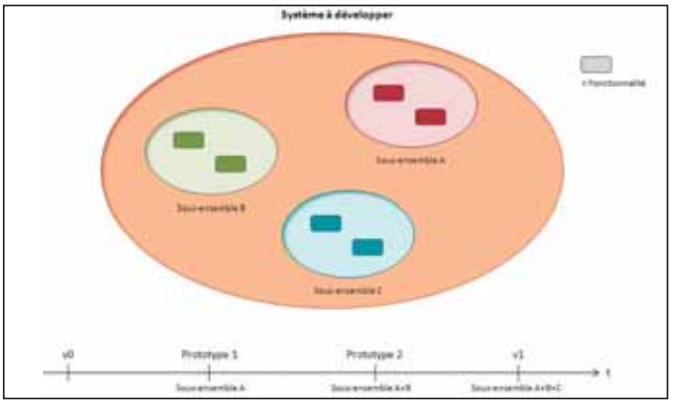


Fig.3 - Cycle incrémental

concepteurs et le client à bien réfléchir à tous les aspects de la demande et à vérifier qu'ils se sont bien compris.

- Il est un excellent support à la formalisation des relations entre le client et l'équipe de développement dans le sens où il démontre au client que les demandes en amont du cycle l'engagent fortement sur toute la durée de celui-ci.
- Enfin, il est supporté par de nombreuses normes et méthodes : ISO 9001, MIL-STD498, GAM T17, Do 178 B ; SADT, SART, OMT.

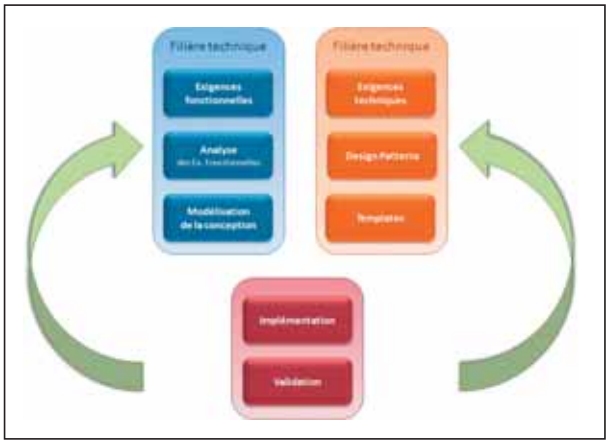


Fig.4 - Cycle itératif en Y

Si l'on retient le cycle en V dans sa configuration la plus simple, en 3 étapes (conception/développement/qualification), les points fondamentaux sont les suivants :

- Le livrable issu de la conception est une spécification qui doit être validée par le client
- Les développements commencent après la validation des spécifications
- La préparation des tests de qualification doit être faite en regard de la conception (via les spécifications) et non par rapport aux développements (ce qui les différencie des tests unitaires). Ils doivent faire l'objet d'un cahier de test formalisé et validé par le client.

Ces principes de base, comme nous allons le voir, sont repris par l'ensemble des autres CVL.

Les cycles itératifs : en route vers l'agilité

Malgré les améliorations importantes qu'il apporte, le cycle en V est critiqué depuis plusieurs années. En effet il est fondé sur le principe que l'ensemble de la solution doit être conçu avant le démarrage des développements, et ceci en essayant de limiter au maximum les retours arrière. Dans l'optique de respecter les jalons définis dans le planning projet, il ne sera pas possible d'intégrer des modifications du cahier de charges pendant le cycle. En conséquence, les demandes d'évolution par rapport à l'expression de besoin initiale pourront à la rigueur être intégrées dans un lot ultérieur, ou devront faire l'objet d'une nouvelle version dans la plupart des cas.

Or, dans un contexte économique toujours plus concurrentiel, les directions métiers doivent s'adapter en permanence, et les outils informatiques fournis par les DSI y prennent une importance croissante (Fig.3) et (Fig.4).

En conséquence la capacité à s'adapter et la réactivité des équipes techniques deviennent plus importantes que le strict respect des engagements et des délais. Il faut donc des méthodes qui permettent d'être plus réactif, plus flexible, plus efficace, en un mot, plus **agile.**

Pour atteindre ce but, le premier pas est de changer d'approche par rapport aux méthodes classiques. Le manifeste agile, mis au point en 2001 par 17 personnalités du monde de l'informatique, établit les valeurs fondatrices de l'agilité :

- individus et interactions plutôt que processus et outils
- développement logiciel plutôt que documentation exhaustive
- collaboration avec le client plutôt que négociation contractuelle
- ouverture au changement plutôt que suivi d'un plan rigide

Les premières méthodes pouvant être considérées comme agiles, telles que RAD (Rapid Application Development), définie au début des années 80 par James Martin, ou UP (Unified process), qui intègre les principes d'UML, ont précédé la signature du manifeste. Les méthodes agiles ont pour point commun de ne plus considérer une demande de changement comme une perturbation mais comme élément naturel et normal du déroulement du projet. Le cycle de vie doit donc être conçu pour être capable d'intégrer des modifications nécessaires tout au long de la construction du projet. Le principe est le suivant : au lieu de concevoir complètement l'ensemble de la solution avant de l'implémenter, le projet est construit par itérations successives, fonctionnalité par fonctionnalité. C'est ce qu'on appelle parfois le prototypage.

Le cycle RAD est un cycle de développement court (120 jours maximum), composé des 5 phases suivantes :

- **Initialisation :** préparation de l'organisation et communication
- **Cadrage :** analyse et expression des exigences
- **Design :** conception et modélisation
- **Construction :** réalisation, prototypage
- **Finalisation :** recette et déploiement

Les 3 premières phases sont séquentielles. Au terme du design, l'architecture fonctionnelle et technique générale ayant été définie, il est possible de réaliser un découpage du projet en terme de fonctionnalités (services), et de niveau au sein des ces fonctionnalités. Ce découpage fonctionnel est étudié en fonction

des découpages techniques correspondants. Le but est d'identifier des " briques " qui doivent permettre la réalisation et la mise en service du projet module par module. Une fois la priorisation des services décidée commence la phase de construction, qui est une phase itérative. Module par module, on applique le cycle en V : conception détaillée, développement, recette. Afin d'être le plus efficace possible, et conformément aux valeurs de l'agilité, des utilisateurs sont impliqués dans le développement (sur les tests notamment), ce qui évite les lourds process contractuels et garantit la conformité de la production aux besoins métiers. A la fin d'une itération de la phase de construction, un comité évalue la production : si le prototype est adapté au besoin alors il sera aussitôt mis en production. Dans le cas contraire, il pourra faire l'objet d'une nouvelle itération, ceci jusqu'à la finalisation du projet. RAD est qualifié de cycle semi-itératif dans le sens ou les itérations ne concernent que la phase de construction.

Les méthodes itératives ont plusieurs avantages : elles permettent d'obtenir la livraison au plus tôt des fonctionnalités essentielles. Elles donnent une vision précise dès le début du projet aux utilisateurs, qui pourront alors enrichir et préciser leur besoin. Elles facilitent la phase de recette dans le sens où les tests sont réalisés par le métier de manière continue et non en " one shot " lors de la livraison.

Parmi les méthodes itératives et incrémentales les plus connues on peut citer RUP, qui est une instanciation d'UP par Rational (société rachetée par IBM), adaptée de manière à fournir une méthode " clef en main ". Rappelons qu'UP a pour particularité d'intégrer les préceptes d'UML. SCRUM s'inspire des valeurs du rugby et propose une méthode focalisée sur l'esprit d'équipe. DSDM (Dynamic Software Development Method), étend la méthode RAD en comblant certaines lacunes.

Le dernier né : eXtrem Programming

Le dernier courant, apparu au début des années 2000 est l'eXtrem Programming. On peut considérer qu'il a donné ses lettres de noblesse à l'agilité en apportant une approche à la fois novatrice et pragmatique. XP a pour originalité de recueillir les best practices de l'industrie informatique et de les pousser à l'extrême. On retiendra parmi quelques exemples marquants, la programmation en binôme, la conception et les tests continus.

En marge de ces approches classiques il existe également des cycles qui modélisent spécifiquement certains types de projets, tels que le cycle de vie d'un projet décisionnel.

Comment choisir son modèle ?

Paramètres à prendre en compte et recommandations

Du cycle en cascade à l'eXtrem Programming, il existe forcément une ou plusieurs méthodes adaptées à un projet. Les critères du choix pour la sélection d'un CVL se jouent sur différents plans : fonctionnel, technique, contractuel, organisationnel et culturel. Du point de vue **fonctionnel,** est-il possible de définir complètement le besoin dès le commencement du projet ? C'est typiquement le cas des projets réglementaires. A l'opposé, un portail web peut être plus facile à définir à partir d'une maquette.

Techniquement, le projet utilise-t il des technologies plutôt maîtrisées ou novatrices ? Quel est le niveau de risque ? Lorsqu'on utilise une nouvelle technologie, tenter de spécifier l'intégralité du projet avant de développer peut s'avérer dangereux, alors que les itérations peuvent permettre de corriger le tir et de tirer le meilleur parti de la technologie.

En revanche, le cycle en V, qui permet une meilleure maîtrise du planning et des dates de livraisons est adapté à la gestion d'un forfait, alors qu'une convention de service sera plus appropriée en utilisant un cycle itératif. De manière générale, les valeurs de l'agilité sont incompatibles avec un contexte de relation très **contractuelle.**

Du point de vue **organisationnel,** il faudra également vérifier la compatibilité de la méthode et des normes qualité de l'entreprise. Par ailleurs, les méthodes agiles nécessitent une implication continue des utilisateurs tout au long du projet, alors que les cycles en cascade demandent un effort de recette cumulé sur la fin du projet : cet effort est parfois rebutant, au point de faire échouer certains déploiements. Pourtant il n'est pas toujours facile d'obtenir une implication des utilisateurs sur la durée. Nous arrivons donc sur la question de la **culture :** les équipes ne sont pas toujours prêtes à modifier leurs habitudes pour adopter de nouveaux modes de fonctionnement. Il sera inutile d'expérimenter l'agilité s'il n'y a pas une réelle motivation de la part de tous les intervenants. Il est primordial de garder à l'esprit qu'au-delà des méthodes, ce sont les valeurs qui caractérisent l'agilité par rapport aux CVL traditionnels. **Tous les intervenants doivent partager les mêmes valeurs pour que le projet fonctionne.** Une fois le choix effectué entre les grandes familles de cycle (Cascades ou méthodes agile), le choix précis d'une méthode pourra être fait en fonction :

- de l'existant dans l'entreprise, au niveau des méthodes et des normes,
- du périmètre couvert par les différentes méthodes,
- de l'environnement humain : profil et motivation des équipes !

Mise en œuvre : les pièges à éviter

L'art de l'équilibre.

Le premier défi du chef de projet sera de jauger son environnement avec soin afin de sélectionner une méthode adaptée. Un bon choix facilitera le travail mais toutes les méthodes précédemment citées peuvent quasiment être mises en œuvre sur un projet. La réussite ou l'échec dépendra moins de la méthode retenue que de la manière de l'appliquer.

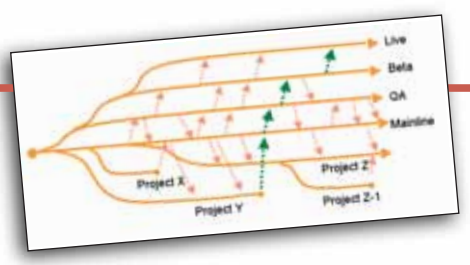
Comme tout modèle, une adaptation est parfois nécessaire pour passer de la théorie à la pratique. Prenons un exemple : une difficulté classique rencontrée par les chefs de projets sur le cycle en V est d'obtenir la validation des spécifications avant le démarrage des développements. En régie comme au forfait, les contraintes de planning induisent souvent le client à mettre une très forte pression pour contourner cette règle.

Y céder est dangereux car cela peut entraîner l'augmentation de la charge globale de développement, une moindre maîtrise du planning, la diminution de la qualité du code due aux modifications successives, la démotivation des ressources... A l'inverse, refuser tout assouplissement peut rendre impossible l'atteinte des objectifs du projet dans les délais.

Autre exemple, pour un cycle itératif, si la construction par prototypages peut paraître sécurisante, il faut qu'elle aboutisse et que le nombre de cycles reste limité ! Pour la réussite de ce type de cycle, la participation des utilisateurs aux phases de développement et de test est indispensable même si elle peut être difficile à obtenir.

L'ensemble des méthodes compose donc un panel complet de bons outils, au chef de projet d'en être le bon ouvrier !

Sarah Pol
Référent en Gestion de Projets Décisionnels
chez EDIS Consulting
http://www.edis-consulting.com



La gestion du code source et des versions selon Perforce

Sur le CD ROM

Le cycle de vie d'un logiciel consiste à suivre le code source dans les différentes étapes de sa vie, du développement au test, en passant par le versioning, la réutilisation et la désaffectation. Nous allons voir comment Perforce prend en charge ce cycle et définit un modèle de référence. Dans ce modèle, le tronc du code source a un objectif unique : la ligne directrice du développement. Autrement dit, son niveau de promotion (ou en cascade) sera toujours considéré comme étant le niveau de "développement".

La première fois que les fichiers sont promus, ils sont ramifiés depuis le tronc dans une branche. Les révisions ultérieures qui ne sont pas promues sont fusionnées depuis le tronc dans la branche. Les différents niveaux de promotion étant en fait des branches distinctes, ils peuvent évoluer distinctement. Le mécanisme de ramification, appelé Inter-File Branching, crée des branches en copiant des fichiers et en leur attribuant un nouveau nom. Chaque fichier possède un processus de révision indépendant. Aussi, lorsqu'un fichier du tronc est ramifié, il pourrait s'agir de //depot/main/db/ dbhdr.h qui est ramifié dans //depot/ bugfix/2.1/db/dbhdr.h. Il est possible de ramifier des fichiers distincts, des groupes de fichiers ou l'ensemble du "dépôt". Toutefois, Perforce ne prend pas en charge la ramification ad hoc lorsqu'un fichier peut être extrait d'une branche dans une autre. L'une des raisons tient à ce que c'est beaucoup plus simple à comprendre ; la ramification est toujours un acte explicite.

Le modèle de cycle de vie de référence Perforce

Il est possible d'imiter le modèle de promotion (ou modèle en cascade) « standard ». Il suffit d'utiliser la ramification pour promouvoir des groupes de fichiers de //depot/development dans //depot/qa, puis de //depot/release, etc. Mais cela peut impliquer que la ligne de code de la "version" soit utilisée à plusieurs reprises pour les versions multiples, intégrant à chaque fois les changements à partir de la ligne de code "AQ" (Assurance Qualité), puis de la ligne de code "développement". Un critère plus précis impose la ramification d'une ligne de code si son utilisation prévue est incompatible avec sa politique d'utilisation. Par exemple, la politique de la ligne de développement (stipulant qu'elle peut accepter une nouvelle fonctionnalité) est généralement incompatible avec la création de versions stables ; ainsi, les lignes de version doivent être ramifiées depuis les lignes de développement. En outre, la politique des anciennes versions autorise généralement les modifications pour les correctifs d'urgence uniquement. Ce qui n'est pas cohérent avec la politique d'une version plus récente, qui autorise une correction des bogues plus fréquente. C'est pourquoi nous proposons un modèle qui tire avantage de la ramification performante de Perforce en créant une branche pour chaque ligne de code ayant une politique unique.

Quelques fonctions de gestion de code

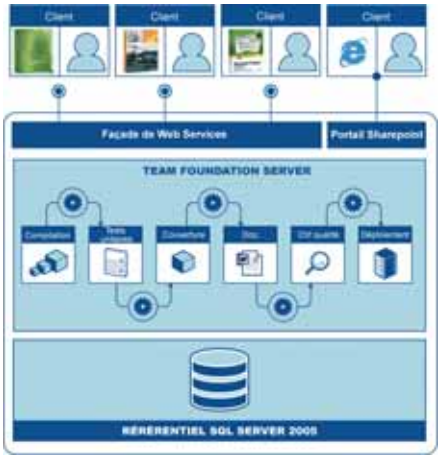
- **Ligne directrice :** le tronc est la ligne de développement primaire. Elle est appelée ligne "directrice", ce qui signifie que les fichiers se trouvent sous //depot/main ; elle représente les bijoux de famille de l'entreprise. Tandis qu'un code plus récent peut être développé dans des branches de développement spéciales, la ligne directrice est la destination ultime pour tous les développements logiciels. Les correctifs et les changements de portage effectués dans les lignes de la version doivent l'être en retour dans la ligne directrice.
- **Lignes de maintenance :** au cours du projet, des versions (release) doivent être réalisées. La ligne directrice est ramifiée au point de divergence entre la politique de la ligne directrice et les besoins de la version, à savoir, lorsqu'une nouvelle fonctionnalité n'est plus souhaitée dans la version.
- **Lignes de version :** au cours du test, la ligne de maintenance sera détectée comme étant prête pour passer au niveau version. La ligne de maintenance est ramifiée au point de divergence entre la politique de la ligne de maintenance et les besoins d'une version, à savoir, lorsque seuls des correctifs d'urgence sont autorisés.
- **Commentaires :** une caractéristique importante du modèle Perforce est que la pertinence d'un fichier est encodée dans son nom : //depot/release/3.5/01/db/dbhdr.h est un fichier qui fait partie d'une version, et qui n'est clairement pas l'emplacement pour des améliorations diverses. De même, //depot/main /db/dbhdr.h n'est clairement pas l'emplacement où des produits pouvant être expédiés sont construits. De plus, l'ensemble de la version 3.5/01 se trouve sous //depot/release/3.5/01 et nulle par ailleurs dans le "depot". Dans ce modèle, une ligne de code peut donner naissance à plusieurs lignes "au niveau de promotion suivant". Autrement dit, la ligne directrice peut créer plusieurs lignes de maintenance qui peuvent être actives en même temps.
- **Variations :** le développement ne doit pas directement se produire sur la ligne directrice. Les développeurs individuels ou les groupes peuvent ramifier la ligne directrice en lignes de projet nommées //depot/dev/projet, où "projet" est le nom de l'utilisateur ou du groupe. Dans ce cas, il n'est probablement pas nécessaire de ramifier l'ensemble de la ligne directrice, mais seulement les zones susceptibles d'être modifiées. Par exemple, le groupe de développement de la base de données peut ramifier les fichiers sous //depot/main/db dans //depot/dev/ fast-db/db pour un projet d'amélioration des performances. Tous les projets n'ont pas besoin à la fois de lignes de maintenance et de lignes de version. Une version peut parfois être réalisée directement à partir des lignes de maintenance.

Christopher Seiwald
D'après le livre blanc de cet auteur

RETOUR D'EXPÉRIENCE

Team System : les cycles de vie du logiciel

L'expression « cycle de vie du logiciel » contient l'idée même de vie du logiciel. Non pas que les logiciels soient capables de se reproduire ou de s'auto-organiser, mais l'observation de leur évolution dans le temps nous apprend qu'ils traversent différents contextes en s'y adaptant en permanence.



Référentiel unique de Team System.

A travers l'histoire revisitée d'une start-up (qui a réellement existé), nous verrons quels sont ces différents contextes et cycles de vie du logiciel, comment les logiciels s'adaptent à leurs contraintes, et en quoi Team System peut nous aider à faire vivre un logiciel.

L'idée

Accoudé au comptoir, Loïc s'exclame : « *Tu comprends, moi je trouve que les sites d'emplois ne sont pas du tout pertinents. Pour trouver un job de contrôleur de gestion, tu dois lancer une recherche sur la fonction Finance, et après tu te noies dans des centaines d'offres d'actuaire, de gestionnaire middle office, de trader ForEx, … voire de directeur financier ! Franchement, y'a un truc à faire, non ?!* ».

Loïc développe son idée et Christophe suit attentivement en acquiescant de la tête. Saeed et moi écoutons la conversation avec intérêt. Son idée consiste tout simplement à offrir un service de recherche d'emplois pertinent car spécialisé dans les métiers de la banque, finance et assurance (« BFA »). Comment est-il possible que personne n'y ait pensé auparavant ? Et c'est comme ça que toute l'histoire a commencé !

De l'idée à la v. 1.0

Un business plan et une SARL plus tard, quelques stagiaires, Saeed et moi, nous nous mettons à développer le site. Le plus important est de sortir la v1 le plus vite possible pour prendre le maximum d'avance sur de futurs concurrents…

Le but du jeu : écrire vite une version simple et opérationnelle implémentant un moteur de recherche multi critères spécialisé en BFA, un processus d'inscription de candidats et de recruteurs ainsi qu'un back office d'administration de données en ASP.NET, C#, et SQL Server.

Côté build, je propose un Team Foundation Server auquel tous les développeurs de l'équipe se connectent facilement dans Visual Studio pour coder. Team Source contient notre référentiel de sources, qui est unique, consistant et partagé. Sympa : les conflits entre codes de développeurs se résolvent lors du check-in, graphiquement et en 3 clics.

Je paramètre Team Build pour que chaque check-in déclenche la compilation de l'application, la vérification de nos tests unitaires, et nous notifie immédiatement par email en cas de régression.

Team System nous calcule aussi les taux de couverture et nous vérifions régulièrement que les couches centrales sont suffisamment testées. J'étends Team Build pour que cette intégration en continu déploie le site à chaud, de sorte que la dernière version du site soit visible à tout moment par Christophe et Loïc, nos maîtres d'ouvrage et associés.

Côté méthodologie de développement, on recherche la vélocité à tout prix.

Primo, pas de méthodologie outillée de développement : Team System le permet et nous n'en ressentons pas le besoin à 2 dans le même bureau, avec Loïc et Christophe en permanence à nos côtés.

Ensuite, on n'investit pas sur des spécifications sous Word, mais sur des tests unitaires. Loïc et Christophe travaillent à nos côté pour définir des données de tests univoques et assez détaillées : état initial, entrée, sortie à vérifier.

En voici un échantillon : rechercher 'contrôleur gestion' retourne une liste non vide de résultats, et contient les offres n° 123 et 456. Idem pour 'contrôleur de gestion'. Ces données sont prises en entrée de l'écriture de nos tests unitaires. Une fois un test unitaire écrit, on écrit le code applicatif pour faire passer ce test. On pense et construit l'application par les tests.

Le brouillard fonctionnel se dissipe chaque jour un peu plus dans la tête de Loïc et Christophe au fur et à mesure que nous avançons… sans avoir la moindre idée précise de ce que sera le site, ne serait-ce que le mois prochain !

Après quelques semaines d'une course effrénée, le 20 septembre, le premier site emploi dédié aux métiers de la banque – finance – assurance naît enfin !

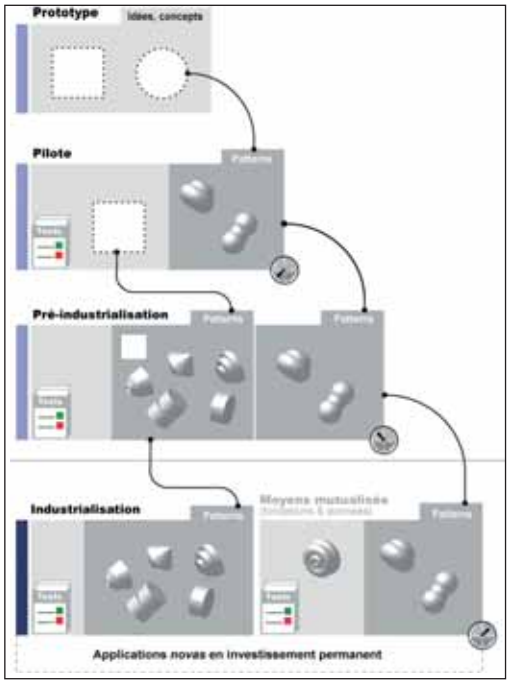
Au bout de quelques semaines d'existence, on propose près de 200 offres mensuelles et les tags Xiti identifient 10 000 visiteurs uniques mensuels.

Côté run : un serveur mutualisé, un accès en FTP et SQL Enterprise Manager font qu'à l'arrivée, du besoin exprimé au run, c'est l'autoroute ! Avec la production au bout des doigts, je peux écrire des tests, coder, vérifier que mes tests passent, puis déployer autant de fois que nécessaire. Et jusqu'à plusieurs versions par jour s'il le faut.

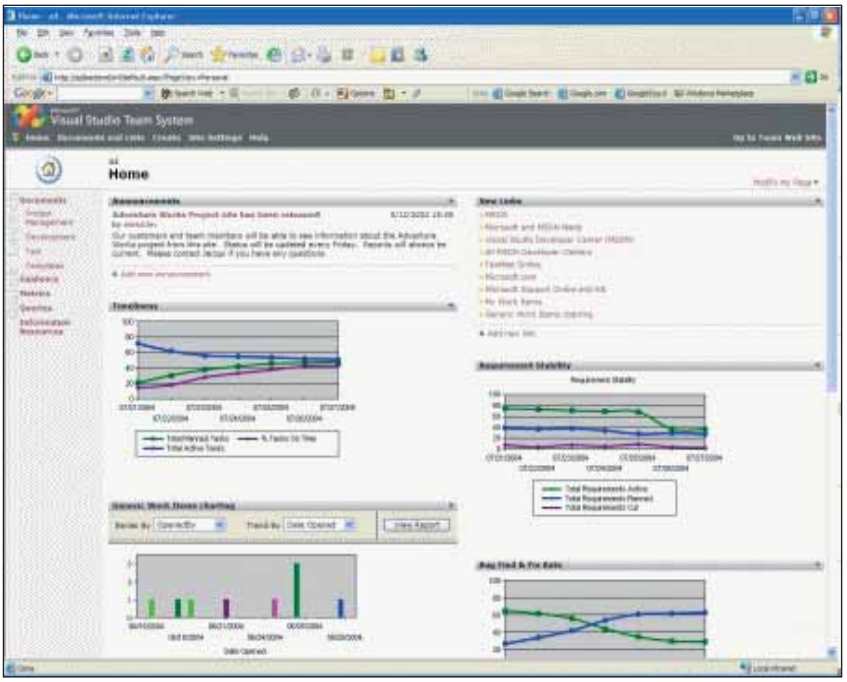
Trois ans plus tard…

Aujourd'hui, c'est l'anniversaire du site : trois ans jour pour jour ! Un verre de champagne à la main, je m'éloigne du reste de l'équipe pour me poser un instant. Inconsciemment, je mesure le chemin parcouru et je réalise que le site a énormément évolué depuis 3 ans. Et nous aussi !

Naïve à ses débuts, l'application ne l'est pas restée bien longtemps : le moteur de recherche corrige les fautes d'orthographe les plus fréquentes, est utilisé en marque blanche



Les cycles de vie du logiciel.



Rapports Team System.

chez une multitude de partenaires, on vend une CVthèque aux recruteurs, un module d'import/export des offres d'emplois, transactionnel et multi canal est devenu indispensable, …

Avec toutes ces évolutions métier, nous sommes devenus les rois du TDD, du refactoring et des tests unitaires. Team Test et Visual Studio sont nos meilleurs alliés pour vérifier la non régression de notre application de manière automatisée et productive. Sans cela, nous croulerions sous les bugs.

Côté méthodologie de développement, quelques changements importants ces 3 dernières années. Avec 5 développeurs dans l'équipe, la feuille Excel partagée en réseau nous conduisait systématiquement à des problèmes de concurrence d'accès. Plus possible… Et puis, avec le temps, les demandes de développement sont devenues plus gourmandes en charge et le développement au jour le jour devint inapproprié. Chose nouvelle, nous développons donc maintenant sur la base d'itérations fixes de deux semaines avec le gestionnaire de demandes de développement du processus MSF for Agile de Team System. Son workflow des demandes a même été adapté avec une étape supplémentaire de planification en début d'itération.

Depuis Visual Studio, les développeurs peuvent consulter, ouvrir ou fermer les demandes de développement qui leur sont assignées. Ils peuvent aussi demander des précisions si un scénario métier est mal compris sous la forme de jeu de données de test complémentaires. Ce lien est fluide puisque Team System est fondé sur un pattern de référentiel unique (accessible depuis Visual Studio, Excel et Project) : une base SQL contient tous les éléments pour développer en équipe sur un projet : code source, demandes de développement, tests unitaires, documents projets par exemple.

Au moment de la remontée de code source, les développeurs ont appris à fermer la demande correspondante ou à en faire progresser le pourcentage d'avancement.

Loïc et Christophe peuvent suivre de temps en temps l'avancée des développements en accédant au référentiel unique à l'aide d'Excel qui présente la liste des demandes et leur statut actuel : planifié, ouvert, en cours de développement, fermé.

Une démonstration de la dernière version du site est toujours organisée pour Loïc et Christophe en fin d'itération.

Chaque artefact de build qui est déployé en production est aussitôt tagué. Avec ce tag, on peut aller requêter Team System et retrouver facilement la liste des demandes implémentées ainsi que l'image exacte du code source correspondant, et les tests associés. Tout se tient !

Côté run, nous sommes maintenant sur deux serveurs dédiés, après avoir souffert de lenteurs en environnement mutualisé. Nous pouvons déployer une nouvelle version en production à chaque fin d'itération. Ce sont Loïc et Christophe qui prennent cette décision. Dans les faits, on déploie une nouvelle version tous les mois.

Au bout de 3 années d'existence, on propose près de 1 000 offres mensuelles et Xiti compte 50 000 visiteurs uniques mensuels et un taux de pertinence moyen estimé à 90 %.

Cinq ans plus tard…

Que de changements ces cinq dernières années ! Fort de notre succès, notre petite entreprise s'est fait racheter par la World JobBoard Company (WJB) ! Me voilà donc petit mercenaire dans un grand groupe international spécialisé dans le recrutement.

Le site ne cesse d'évoluer fonctionnellement, et est même devenu une référence en la matière. Il a même servi à produire d'autres sites spécialisés en marketing, vente, sport, distribution, … quasiment en faisant un copier/coller. Aujourd'hui, c'est une véritable plate-forme mutualisée de sites emplois spécialisés qui s'est construite.

Et nos tests sont toujours là. Toujours plus nombreux et toujours verts. Ils ne demandent qu'à être refactorés au besoin, pour être encore plus nombreux, et encore plus verts J

Team System est toujours là, lui aussi, mais il a sacrément évolué. Voici comment nous nous en servons. La World JobBoard Company cible la certification CMMi niveau 3 pour l'année prochaine. Nous utilisons donc le processus MSF for CMMi de Team System.

Les documents projet conformes à CMMi sont stockés dans le référentiel unique de Team System. On peut y accéder directement depuis Word ou Excel. Grâce au pattern de référentiel unique de Team System, ces documents sont reliés aux demandes de développements, qui sont elles-mêmes reliées au code source et aux tests. On peut naviguer entre ces éléments.

Le processus de développement est défini et suivi à la lettre. Cela ne nous empêche absolument pas de continuer à travailler de manière itérative et pilotée par les tests, mais c'est simplement plus rigoureux et un peu plus documenté.

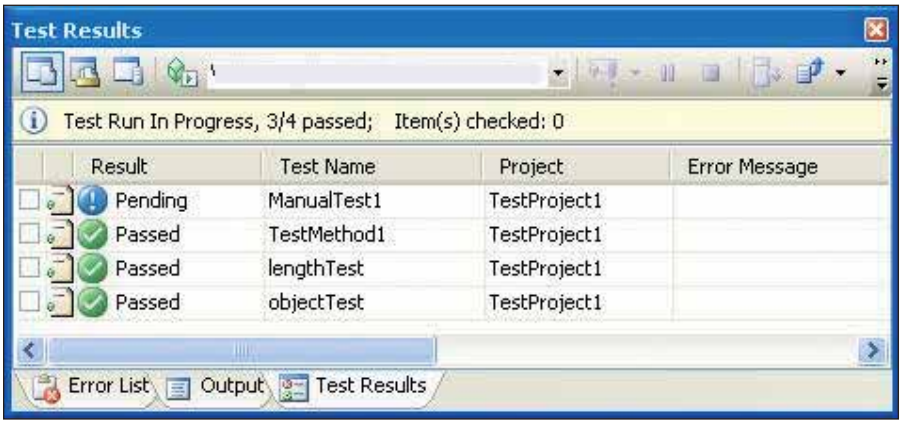
La traçabilité est devenue primordiale dans le développement. Grâce au pattern référentiel unique, on connaît la motivation d'une modification de code source. Par exemple, c'est comme cela qu'on a réussi à expliquer précisément pourquoi la facture de ManBower présentait une erreur de 1% : c'est l'ancienne TVA qui avait été appliquée dans le moteur de pricing, et la différence entre la nouvelle et l'ancienne TVA vaut 1%. Heureusement que les check-in sont reliées aux demandes de développement et aux tests.

Les managers de la World JobBoard Company se servent des rapports Team System. Par exemple, ils connaissent l'état d'avancement réel du projet sur la base des informations renseignées par les développeurs dans Visual Studio. Ils regardent aussi la courbe de vélocité de l'équipe de développement afin de calibrer des demandes de manière réaliste à la capacité de production, ou encore de renforcer les effectifs en cas de besoin…

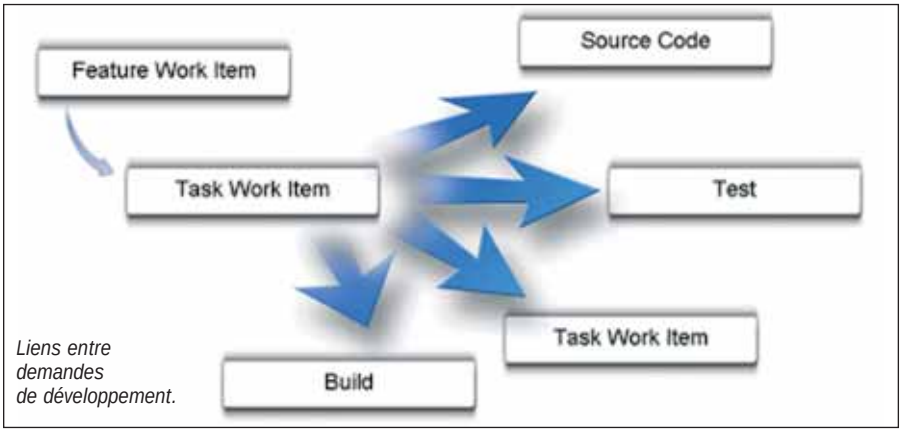
L'évolution du nombre de demandes de développement implémentées, et celle du nombre de tests unitaires implémentés les renseigne aussi sur la « bonne santé » du projet.

Côté run, nous sommes maintenant hébergés sur une ferme de 10 serveurs IIS dédiés et un datacenter SQL Server en cluster actif – passif, hautement disponibles, totalement externalisés chez un hébergeur de renom. Celui-ci s'engage sur une qualité de service, et planifie les demandes de mise en production des mois à l'avance. Personne ne peut accéder directement à la production hormis eux. Les mises en production de nouvelles versions sont au nombre de 2 par an. Et jamais le vendredi.

Au bout de 10 années d'existence, on propose près de 100 000 offres mensuelles et on compte près de 10 millions de visiteurs uniques mensuels.



Exécution de tests Team System.



Liens entre demandes de développement.

Team System, l'outil pivot de la transition

Les projets de ces deux zones adressent des enjeux radicalement différents : complexité, planification, respect de structures procédurières dans le premier, et rapidité, brouillard fonctionnel, time to market dans le second.

Inspirée de la réalité, cette histoire illustre des natures de projets très différents : ceux qui tendent à rationaliser et ceux qui expérimentent de nouveaux concepts métier, comme un nouveau site emploi spécialisé.

Adresser simultanément les enjeux de l'innovation et de la rationalisation est impossible compte tenu des divergences de contraintes et contextes. Dans le SI, la première des stratégies consistera donc à organiser des univers pour chacun, et d'instaurer un flux permanent entre les deux : les innovations qui marchent traverseront la frontière pour être réintégrées aux SI rationalisés.

Il faut en être conscient et permettre cette trajectoire d'évolution dès que l'orientation principale du logiciel ne sera plus le time to market et le brouillard mais la sécurité, l'intégration et la maîtrise des coûts.

Team System saura être **l'outil pivot de cette transition.** Il propose un outillage intégré et productif autour du build, des tests unitaires, et de l'intégration à Visual Studio particulièrement utile aux contextes innovants. Sur la base d'un pattern référentiel unique, il autorise un pilotage et une traçabilité tout à fait intéressants dans les contextes rationalisés.

On the road again…

Accoudé au comptoir, Jean-Marc m'observe puis s'exclame : « *Tu vois, moi je trouve que les sites d'emplois, ils sont mal faits. Et y compris les spécialisés ! Nous faire remplir des formulaires, c'est nous faire rentrer dans des cases. Mais moi, j'en ai déjà un de CV : c'est un document au format Word qui contient mes nom, prénom, âge, profession et mes expériences passées. Franchement, y'a un truc à faire, non ?!* ». Et c'est reparti…

Messaoud Oubechou
Architecte chez OCTO Technology,
meo@octo.com
Centre de Compétences .NET
OCTO Technology est certifié Centre de Compétences Team System

Intégrer Maven 2 à Eclipse et NetBeans

Cet article n'a pas pour fonction première l'explication détaillée de la configuration et de l'usage de Maven 2 mais de vous expliquer comment l'intégrer au sein des IDE Eclipse et NetBeans via leur plug-in respectif.

Maven 2 est un outil de build permettant de suivre le cycle de vie d'un projet en proposant les phases suivantes (http://www.oqube.com/formations/maven/lifecycle.html) :

validate	Valide que le projet est syntaxiquement correct et que toutes les informations nécessaires sont présentes
generate-sources	Génère le code source à inclure dans la compilation
process-sources	Pré-traitement du code source, par exemple pour filtrage de valeurs ou instrumentation
generate-resources	Génère les ressources à inclure dans l'artefact
process-resources	Copie et traite les ressources dans le répertoire de destination avant empaquetage
compile	Compile le code source
process-classes	Post-traitement du bytecode compilé
generate-test-sources	Génère le code source des tests
process-test-sources	Pré-traitement des sources de test avant compilation
generate-test-resources	Génère les ressources pour le test
process-test-resources	Copie et traite les ressources pour le test
test-compile	Compile les sources de test
test	Exécute les tests unitaires
package	Empaquette le code compilé (et les ressources) pour distribution
pre-integration-test	Exécute des actions nécessaires à la réalisation des tests d'intégration (eg. instancier l'environnement d'exécution)
integration-test	Traite et déploie le paquet si nécessaire pour exécution des tests d'intégration
post-integration-test	Exécute les actions requises après la fin des tests d'intégration (eg. nettoyage)
verify	Vérifie l'intégrité et la qualité du paquet
deploy	Déploie le paquet final pour intégration ou production

Installer Maven 2

Pour permettre une installation de Maven 2, la jdk doit être installé et la variable d'environnement JAVA_HOME doit être renseignée.

- La premiere étape consiste à télécharger l'archive de Maven sur le site d'Apache (http://maven.apache.org/) et à la décompresser dans un répertoire de votre choix (Exemple : c:\maven2).
- Il convient ensuite de mettre à jour le PATH en y ajoutant le chemin du répertoire bin de maven : C:\maven2\bin.
- La derniere étape consiste à vérifier la bonne installation de Maven via une ligne de commande DOS : mvn --version permettant d'afficher la version de Maven installée (Figure 1)

Préparation de l'installation des plug-in

L'utilisation du plug-in Maven d'Eclipse requiert la création dans le " document and settings " de l'utilisateur, d'un répertoire .m2 contenant le fichier de configuration de Maven sous peine de voir afficher un message d'erreur lors de la vérification de l'installation du plug-in dans Eclipse. Pour la création du répertoire " .m2 ", cela est possible via la commande Dos " md .m2 " car notons que Windows refuse la création d'un répertoire commençant par un point (cf figure 2).

Fig. 1 - Verification de l'installation de Maven 2

Fig. 2 - Création du répertoire .m2

Une fois ce répertoire créé, le fichier settings.xml se trouvant dans le repertoire conf de maven doit y être copié.

Installation du plug-in Maven dans Eclipse

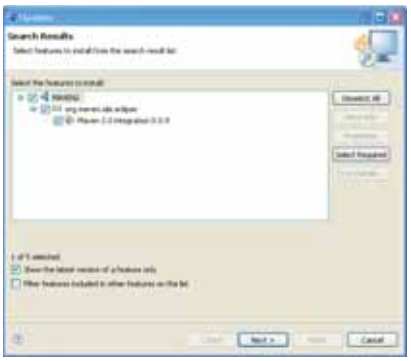


Fig. 3 - Choix des fichiers de Maven

Dans le menu " Help " d'Eclipse, choisir " software updates > Find and install ", cela a pour conséquence l'ouverture d'une fenêtre où l'option " Search for new features to install " est à choisir.

Nous allons télécharger le plug-in sur le site de l'éditeur de Maven.

Pour ce faire, cliquer sur le bouton " New remote Site... " et saisir les informations suivantes : Name : Maven 2 - URL : http://m2eclipse.codehaus.org/

Dans la nouvelle fenêtre qui vient de s'ouvrir, choisir l'ensemble des fichiers Maven comme présenté dans la figure 3, puis cliquer sur le bouton next pour pouvoir accepter la licence d'utilisation.

La suite de l'installation est assez simple, car elle consiste à valider l'ensemble des questions posées par Eclipse. Une fois l'installation terminée, Eclipse doit être redémarré.

Vérification de l'installation du plug-in

Aprés avoir redémarré Eclipse, rendez-vous dans le menu " Windows->Préférences " et selectionnez le plug-in " Maven 2 ".

Si vous avez respecté les prés requis d'installation du plug-in, vous devriez être en mesure de visualiser la fenêtre de configuration comme présenté dans la figure 4 signifiant la réussite de l'installation du plug-in au sein d'Eclipse.

Installation du plug-in Maven 2 dans Netbeans

Dans la fenêtre de gestion des plug-in (accessible via le menu tools->plugins) saisir " MAVEN " dans le champ de recherche. Sélectionner ensuite le plug-in Maven dans la partie gauche de cette fenêtre. Cliquer

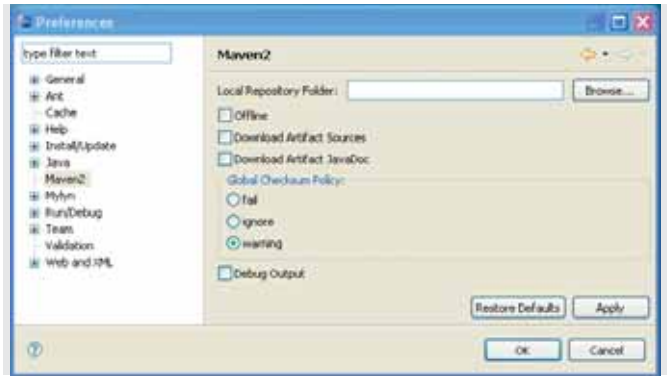


Fig. 4 - Configuration de Maven 2 dans Eclipse

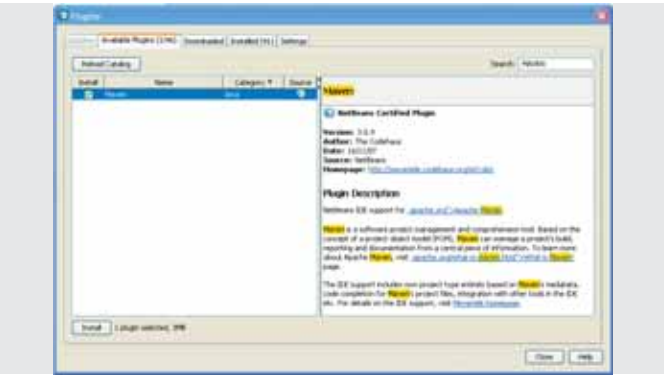


Fig. 5 - Installation du plug-in Maven dans NetBeans

ensuite sur le bouton " install " situé en bas à gauche (cf figure 5) et suivre les instructions à l'écran pour terminer l'installation du plug-in.

Vérification de l'installation du plug-in

Pour vérifier la bonne installation du plug-in, il suffit de créer un nouveau projet via le menu " File->New Project " et de vérifier la présence du nouveau type " Maven " (cf figure 6).

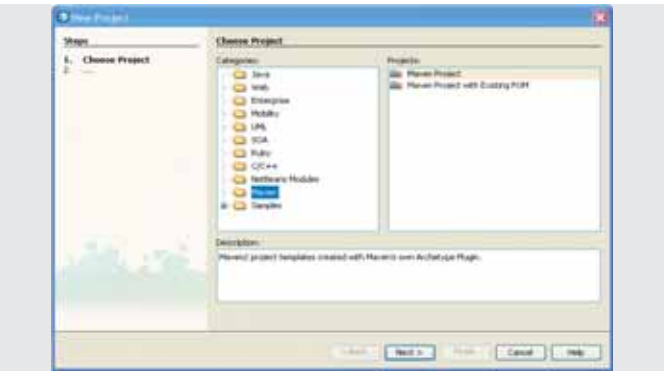


Fig. 6 - Type de projet Maven

Création d'un projet MAVEN à l'aide de NetBeans

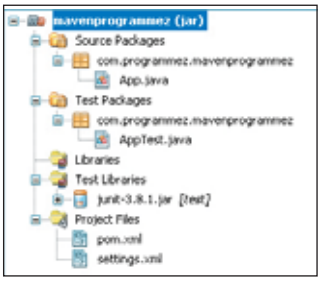


Fig. 7 - Architecture du projet Maven

Nous allons créer un nouveau projet Maven qui nous permettra de comprendre le fonctionnement de cet outil. Pour commencer, créons un projet de type " Maven " (File ->New Project). Sélectionner ensuite " Maven QuickStart Archetype ". Enfin, éditons les propriétés de notre projet comme suit :
Project Name : " MavenProgrammez "
Group Id : " com.programmez "
Maven construit ensuite son architecture comme en figure 7. Ce projet contient une classe initiale chargée d'afficher le traditionnel " Hello world ".

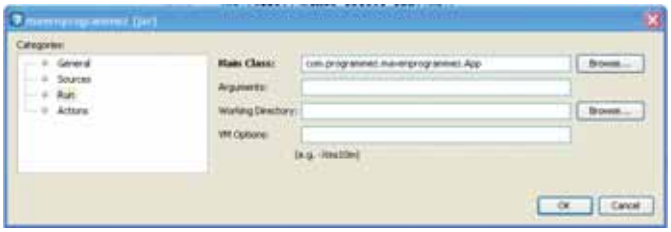


Fig. 8: Spécification de la classe principale du projet

Phase de test du projet

Pour tester le projet, nous devons créer un profil dans le fichier pom.xml. Ce profil est créé automatiquement en éditant les propriétés " Run " du projet (clic droit sur le projet->properties), et consiste en la spécification de notre classe principale (cf figure 8) : com.programmez.mavenprogrammez.App

Nous pouvons à présent tester notre projet. Via un clic droit sur le nom du projet, choisir le menu " Test " et observer le travail de Maven qui consiste à tester l'exécution et la compilation du projet.
Si tout se passe bien, la réussite du test doit être constatée.

Phase d'exécution du projet

Via la même procédure qu'énoncée précédemment, choisir cette fois-ci l'option " Run ". Après que Maven ait exécuté un ensemble de tests, nous pouvons voir apparaître le message " Hello World ".

Phase de build du projet.

Toujours, de la même façon, sélectionnons l'option " Build ".
De nouveau, Maven effectue une série de tests et nous informe que la création du JAR de notre application s'est correctement déroulée et se trouve pour mon cas, dans le répertoire suivant : C:\Documents and Settings\avannieuwenhuyze\Mes documents\NetBeansProjects\mavenprogrammez\target\mavenprogrammez-1.0-SNAPSHOT.jar

Phase de génération de la documentation

Via le menu contextuel de notre projet, sélectionnons " Generate JavaDoc ". Le résultat de cette opération est visible pour mon cas dans le répertoire : C:/Documents and Settings/avannieuwenhuyze/Mes documents/NetBeansProjects/mavenprogrammez/target/site/apidocs\
Vous êtes à présent en mesure de créer un projet Maven à l'aide de NetBeans. Bien entendu, les possibilités de Maven ne s'arrêtent pas à celles énoncées à travers l'exemple, mais ce dernier vous permet d'avoir un bref aperçu des fonctionnalités proposées dans le domaine de la gestion du cycle de vie d'un projet. La maîtrise de Maven passe par une étude approfondie de sa documentation, sur laquelle il me semble judicieux de s'attarder quelques instants afin de pouvoir profiter pleinement de la puissance de cet outil.

Aurélien Vannieuwenhuyze
Chef de projet Contentia Belgique (www.contentia.be)
Blog : aurelienv.no-ip.org

Quelques outils pour le cycle de vie des applications

Les solutions autour de la notion du cycle de vie logiciel (on n'abordera pas ici les données et les services), abondent et plusieurs poids lourds se partagent le marché comme Telelogic, HP, IBM, Microsoft et dans une moindre mesure, Borland, Compuware. Finalement, les solutions ALM complètes de bout en bout sont peu nombreuses, on trouvera beaucoup de solutions spécialisées sur le processus ALM (méthodologie, bonnes pratiques), les gestions de bugs, le changement, les tests, etc. Car il ne faut pas oublier que la notion même d'ALM recouvre une multitude de notions. Il est donc fréquent d'utiliser plusieurs outils différents pour monter sur ALM et ensuite avoir plutôt une tour de contrôle pour piloter le tout, une sorte d'Open ALM dans la définition de Borland.

ALM : une valeur ajoutée pour les éditeurs ?

On remarque aussi que l'offre ALM open source est en retrait par rapport à l'offre commerciale. Dans tout ce qui touche à la gestion du changement, de configuration, des exigences, gouvernances, les solutions open source sont peu fréquentes, et restent, plutôt l'apanage des éditeurs commerciaux qui y voient là, une valeur ajoutée à proposer aux clients / utilisateurs. Il y avait bien le projet Eclipse ALM, mais il est moribond depuis ses débuts.

Par contre, dans la collaboration ou le processus projet, il y a du mouvement. Avec notamment Eclipse Framework Project qui est actif et disponible dans plusieurs outils comme OpenUP. Dans la collaboration, le très attendu Jazz sera disponible à partir de mi-2008. Côté gestion et suivi des erreurs / bugs, il y a le très performant Mylyn. Dans la gestion de projets, il existe le projet Trac qui rencontre un beau succès et permet, à moindre frais, de mettre en place un environnement ALM léger.

Une sélection d'outils de cycle et de build

Ant	*APACHE*	Le meilleur outil open source pour créer et automaiser les build. S'adapte à de nombreux langages et outils.
Caliber DefineIT, CaliberRM	BORLAND	Solution pour définir les projets, les demandes, les besoins. Inclut la planification et la gestion du changement.
StarTeam	*BORLAND*	Outil de gestion du changement et des configurations logicielles permettant aux équipes de développement d'implémenter des processus sécurisés de suivi des versions tout au long du cycle de vie des applications.
OptimalJ	*COMPUWARE*	Suite logicielle de développement dont les bénéfices sont la rapidité de développement, un code de meilleure qualité et conforme aux standards Java EE. Utilise les spécifications MDA
Changepoint	*COMPUWARE*	Environnement de gouvernance et de gestion du portefeuille de projets.
Maven	*APACHE*	Outil de conduite / gestion de projet
Surround SCM	*SEAPINE*	Surround SCM est un logiciel de gestion de configuration.
TestTrack Pro	*SEAPINE*	TestTrack Pro est un logiciel de gestion des anomalies, des demandes d'évolutions.
Visual SourceSafe	*MICROSOFT*	Outil de gestion de sources, de versions
Visual Studio Team System	*MICROSOFT*	Environnement complet de développement en équipe de Microsoft avec une gestion complète du cycle de vie du logiciel.
Perforce	*PERFORCE SOFTWARE*	Perforce, le système rapide de gestion de configuration logicielle, assure le suivi et la gestion du développement logiciel pour les entreprises de toutes tailles.
Rally	*Rally Software*	Outils de gestion de projets avec méthode agile.
ChangeMan	*SERENA*	Outil collaboratif et de communication pour les équipes de développement. S'intègre avec les IDE .Net et Java. Supporte LDAP, WebDav
Dimension Express	*SERENA*	Outil de configuration et gestion de version
Dimension RM	*SERENA*	Environnement pour la gestion des exigences et de traçabilité
PVCS Version Manager	*SERENA*	Gestion de version
PVCS Professionnal Suite	*SERENA*	Gestion du changement
Change	*TELELOGIC*	Solution de gestion du changement et de traçabilité.
Doors	*TELELOGIC*	Outil de gestion des exigences
Synergy	*TELELOGIC*	Outil de gestion de configuration
Change Control Management	HP Sofware	Environnement complet pour gérer les changements couvrant la problématique de bout en bout.

Du côté IBM Rational

ClearQuest	Automatise et applique les processus de développement. Il permet de mettre en place une gestion de cycle de vie et s'intègre aux outils de tests, de build, etc. Permet une gestion de configuration.
ClearCase	Environnement de gestion de cycle de vie et de gestion du changement. Intègre un outil de versions, une gestion de build. L'édition ClearCase MultiSite est une extension spécifique aux environnements distribués.
RequisitePro	Une solution simple et efficace de gestion des exigences qui permet d'aiguiller les équipes de projet en leur permettant de créer, d'analyser et de gérer les exigences applicatives et les cas d'utilisation.
Porfolio Manager	Environnement complet de gestion du portefeuille projet. S'inscrit dans une stratégie de gouvernance.
ProjectConsole	Tableau de bord pour gérer au mieux ses projets en générant des graphiques, des rapports. Permet aussi d'avoir une vue globale sur les métriques projets utilisés.

François Tonic

Salon Studyrama

des Métiers de l'Informatique & des Nouvelles Technologies

Formations & Orientation
De Bac à Bac +5

20 JANVIER 2008

PARIS

CITÉ DES SCIENCES ET DE L'INDUSTRIE

ESPACE CONDORCET

Ⓜ Ⓣ Pte de la Villette - de 11h à 18h

INVITATIONS GRATUITES SUR WWW.STUDYRAMA.COM

Infoline 0891 65 1000 (0,225 €/min.)

Lesjeudis.com Emploi Informatique et Ingénierie

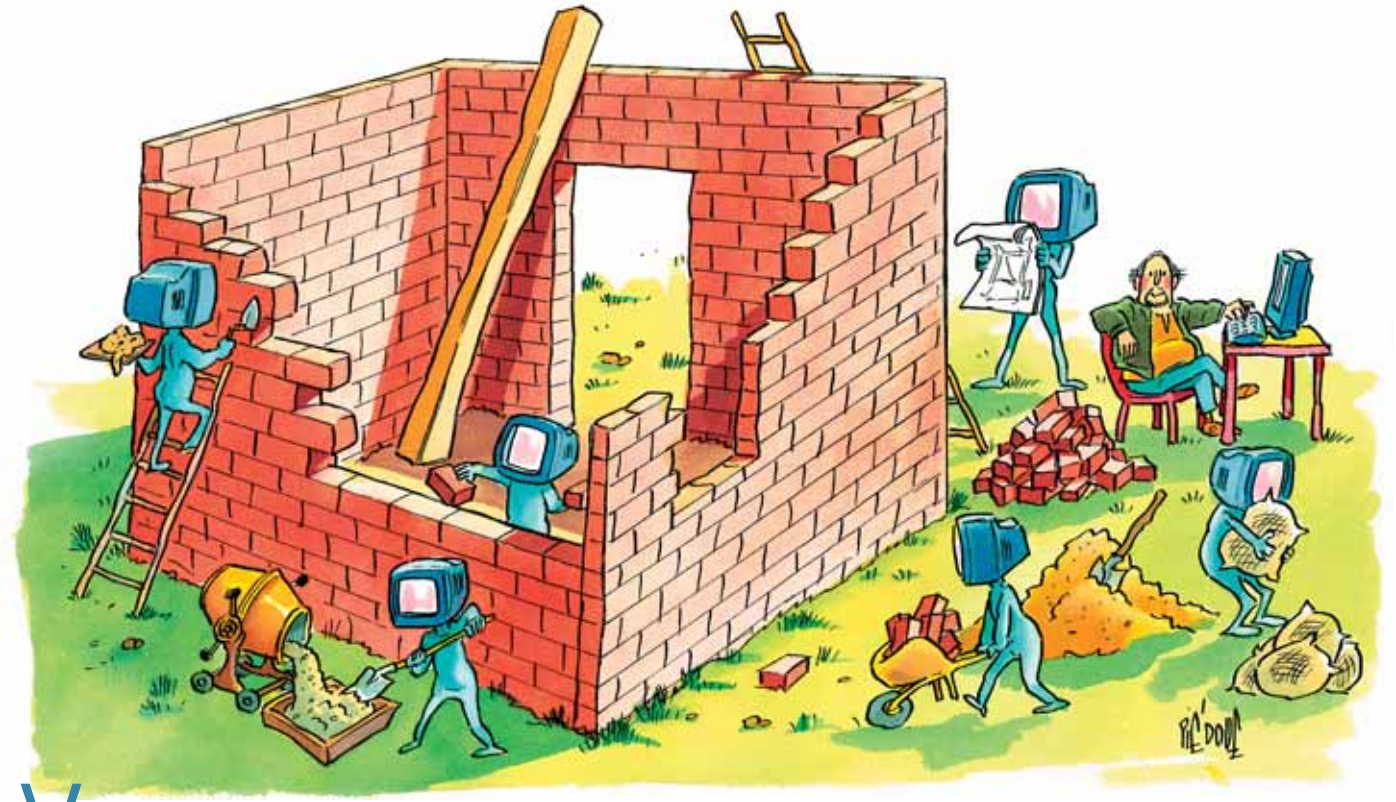
A NOUS PARIS LE NEWS URBAIN www.anous.fr

La Tribune

Courrier cadres

L'usine à build

ou l'art de construire les applications



Vous avez sans doute déjà installé des outils en version " nightly build ". Ce sont des versions souvent générées chaque jour, durant le soir ou la nuit. Tout développeur, ou presque, produit des build. Il s'agit de vérifier et de compiler le code de son application, de son projet. Le build peut servir à créer une application exécutable, utilisable et à tester le code, les composants, les modules d'un projet, notamment grâce aux tests unitaires, générant un rapport de tests qui fournira une listes des erreurs, manques, bugs, etc. D'autre part, les IDE, dans leur grande majorité, intègrent un ou plusieurs modules de build paramétrables, plus ou moins avancés. Aujourd'hui, les systèmes de build les plus connus sont Ant / Nant, Maven, MS Build ou encore les autotools. Bref, le choix ne manque pas. Dans ce dossier, nous avons voulu, mettre en avant la pratique avec Ant, Maven, autotools et MS Build avec une large portion de mise en œuvre, d'intégration. Selon les éditeurs, on bénéficiera de solutions plus ou moins intégrées, avec de véritables usines à build comme peut l'être un BuildForge d'IBM Rational.

Du simple build à l'intégration continue

Que l'on parle d'itération ou d'intégration continue, dans le cycle de vie d'un projet, il y aura tôt ou tard des outils de build à mettre en place, à utiliser. L'intégration continue constitue une facilité de tests et de vérification constante des projets. Pour cela, on peut utiliser des solutions commerciales ou open source telles que Hudson, Continuum, Cruise Control ou encore Luntbuild. On touche avec l'intégration continue à la notion de qualité logicielle. On peut ainsi faire du build en itération ou encore simplement, à chaque modification du code, d'un module.

Bien entendu, on ne peut faire de bonne usine à build sans les outils nécessaires. Il est ainsi important de gérer les sources dans une gestion de source de type CVS, Subversion. Les deux vont pour ainsi dire de pair. Faut-il alors un build unique, par exemple, en mettant en place du nightly build ? Les avis divergent. Tout dépend de la taille du projet, s'il s'agit d'un développement distribué entre plusieurs sites à fuseaux horaires différents, etc. Il faut avoir un équilibre entre le nombre de build, l'avancée du projet, les tests, les corrections des erreurs. Ensuite, vous devrez aussi choisir entre un build complet (qualifié de clean build) et un " build incrémental ". Le premier permet de partir d'un build plus sain, dans le second cas, le build sera plus rapide car ne prenant en compte que le code modifié, rajouté. Cependant, on peut alors introduire des effets de bord, des bugs. Le clean build, lorsqu'il est possible, est sans doute à privilégier. Les outils de build ne possèdent pas forcément des mécanismes d'automatisation de build passant par des scripts mais aujourd'hui, ils se sont très largement généralisés. À vous de construire au mieux votre application !

François Tonic

Ce qu'apporte le cycle de vie de la livraison de logiciels

Si dans ce numéro, on traite à la fois du cycle de vie et du build, ce n'est pas un hasard. Comme nous l'avons déjà dit à plusieurs reprises, les deux notions interagissent ensemble. Surtout si nous mettons en avant la notion de cycle de vie de la livraison de logiciels ou SDLC (Software Delivery Lifecycle).

Cette notion peut être indépendante de l'ALM, ou totalement intégrée. Nous considérerons ici son indépendance pour mieux comprendre son approche, son principe. Le SDLC peut se faire avec des solutions intégrées mais aussi avec de " simples " outils de build comme Maven ou Ant qui autorisent un certain cycle de vie du build, mais pas spécialement de la délivrance applicative. Encore une fois, n'ayez pas peur des termes et de l'approche. Si cela change effectivement la vision purement développeur, le SDLC offre une rationalisation des processus de build et une disponibilité bien plus grande que dans des procédures manuelles, même si elles sont structurées.

Mais c'est quoi réellement le SDLC ?

Dans le SDLC, il y a une notion de "rôles". Selon IBM, il faudrait en voir quatre (pas forcément des personnes différentes) :

- le développeur
- l'ingénieur de build
- le responsable de déploiement / de production
- le testeur

Au-delà des rôles, voici un exemple d'architecture d'un SDLC (Fig. A).

Cela permet de décloisonner le rôle de chacun et surtout d'avoir une approche transverse du build, un peu à l'image de l'ALM. On peut ainsi : favoriser la réutilisation, mieux gérer les sources, les projets, les branches, le versionning, les scripts de build, les différentes configurations, gérer l'accès des équipes, appliquer des règles de fonctionnement, de codage communes, établir des tableaux de bord de fonctionnement, etc. On peut ainsi définir deux challenges, deux pôles d'intérêts pour le SDLC :

Les défis	Les impacts
Automatiser des tâches, fonctions (à opposer à l'organisation purement manuelle)	Etre plus respectueux des délais, des itérations
	Mieux gérer les versions du projet
Définir de vrais processus consistants, évoluer avec les besoins réels des personnes et des projets	Réduire les coûts des build pour un processus automatisé et centralisé
Limiter les dépendances envers une personne, une équipe. Ainsi dans le build, éviter qu'une seule personne ou équipe comprenne et définisse le build, il faut un partage de l'information, et faire les scripts en transparence et collaborer avec les autres intervenants du projet (d'où la mise en place de processus)	Réduire les coûts en réutilisant les composants, objets, en harmonisant les environnements de conception, de développement, de build
	Mieux tracer le projet, le build
Améliorer la mise en place des tests, améliorer la résolution des problèmes, des bugs, le plus tôt possible. Mieux gérer les exigences, les cas de tests, les tests en général	

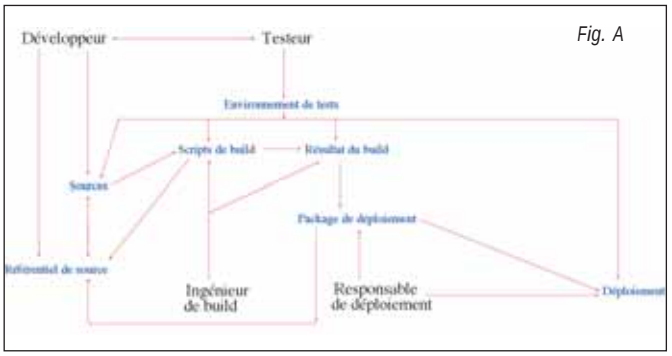


Fig. A

Bref, créer et gérer une assembly, un package, un JAR, etc. ne doit plus être une phase isolée de son projet, de son cycle de vie. Il rentre dans un cadre structuré pour simplifier le travail du développeur, du responsable de build. Même si aujourd'hui, le SDLC n'en est qu'à ses débuts, les outils nécessaires existent pour améliorer encore le build…

Adobe (source : IBM Rational), indique économiser pas moins de 25 millions de dollars par an en automatisant sa délivrance logicielle. Cela est particulièrement significatif si vous gérez des dizaines de projets. Pour une petite structure, un développeur indépendant, il est inutile d'investir dans une suite SDLC de type Borland ou IBM (ou autres), mais vous pouvez sans aucun doute gagner du temps en organisant la gestion des projets, des sources et des builds dans un référentiel unique…

En moyenne (source : Hurwitz & Associates), on peut espérer gagner par le SDLC :	
Build plus " rapide ", plus souvent	110 %
Productivité des équipes	42 %
Fréquence des versions, itération	40 %
Réduction des erreurs	30 %
Productivité du développeur	28 %
Réduction des coûts de développement	25 %

Il ne faut pas voir le SDLC et l'automatisation du build comme une source importante et rapide de réduction des coûts, car la mise en place peut, au contraire, charger le budget durant les premiers mois : mise en place, formation, montée en charge des processus. Mais à terme, il y a un avantage certain.

Un SDLC intégré : exemple de Build Forge (Fig. B)

Build Forge n'est donc pas une usine à build mais un cycle de vie du build dans son ensemble. Il a la capacité d'être indépendant de la plate-forme, des langages, des outils, des projets. C'est peut être basique de le préciser, mais il est important de le noter. On retrouve une partie gestion via une

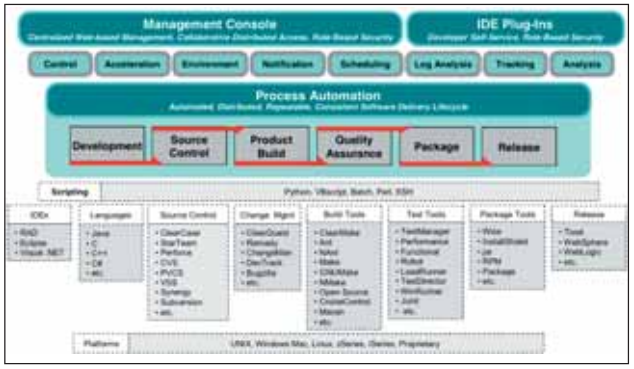


Fig : B

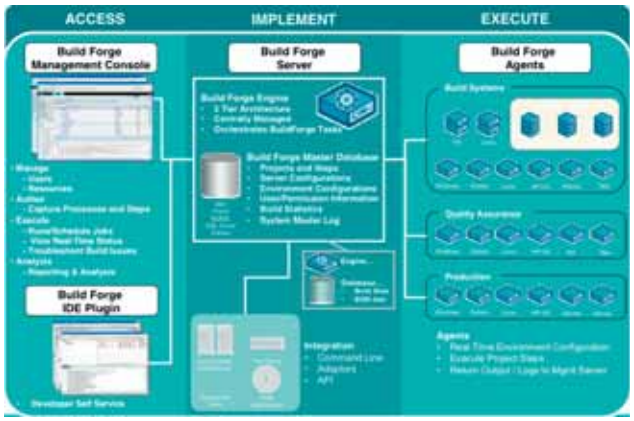


Fig : C

console web (avec la collaboration) et la partie intégration IDE (plug-in). Son fonctionnement est assez simple (Fig : C).

L'environnement se découpe en plusieurs zones :
- console d'administration et intégration avec les IDE (c'est l'accès)
- disponibilité d'un serveur Build Forge : architecture 3-tiers, gestion et orchestration des tâches, gestion des données, des projets, des étapes, disposition d'un serveur de configurations, etc. interfaces d'interface disponibles (c'est l'implémentation).
- déploiement d'agents pour les systèmes de build, la qualité assurance et la production (c'est l'exécution).

Dans le cas d'un développement décentralisé : une équipe à Paris, une autre à Grenoble, une autre à Tunis, L'outil Rational offre une vision dite GDD : Geographically Distributed Development (développement géographiquement distribuée). On dispose au " siège central " d'une base de données maître avec l'ensemble des données des projets, les configurations, les permissions, l'intégration continue, etc. Sur chaque site, on dispose alors d'un log à la base de données puis l'environnement de développement et de build standard à Build Forge.
Côté licence, l'offre se veut flexible pour s'adapter aux besoins. Bien entendu, le nombre d'utilisateurs reste monétisé mais il y a le choix entre une licence flottante et fixe. Attention, nous sommes là sur des environnements dédiés aux très gros projets, le prix est en rapport…

François Tonic

Maven et le build :

Maven est un outil qui a été conçu historiquement pour un projet de la fondation Apache. Son objectif était de simplifier et de standardiser le processus de compilation de Jakarta Turbine. Ce dernier est un framework permettant le développement d'applications web selon l'approche RAD.

Maven permettait donc de remplacer les scripts Ant devenus trop complexes à maintenir, par des scripts standard et simplifiés. Par la suite, l'utilisation de cet outil s'est élargi à d'autres projets de la fondation et est désormais utilisé par de plus en plus de développeurs.

Les fonctionnalités de Maven

Maven apporte un certain nombre de fonctionnalités clefs par rapport à Ant. Celles-ci permettent notamment une meilleure gestion des dépendances et une capacité à travailler avec plusieurs projets en même temps. Maven est également très efficace en ce qui concerne la gestion des versions et la distribution des applications générées. Ant possède son propre fonctionnement et ses propres mécanismes. Maven ne les utilise pas mais apporte une réponse équivalente pour chacun d'entre eux. Ainsi Maven est à même de produire de la documentation sous forme de PDF ou de sites web et peut être étendu par des plug-in. Maven dispose également d'une compatibilité avec les scripts Ant.

Créer un projet

Contrairement à Ant, Maven est orienté RAD. Il permet donc de générer la structure du projet, un peu à la manière de Ruby On Rails. Ainsi, pour créer votre projet en ligne de commande, tapez ceci :

```
mvn archetype:create -DgroupId=com.mycompany.app -DartifactId=my-app
```

Maven va créer l'arborescence par défaut du projet et notamment le fichier utilisé pour la compilation : *pom.xml.* Celui-ci est le fichier de configuration principal qui contient l'ensemble des paramètres du projet.

Commandes de compilation

Avec Ant, on ne dispose que des cibles que l'ont définit au sein du fichier de configuration. Ce mécanisme était déjà le même avec les Makefile. Maven propose un concept différent avec un socle standard mettant l'accent sur le cycle de vie du projet. Ainsi Maven propose les actions : *compile, test, package, deploy, install.* Il existe également d'autres tâches en dehors de ce cycle de vie dont l'exécution n'est pas assurée systématiquement : *assembly:assembly, site, site-deploy*… Avec Maven, vous pouvez donc compiler le projet par la commande suivante :

```
mvn compile
```

Si vous souhaitez déployer votre application par la commande deploy, Maven va effectuer toutes les autres tâches en séquences au préalable. Il va d'abord compiler puis exécuter les tests et enfin archiver les

une combinaison efficace !

binaires. Si un échec intervient, le déploiement s'arrêtera. Pour exécuter des tâches en séquence, on les tapera simplement dans l'ordre souhaité :

```
mvn clean dependency:copy-dependencies package
```

Configuration de la compilation

Le fichier *pom.xml* contient les informations sur le projet et notamment la configuration de compilation. Il s'agit de la partie la plus importante du fichier. Elle se décompose en quatre sections : *properties, packaging, build et reporting.* La première permet d'insérer des variables qui seront utilisées notamment par les plug-in. Elle n'est pas d'une importance capitale. La section packaging, en revanche, contient les informations relatives aux packages qui peuvent être générés : *pom, maven-plugin, ejb, war, ear....* Bien sûr, par défaut c'est un *jar* qui est généré. La section build décrit le processus de compilation que Maven devra utiliser. On y retrouve principalement l'emplacement des sources ainsi que du dossier de sortie. Quelques autres emplacements sont définis comme le dossier contenant les ressources du projet. C'est également ici que sont définis les plug-in.

```
<build>
  <sourceDirectory>src/main/java</sourceDirectory>
  <scriptSourceDirectory>src/main/scripts</scriptSourceDirectory>
  <testSourceDirectory>src/test/java</testSourceDirectory>
  <directory>target</directory>
  <outputDirectory>target/classes</outputDirectory>
  <testOutputDirectory>target/test-classes</testOutputDirectory>
  <finalName>${artifactId}-${version}</finalName>
  <resources>
    <resource>
      <directory>src/main/resources</directory>
    </resource>
  </resources>
  <testResources>
    <testResource>
      <directory>src/test/resources</directory>
    </testResource>
  </testResources>
</build>
```

La section reporting définit les processus de génération de rapports. Cela comprend aussi bien la documentation Javadoc que les rapports de tests. Un très grand nombre de rapports sont pris en charge par Maven.

Gestion des dépendances

La caractéristique principale de Maven est de fonctionner en réseau. Ainsi, il est capable de récupérer les modules nécessaires à la compilation. Pour cela, nous devons définir les dépendances, Maven va se charger tout seul de récupérer les archives nécessaires sur le dépôt. Il se comporte ainsi comme un gestionnaire de paquets que l'on peut retrouver sur les distributions Linux. Le code suivant montre comment ajouter une dépendance envers la librairie JUnit :

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Créez vos propres *Nightly-Build* !

Les tâches principales assumées par Maven sont les suivantes : compile, test, package, install, deploy. Si nous souhaitons mettre en place un mécanisme de nightly build, nous devons écrire notre propre tâche. Celle-ci devra récupérer des sources sur le dépôt Subversion et compiler les sources projet. Elle se définit donc par le goal suivant :

```
<goal name="nightly-build">
    <attainGoal name="update" />
    <attainGoal name="package" />
</goal>
```

Vous connaissez déjà la tâche package qui va archiver les binaires dans un jar. La tâche update qui sera exécutée juste avant va mettre à jour les sources :

```
<goal name="update">
    <ant:exec executable="svn">
        <arg line="update" />
    </ant:exec>
</goal>
```

Pour planifier cette tâche, vous devrez, en fonction de votre système d'exploitation, utiliser le bon outil. Sous Windows, le planificateur de tâches est accessible via le menu démarrer dans les outils systèmes. La configuration se déroule à l'aide d'une interface graphique simple à utiliser. En revanche, sur les systèmes de type Unix, vous devrez passer par l'utilisation de Cron. L'exemple suivant exécute la nightly build tous les jours à 23h30 :

```
30 23 * * * mvn nightly-build
```

Conclusion

Cet article montre comment fonctionne Maven et le compare à la solution Ant. C'est un outil de plus en plus utilisé qui se distingue par sa grande modularité et sa prise en charge avancée des dépendances. Il s'agit à ce jour de la solution de build open source la plus évoluée pour les projets Java et Java EE.

Loïc Guillois

built by: maven

Compilation de projets Java avec Ant

Ant est un outil de build développé par la fondation Apache. Celle-ci est connue pour son serveur web du même nom. La fondation assure également les développements du framework Struts et du serveur Tomcat. Ant est donc le projet de référence en matière de compilation de solutions Java.

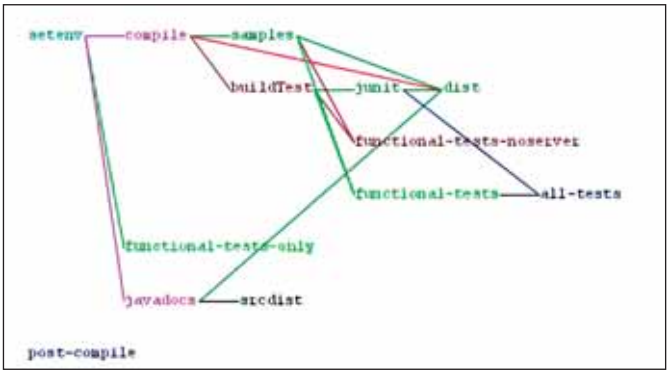
Selon votre environnement de travail, vous aurez la possibilité de travailler, soit en ligne de commande avec un éditeur texte, soit dans le cadre d'un IDE. Eclipse et Netbeans disposent tout deux d'une intégration de la solution Ant et son utilisation est transparente pour le développeur. Celui-ci a cependant la possibilité d'écrire lui-même ses scripts. Pour une utilisation sans environnement de développement intégré, vous devrez installer Ant. Il faudra pour cela télécharger les binaires du projet sur le site http://ant.apache.org. Une fois l'archive extraite où vous aurez choisi, il faudra ajouter l'emplacement de son dossier bin dans la variable d'environnement PATH. De même, les fichiers .jar de son dossier lib devront être ajoutés à la variable d'environnement CLASSPATH.

Les bases de Ant

Un fichier de compilation Ant se présente sous la forme d'un document XML. Pour les éditer, il suffit donc d'utiliser un éditeur de texte standard ou des outils plus évolués comme UltraEdit. Entrons dans le vif du sujet. Voici le squelette de base que l'on recommande souvent d'utiliser. Nommez ce fichier build.xml et placez-le à la racine du projet.

```
<?xml version="1.0"?>
<project name="test" default="compile" basedir=".">

    <property name="src.dir" value="."/>
    <property name="build.dir " value="build"/>
    <property name="dist.dir " value="dist"/>

    <target name="init">
        <mkdir dir="${build.dir}"/>
      <mkdir dir="${dist.dir}"/>
    </target>

    <target name="compile" depends="init">
        <javac srcdir="${src.dir}" destdir="${build.dir}"/>
    </target>
</project>
```

La première balise définit le projet et notamment son nom et la cible par défaut à exécuter lors de l'appel à la commande ant. L'attribut basedir correspond à la racine du projet, utile dans le cas où celle-ci n'est pas au même endroit que le fichier de compilation. Le projet peut disposer de propriétés. Celles-ci permettent de paramétrer plus facilement le fichier de compilation. Ici, les propriétés src.dir et build.dir contiennent respectivement l'emplacement des fichiers sources et des fichiers compilés. Nous utiliserons dist.dir par la suite pour y placer la javadoc ainsi que l'archive jar du projet. Une fois les propriétés du projet déterminées, il faut définir les cibles. Celles-ci jouent le même rôle que les cibles des Makefile. Elles définissent une tâche de compilation. Ici, la tâche init va simplement créer le dossier où seront placés les fichiers



compilés. Enfin, la cible compile constitue le cœur du script. Elle définit la commande de compilation ainsi que les cibles dont elle dépend. Dans notre exemple, nous allons simplement compiler les fichiers du dossier source vers le dossier de build. Pour exécuter le script, tapez la commande ant suivie de la cible. Si vous ne la précisez pas, celle par défaut sera prise en compte.

Quelques tâches utiles…

Ce premier exemple est simple, un peu trop pour exploiter la puissance de l'outil. Il ne permet pas la création des archives Jar et n'est pas capable de générer la Javadoc. On pourrait également souhaiter nettoyer les fichiers de build. Nous allons ajouter trois tâches au projet Ant pour combler ces manques. Tout d'abord, nous allons ajouter une petite cible de nettoyage. Celle-ci permet de supprimer l'ensemble des fichiers générés. Il est également possible d'avoir un traitement plus fin en supprimant les fichiers en fonction de leur type. Dans notre cas, nous souhaitons simplement supprimer les deux dossiers :

```
    <target  name="clean">
        <delete dir="${build.dir}"/>
        <delete dir="${dist}"/>
    </target>
```

Par commodité, les développeurs ne fournissent pas le résultat de leur travail sous forme de dossiers et de fichiers compilés disparates. Ils préfèrent utiliser un moyen de distribution plus efficace : l'archivage. La cible suivante génère le fichier jar avec pour classe de lancement la classe Test, s'il s'agit d'une simple librairie, la balise *manifest* devient inutile.

```
<target  name="jar" depends="init">
  <jar destfile="${dist.dir}/test.jar" basedir=".">
    <include name="${build.dir}"/>
    <manifest>
      <section name="Test.class" />
    </manifest>
  </jar>
</target>
```

Enfin, nous allons ajouter une dernière cible qui va permettre la génération de la documentation embarquée. La commande javadoc prendra en compte les fichiers sources et quelques paramètres pour générer la documentation dans le dossier dist/doc :

```
<target name="javadoc" depends="init">
  <mkdir dir="${dist.dir}/doc"/>
  <javadoc
        destdir="{dist.dir}/doc"
        author="true"
        version="true"
        use="true"
        windowtitle="Test">

    <packageset dir="${src.dir}" defaultexcludes="yes">
      <include name="**"/>
    </packageset>

    <doctitle><![CDATA[<h1>Test</h1>]]></doctitle>
      <bottom><![CDATA[<i>Copyright &#169; 2007 XXXXXX.</i>]]>
</bottom>
  </javadoc>
</target>
```

Surveillance

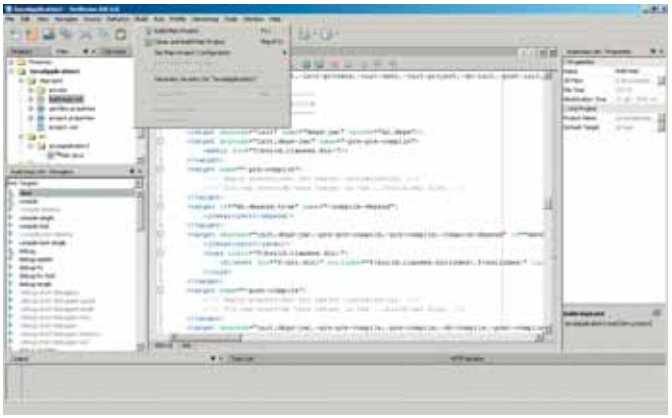
Après avoir vu les fonctionnalités principales de Ant, nous allons nous intéresser aux deux moyens d'assurer le suivi d'exécution des scripts : les loggers et les listeners. Dans les deux cas, vous pouvez soit utiliser ceux fournis par Ant soit développer votre propre solution. Un listener est alerté lors de l'apparition des différents événements Ant : début et fin du script, d'une cible et d'une tâche.
Un listener est également alerté lorsqu'un message est loggé. Un des listeners fournis par Ant utilise Log4J : la solution la plus répandue en ce qui concerne le traitement des messages applicatifs en mode console. Pour l'utiliser, on ajoutera le paramètre suivant :

```
ant -listener org.apache.tools.ant.listener.Log4jListener
```

Vous pourrez ainsi éditer le fichier log4j.properties et bénéficier d'une granularité appréciable dans la configuration de l'affichage. Vous pourrez, par exemple, écrire dans un fichier et filtrer les messages à afficher en fonction de leur niveau d'importance. Voici un exemple de configuration :

```
log4j.rootLogger=ERROR, LogFile
log4j.logger.org.apache.tools.ant.Project=INFO
log4j.logger.org.apache.tools.ant.Target=INFO
log4j.logger.org.apache.tools.ant.taskdefs=INFO
log4j.logger.org.apache.tools.ant.taskdefs.Echo=WARN

log4j.appender.LogFile=org.apache.log4j.FileAppender
log4j.appender.LogFile.layout=org.apache.log4j.PatternLayout
log4j.appender.LogFile.layout.ConversionPattern=[%6r] %8c{1} : %m%n
log4j.appender.LogFile.file=build.log
```

L'utilisation de ce module nécessite la compréhension du fonctionnement de Log4J. Il peut donc être intéressant de se reporter sur la documentation de cet outil afin de l'exploiter pleinement.



Ant et les IDE

Ant peut s'intégrer à la majorité des environnements de développements. Cela permet notamment de gérer l'ensemble des actions de compilations par l'interface graphique : compiler les sources, packager les fichiers compilés, nettoyer, générer la javadoc, compilation et exécution des tests JUnit… Ant est intégré soit nativement soit via un plug-in dans les outils tels Eclipse, Netbeans, JBuilder ou encore jEdit.

NAnt

A l'origine Ant était conçu pour le langage Java, cependant, son utilisation s'est largement répandue. A un tel point même, que certains développeurs .NET ont regretté de ne pas disposer d'outils équivalents. C'est pour cette raison que NAnt est né. Il ne s'agit pas d'un portage de l'outil sur la plate-forme .NET mais l'utilisation est très ressemblante. De la même façon, il existe de nombreux projets dont l'objectif est de porter des fonctionnalités de Java vers .NET : NUnit, NHibernate, Log4Net, NDoc…

Tests unitaires

Il existe un certain nombre de modules optionnels permettant à Ant d'étendre ses fonctionnalités dans un domaine bien spécifique. Nous allons nous focaliser sur deux d'entre eux : JUnit et JUnit Report. Le premier permet l'exécution des tests unitaires et le second génère les rapports de test. Voici une cible JUnit. On y retrouve la configuration de la sortie (un fichier xml), ainsi que la liste des tests à exécuter. Ces derniers sont définis au sein de la balise fileset par une expression régulière :

```
<target name="tests" depends="compile">
<junit printsummary="yes" haltonfailure="yes">
  <classpath>
    <pathelement location="${build.tests}"/>
    <pathelement path="${java.class.path}"/>
  </classpath>

  <formatter type="plain"/>

  <test name="my.test.TestCase" haltonfailure="no" outfile="result">
    <formatter type="xml"/>
  </test>

  <batchtest fork="yes" todir="${reports.tests}">
```

```
    <fileset dir="${src.tests}">
      <include name="**/*Test*.java"/>
      <exclude name="**/AllTests.java"/>
    </fileset>
  </batchtest>
</junit>
</target>
```

Le fichier généré par l'exécution de cette tâche peut être mis en forme grâce à JUnit Report. Le rapport final généré sera au format html et comprendra des cadres. JUnit Report peut en effet fusionner les résultats de plusieurs séries de tests en un seul rapport.

```
<junitreport todir="./reports">
  <fileset dir="./reports">
    <include name="TEST-*.xml"/>
  </fileset>
  <report format="frames" todir="./report/html"/>
</junitreport>
```

Pour conclure sur les tests unitaires, sachez qu'il existe un outil permettant de tester les applications Java EE et notamment les composants EJB. Cet outil s'appel Cactus et peut être également intégré et utilisé avec Ant. Cactus vous permettra ainsi de tester votre code au sein du serveur d'application (Tomcat, JBoss, WebLogic…).

Outils complémentaires

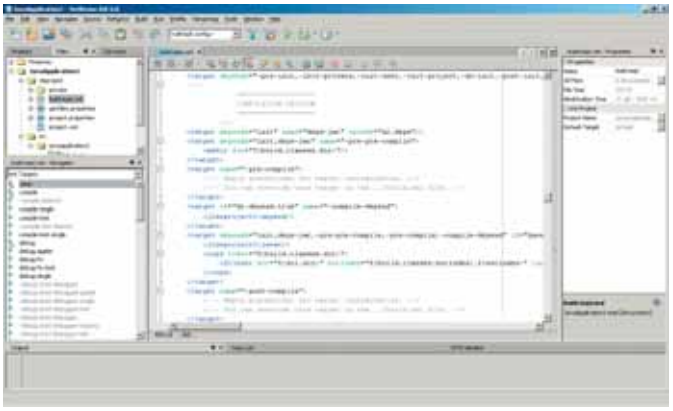
Ant peut fonctionner avec des outils externes qui permettent de faciliter et d'améliorer le processus de compilation ainsi que la qualité du code. Il en existe des dizaines et il serait trop long de tous les aborder. Nous allons donc passer en revue les plus intéressants, dans le cadre du développement d'un projet Java. Checkstyle est un outil permettant de vérifier que le code source respecte bien les conventions de codages mises en place en début de projet. En l'intégrant au script Ant, il est possible de s'en servir comme d'un outil de validation permettant d'éviter les revues de codes longues et ennuyeuses. Checkstyle est, entre autres, capable de vérifier l'indentation du code, le nommage des classes, méthodes et variables, ainsi que certaines erreurs d'exécution (code mort, types…). Voici la tâche Ant :

```
<checkstyle config="docs/sun_checks.xml">
  <fileset dir="src/checkstyle" includes="**/*.java"/>
  <formatter type="plain"/>
  <formatter type="xml" toFile="build/checkstyle_errors.xml"/>
</checkstyle>
```

Java n'offre pas actuellement de système de mise à jour des jar. Un outil permet la création de patchs qui peuvent être utiles dans certains cas : JarPatch. Celui-ci peut fonctionner en standalone ou bien via une tâche Ant. Le principe est simple. JarPatch va calculer la différence entre les deux jar et générer une archive qui fera office de mise à jour :

```
<jarpatch newJar="${build}.lib/${app.jar}" oldjar="${dist}/lib/${app.jar}"
      resultPatch="${dist}/patch/patch-${build.number}.zip" excludes=".*\.gif, .*\.jpg"/>
```

ProGuard est un obfuscateur de code. Il permet de modifier les binaires générés par le compilateur java afin de rendre très difficile les techniques de reverse engineering utilisées pour récupérer le code source à des fins malhonnêtes. Cette technique peut être nécessaire pour protéger l'algorithme qui vous aura coûté des mois, voire des années hommes de développements. ProGuard optimise également les binaires en supprimant les classes, méthodes et variables non utilisées. Il peut être utilisé avec Ant grâce à la tâche suivante. Le nombre de paramètres qu'il peut prendre en compte est très important :



```
  <proguard>
    -injars ${temp.jar}
    -outjar ${obfuscated.jar}
    -libraryjars ${midp.jar};${cldc.jar}
    -overloadaggressively
    -defaultpackage ''
    -allowaccessmodification
    -dontusemixedcaseclassnames
    -printmapping obfuscation-map.txt
    -keep public class * extends javax.microedition.midlet.MIDlet
  </proguard>
```

Pour finir, nous allons aborder un outil qui peut être utile pour comprendre les fichiers de compilation Ant. Ant2Svg permet la génération d'un schéma à partir du fichier ant. Le format Svg étant vectoriel, celui ci peut être zoomé et dézoomé sans perte de qualité, ce qui le rend très pratique pour une visualisation sur un navigateur ou un simple logiciel d'imagerie. Voici la tâche de génération du schéma vectoriel :

```
<ant2svg srcfile="mybuild.xml" destfile="doc/build.svg"/>
```

Il existe de nombreux autres outils qui valent le détour. Le site officiel du projet Ant référence les plus répandus. Vous y trouverez également des outils plus spécifiques permettant une intégration de Ant avec des logiciels propriétaires tels que JBuilder, ClearCase ou encore le studio de développement WebSphere.

Conclusion

Ant est l'outil open source le plus répandu en ce qui concerne la compilation de projet Java. Sa souplesse et sa robustesse en font la solution idéale pour de nombreuses configurations et il y a fort à parier que l'on ne le verra pas disparaître de si tôt. Cependant, d'autres solutions voient le jour comme Maven qui étend les concepts déjà mis en place avec une prise en charge avancée des dépendances.

Loïc Guillois

Présentation des Autotools

Les Autotools sont le système de référence dans la compilation de projets pour les systèmes Unix. Cet article a pour but de les présenter en s'appuyant sur un exemple réel qui est celui de NuFW.

Il s'agit d'un pare-feu authentifiant basé sur Netfilter, la couche pare-feu de Linux. Vu du côté des développeurs, c'est une série d'applications et de bibliothèques développées en C et utilisant les autotools pour la compilation.

Autotools

Principes

Le but des autotools est de fournir une méthode portable et standardisée de compilation des projets complexes. Pour la personne cherchant à compiler un logiciel utilisant les autotools, la démarche est toujours la même. Une fois l'extraction des sources effectuée, il suffit de lancer :

```
./configure
make
make install
```

L'exécution du script ./configure permet de détecter les particularités de l'environnement de compilation et d'y adapter celle-ci. L'utilisation des autotools permet aussi de fournir à l'utilisateur une série d'options intéressantes sans surcoût important de mise en place par le développeur :

- help : affiche les options disponibles de compilation
- prefix : installe le logiciel en prenant comme base le répertoire passé en argument
- sysconfdir : précise l'endroit où seront situés les fichiers de configuration

Il est possible de récupérer les variables, et les rendre accessibles au développeur assez facilement. Par exemple, pour obtenir l'emplacement par défaut d'un fichier de configuration, il suffit de préciser à autotools d'exporter la variable :

```
AC_DEFINE_UNQUOTED(CONFIG_DIR,"/etc/nufw")
AC_SUBST(CONFIG_DIR)
```

Ainsi la valeur est disponible dans la macro CONFIG_DIR, ce qui autorise à écrire dans un code en C la ligne suivante :

```
#define DEFAULT_CONF_FILE    CONFIG_DIR "/nuauth.conf"
```

Make

Le but des autotools est d'automatiser la création des fichiers Makefile qui sont utilisés par l'outil Make, l'outil de compilation standard des Unix, en fonction de l'architecture cible : autotools fonctionne sur tous les Unix, de manière portable.

Make est un "moteur de production", c'est-à-dire un outil visant à lancer les commandes nécessaires à la compilation d'un logiciel. À la différence d'un simple script, un moteur de production exécute les commandes seulement si elles sont nécessaires. Ceci permet d'optimiser le temps de génération du logiciel.

Par exemple, dans le cas d'un programme en C comportant plusieurs fichiers, la démarche standard consiste à compiler chaque fichier C pour obtenir des fichiers objets (.o). Ces fichiers objets sont ensuite assemblés et liés aux bibliothèques du système lors de la phase de lien. Si l'on se place dans le cas où un seul fichier C a été modifié, le chemin de compilation optimal consiste à le transformer en fichier objet puis à refaire la phase de lien. Make s'acquitte parfaitement de cette tâche en gardant une syntaxe élégante comme le montre ce Makefile :

```
OBJS  = main.o window.o interface.o
prog: $(OBJS)
        gcc -o prog $(OBJS) -lm
.c.o:
        gcc -c $*.c
```

La liste des fichiers objets est mise dans la variable OBJS et la ligne suivante indique que "prog" dépend des fichiers objets listés dans OBJS. L'obtention de "prog" est obtenue en lançant la commande de la dernière ligne où l'on a remplacé $(OBJS) par la listes des fichiers objets.

Les fichiers objets sont eux obtenus de manière implicite, grâce à la dernière règle qui dit comment obtenir un fichier .o à partir d'un fichier .c.

Sur un projet, l'édition manuelle des Makefile devient rapidement inenvisageable, d'autant plus que l'adaptation des directives de compilations au système doit être effectuée par l'utilisateur en éditant les fichiers. Pour assurer la portabilité de la procédure de compilation et faciliter la vie du développeur, il est donc nécessaire d'utiliser une surcouche à l'outil Make. La méthode la plus répandue consiste à utiliser les autotools.

Fonctionnement

L'idée globale consiste à partir de fichiers génériques pour les spécialiser ensuite de manière à obtenir des instructions de compilations spécifiques à une machine. Les fichiers distribués dans les archives de distribution doivent être indépendants du système. Ils sont suffixés par ".in". Les autotools sont en fait composés d'une suite d'outils :

- autoconf : génère un script configure depuis un fichier configure.ac ou un fichier configure.in
- autoheader : crée un fichier contenant les #define nécessaires à la spécialisation du code vis à vis du système cible
- automake : crée les fichiers Makefile.in à partir des fichiers Makefile.am
- aclocal : rassemble les macros nécessaires dans un fichier qui sera incorporé aux sources
- libtool : outil de création des bibliothèques dynamiques

Schéma de fonctionnement des autotools

Pour le développeur, l'utilisation des autotools consiste à rédiger un fichier "configure.ac" à la racine du projet et un fichier "Makefile.am" dans chaque répertoire où une action est nécessaire pour construire le logiciel.

configure.ac

Le fichier "configure.ac" est composé de macros et de fonctions qui visent à analyser si les dépendances demandées sont présentes et à détecter les spécificités du système. Il est transformé par "autoconf" en

script de configuration "configure". Ce dernier doit être lancé avant la compilation pour instancier les fichiers .in sur le système de compilation. Ces fichiers sont principalement les Makefile, mais il peut aussi s'agir de fichiers d'en-tête ou encore de scripts.

Syntaxe du fichier configure.ac

Le fichier "configure.ac" commence par des déclarations nécessaires à l'initialisation du script :

```
AC_PREREQ(2.57)
AC_INIT(NuFW, 2.2.10, nufw-devel@nongnu.org)
AM_CONFIG_HEADER(src/include/config.h)
AC_CONFIG_SRCDIR([src/nufw/main.c])
```

La première ligne indique que le logiciel a besoin de la version 2.57, ou supérieure, d'autoconf pour fonctionner. La suivante initialise le système :

- Le logiciel s'appelle NuFW
- La version du logiciel est 2.2.10
- Le mail de contact est "nufw-devel@nongnu.org"

AM_CONFIG_HEADER permet de fixer le nom du fichier qui contiendra les define nécessaires à l'adaptation du logiciel au système de compilation. Il contient en fait une série de définitions du type :

```
/* Define to 1 if you have the <arpa/inet.h> header file. */
#define HAVE_ARPA_INET_H 1
```

Le fichier configure.ac continue ensuite par une série de vérifications standard :

```
# Checks for programs.
AC_PROG_CC
AM_PROG_LIBTOOL
AC_PROG_INSTALL
# Checks for endianess
AC_C_BIGENDIAN()
```

Les trois premières lignes demandent de vérifier la présence du compilateur, de l'utilitaire libtool et du programme "install". La dernière permet de vérifier l'endianness du système de compilation.
La suite du fichier déclenche les tests de divers composants ainsi :

```
AC_CHECK_LIB([pthread], [pthread_mutex_init],, check_pthread=no)
```

Cette ligne demande la vérification de la présence de la bibliothèque phtread en s'assurant de la présence de la fonction pthread_mutex_init. Le test généré par cette simple macro est conséquent puisqu'il fait environ 70 lignes. L'essentiel est la compilation d'un programme minimaliste vérifiant que la fonction pthread_mutex_init est utilisable :

```
#ifdef __cplusplus
extern "C"
#endif
char pthread_mutex_init ();
int
main ()
{
return pthread_mutex_init ();
  ;
  return 0;
}
```

En cas de succès, il ajoute au fichier config.h la définition :

```
#define HAVE_LIBPTHREAD 1
```

et positionne la variable LIBS qui sera utilisée dans les Makefile :

```
LIBS="-lpthread $LIBS"
```

Certains logiciels fournissent des macros qui permettent leur paramétrage en un seul appel. Ainsi pour la bibliothèque Glib :

```
AM_PATH_GLIB_2_0(2.4.0, , check_glib=no,[gthread gmodule])
```

Ces macros ainsi que les macros standard facilitent l'édition du fichier configure.ac. Une fois les dépendances vérifiées, il faut indiquer quels sont les fichiers .in à instancier. Ceci se fait en utilisant AC_CONFIG_FILES :

```
AC_CONFIG_FILES([Makefile
        doc/Makefile
        src/Makefile
        src/include/Makefile
        src/nuauth/Makefile
        ...
        ])
```

Composants optionnels

Les autotools permettent de gérer facilement la compilation de composants optionnels. NuFW utilise un système de modules. Chaque module est déclaré au niveau du configure.ac par une ligne du type suivant :

```
AC_ARG_WITH(ldap, [AC_HELP_STRING(-with-ldap, \
                        Support LDAP directory for acl lookup) \
                ], \
                ldap=$withval, \
                ldap="")
```

L'utilisateur peut alors déclarer qu'il souhaite utiliser ldap en lançant :

```
./configure --without-ldap
```

Il récupérera aussi dans l'aide obtenue en lançant ./configure --help :

```
--with-ldap             Support LDAP directory for users and acl lookup
```

L'appel à AC_ARG_WITH définit principalement une variable ldap qui peut ensuite être utilisée pour lancer des tests :

```
if test "${ldap}" = "yes"; then
        AC_CHECK_LIB([ldap],[ldap_simple_bind_s], \
                        AC_DEFINE([HAVE_LIBRARY_LDAP],[1],[ldap lib flag]),
check_ldap=no)
fi
```

Ce test permet de vérifier la présence de la bibliothèque LDAP uniquement si son utilisation est demandée. L'utilisation de ces arguments est très souvent couplée avec la déclaration conditionnelle, ils sont utilisés dans les Makefile.am pour sauter des étapes de la compilation, rendues inutiles par l'inactivation d'une fonctionnalité. Cela se fait comme suit :

```
AM_CONDITIONAL(USE_LDAP, test x$ldap = xyes)
```

Il est alors possible d'utiliser la variable USE_LDAP dans les Makefile.am.

Makefile.am

Principes

Le Makefile.am contient généralement un nombre d'instructions très limité. Il s'agit principalement de la déclaration de la cible de compilation (le programme, la bibliothèque), des sources et optionnellement des dépendances. Ces dernières sont en effet ajoutées automatiquement par les autotools.

Mise en oeuvre

Le Makefile.am permettant de générer le binaire nutcpc est le suivant :

```
if BUILD_NUTCPC

bin_PROGRAMS = nutcpc

EXTRA_DIST = valgrind.sh valgrind.supp

# nutcpc
nutcpc_SOURCES = nutcpc.c
nutcpc_CPPFLAGS = -I$(top_srcdir)/src/include/ -I$(top_srcdir)/src/
clients/lib/
nutcpc_LDFLAGS = -L$(top_builddir)/src/clients/lib/
nutcpc_LDADD = -lnuclient

check:
	$(top_builddir)/src/clients/nutcpc/nutcpc -V

endif
```

Le code n'est compilé que si la conditionnelle BUILD_NUTCPC définie dans configure.ac est vraie. L'instruction principale est bin_PROGRAMS qui indique que le programme cible est nutcpc et qu'il s'agit d'un programme qu'il faut installer dans un répertoire bin. Le programme sera donc installé sous le *prefix défini,* dans le répertoire nommé bin. Sans option passée au script configure, ce sera donc dans /usr/local/bin/.

Une fois bin_PROGRAMS définie à nutcpc, les variables nutcpc_ se réfèrent alors à la compilation de ce programme :

- nutcpc_SOURCES indique quelles sont les sources du logiciel
- nutcpc_CPPFLAGS contient les chemins vers les include nécessaires à la compilation
- nutcpc_LDFLAGS ajoute un chemin de recherche de bibliothèque pour la phase de lien
- nutcpc_LDADD indique à gcc de lier le programme avec la bibliothèque libnuclient

La compilation est alors faite suivant les règles standard en utilisant ses valeurs. Le lancement de make dans le répertoire de nutcpc lance notamment la commande :

```
gcc -g -O2 -ansi -Wall -Wextra -Wno-unused-parameter -O2 -o .libs/
nutcpc nutcpc-nutcpc.o /usr/lib/libgcrypt.so -L/home/eric/nufw-svn
/trunk/nufw/src/clients/lib /home/eric/nufw-svn/trunk/nufw/src/
clients/lib/.libs/libnuclient.so -lpthread -Wl,-rpath -Wl,/opt/nufw//lib
```

Ce fichier Makefile.am contient une cible personnalisée check qui peut être appelé en lançant make check.

Fonctions avancées

make dist

Outre les cibles classiques comme *clean* ou *install,* les Makefile générés par les autotools contiennent des cibles moins connues visant à faciliter la vie du développeur. Parmi elles, on trouve *make dist* qui génère une archive à partir des sources.

Cette cible ne met dans l'archive que ce qu'elle sait indispensable (les sources par exemple) ou ce qu'on lui demande. Il est donc nécessaire de spécifier les fichiers que l'on souhaite incorporer dans l'archive si ils ne sont pas utilisés lors de la compilation. Par exemple, dans le Makefile.am de nutcpc, l'inclusion des fichiers liés à *valgrind* est demandée de manière explicite :

```
EXTRA_DIST = valgrind.sh valgrind.supp
```

Il est bon de noter qu'il faut faire de même avec les fichiers include.

Il est possible de générer une archive au format bzip2 en utilisant :

```
make dist-bzip2
```

make distcheck

Cette cible est plus évoluée puisqu'elle réalise une compilation des sources dans un répertoire séparé des sources. Cela permet de tester la validité d'une archive réalisée avec make dist. La mise en place de la conformité avec make distcheck suppose d'avoir un fichier Makefile.am bien paramétré. Il faut en effet aller chercher les sources dans le répertoire des sources et les objets compilés dans le répertoire de compilation. Ceci a amené à écrire dans le Makefile.am de nutcpc :

```
nutcpc_CPPFLAGS = -I$(top_srcdir)/src/include/ \
                  -I$(top_srcdir)/src/clients/lib/
nutcpc_LDFLAGS = -L$(top_builddir)/src/clients/lib/
```

Problèmes des autotools

Le problème principal des autotools réside dans sa syntaxe dont la puissance n'a d'égal que la difficulté. Cette syntaxe est d'autant plus difficile à maîtriser que la complexité des mécanismes d'automatisation est quasi impossible à appréhender dans sa totalité. Cela rend l'écriture des fichiers *configure.ac* et *Makefile.am* très difficile et certains développeurs ont surnommé les outils "auto-hell".

Les autotools souffrent aussi parfois de problèmes de compatibilité ascendante. Des souplesses permises dans certaines versions sont absentes d'autres versions ce qui peut rendre une compilation impossible. Cela est d'autant plus préjudiciable que la portabilité visée n'est alors pas au rendez-vous puisque la compilation sur une distribution plus récente que celle du développeur n'est parfois plus possible.

Conclusion

L'utilisation des autotools est une tâche ardue mais ces outils permettent de gérer efficacement la compilation de logiciels complexes. Standard de fait depuis des années pour la compilation de logiciels en C ou en C++, il est difficile de s'en passer et seul cmake semble un candidat sérieux au remplacement comme le prouve le passage du projet KDE des autotools à cmake pour KDE 4. L'avenir nous dira si la vague de migration vers cmake sera suffisante pour en faire un nouveau standard de fait.

Merci à **Pierre Chifflier** pour sa relecture attentive.

■ **Eric Leblond** <*eric@inl.fr*> - *INL SARL*

Le Build en .NET

Assembler pour construire du logiciel n'est pas nouveau. Par contre, l'intérêt d'un processus de construction automatique a été renforcé par la demande croissante de projets itératifs devant évoluer sans régression et rapidement. Nous présentons ici les principaux composants du build en .NET ainsi que son outillage.

Loin de se limiter à la simple compilation de l'application sur le poste développeur, le build s'est étendu : il est maintenant automatisé et inclut a minima l'exécution des tests unitaires (1). Afin de le distinguer de son ancêtre, nous utilisons les termes de Build, avec un B majuscule, ou encore de **" pipeline de Build "**.

Représentons provisoirement et très schématiquement les étapes du pipeline (Fig.1).

Travailler suivant ces étapes appelle plusieurs questions :

- Quel est / quel peut être le déclencheur de ce pipeline ?
- Chaque étape du pipeline peut retourner une erreur et mettre en échec le pipeline tout entier : Comment identifier l'erreur ? Comment notifier l'équipe ? Que faire en cas d'échec ?
- Que produit-il en sortie : un logiciel ? un rapport ? sous quel format ? pour qui ? pour quoi faire ?
- Quel est l'outillage associé à chacune de ces étapes ?

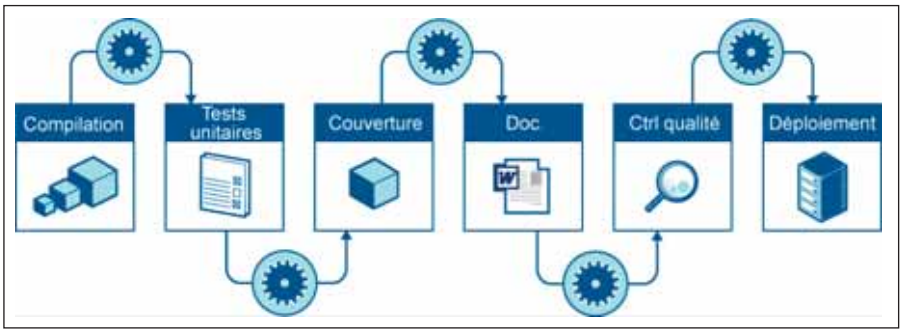


Fig.1 : Les grandes étapes du pipeline de Build

Le pipeline de Build en détail

Le processus de Build met en œuvre les composants suivants (Fig.2).

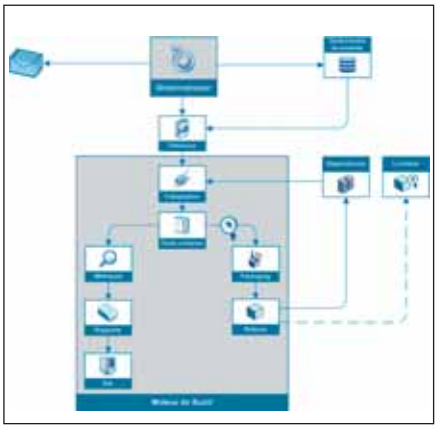


Fig.2 : Les composants détaillés du Build

Le gestionnaire de sources est chargé de centraliser et d'historiser le code source. Celui-ci doit pouvoir :

- Marquer une version des sources, en l'occurrence celle qui est buildée, via un tag, ou une étiquette ;
- Isoler différentes versions du logiciel : les branches sont le mécanisme le plus fréquent pour cela ;
- Effectuer des opérations transactionnelles sur le code source : l'ensemble des fichiers soumis au gestionnaire est rollbacké en cas de conflit sur l'un d'eux ;
- Travailler de manière collaborative : l'édition sans verrouillage, les outils de visualisation des différences et de fusion sont particulièrement utiles dans le cadre de la pratique d'intégration continue ;
- Etre intégré confortablement à l'IDE: le gestionnaire de sources est manipulé de manière fréquente par les développeurs, il faut donc qu'ils l'aient " sous la main ".

Team Source, le module de gestion de source de Team System, et Subversion répondent par exemple à ces critères.

L'ordonnanceur lance le Build sur la plateforme d'intégration. Le déclenchement peut intervenir :

- Sur événement (triggering) : l'ordonnanceur observe le gestionnaire de sources afin de détecter les mises à jour. Un délai avant le démarrage de la construction est généralement paramétré afin de ne pas effectuer deux compilations lors de mises à jour rapprochées. Ce mode est utilisé dans le cadre de l'intégration continue ;
- A échéance fixe : c'est le cas de la construction nocturne (nighty Build) par exemple ;
- Sur demande : un membre de l'équipe peut demander explicitement l'exécution d'un Build.

C'est l'ordonnanceur qui effectue la première étape du Build. Elle consiste à récupérer la version la plus à jour des sources auprès du gestionnaire de sources (checkout ou update).

L'ordonnanceur offre généralement un tableau de bord des constructions en cours et de l'historique des constructions, la possibilité de consulter les journaux d'événements des constructions. Il est chargé d'avertir par mail (ou par autre moyen) les développeurs et personnes intéressées par le résultat du Build à l'issue du processus.

A titre d'exemple, CruiseControl .NET (CC.NET) est l'ordonnanceur .NET Open Source le plus répandu.

Le moteur de Build réalise la construction étendue de l'application :

1. Il lance la compilation du code source, en utilisant éventuellement des dépendances externes (issues d'un éditeur, de la communauté open source ou d'une autre équipe projet) ;
2. Il exécute ensuite les tests unitaires. C'est généralement à l'issue de cette étape que le Build est qualifié (" réussi " ou " cassé ") ;
3. Il calcule alors des métriques sur l'applica-

(1) On se référera à l'article de référence sur l'intégration continue rédigé par Martin Fowler http://martinfowler.com/articles/continuousIntegration.html

INDISPENSABLE !

Demandez-le d'urgence
à votre marchand de journaux !

tion (couverture des tests unitaires, respect des règles de développements, analyse des similitudes,...). Elles sont destinées à estimer la qualité du code compilé ;

4. Le déroulement du Build, les métriques calculées dans l'étape précédente, et le résultat du Build (" réussi " ou " cassé ") sont consignés dans des rapports ;
5. En cas de Build réussi, l'application est packagée afin de fournir un livrable (Release).

En l'occurrence, quelques standards du marché sont à signaler :

- MSBuild, moteur de Build gratuit, inclus dans le framework .NET et extensible ;
- FXCOP, outil de vérification des normes de développements et des bonnes pratiques, fourni gratuitement par Microsoft ;
- NUnit, framework Open Source de tests unitaires.

Le **site projet** agrège et publie les rapports et métriques issus des différentes étapes du processus de Build :

- La documentation du code ;
- La couverture des tests unitaires ;
- Les infractions aux règles de développement (codage / nommage) ;
- Les différentes métriques de qualité (complexité et taille du code, copier/coller, ...).

Ce site permet aux différents acteurs de l'équipe de connaître l'état du projet.

Dans le cas de Team System, c'est un portail SharePoint qui tiendra ce rôle. Il expose les documents projet.

Qu'est ce qu'un Build cassé ?

Un Build est considéré comme " cassé " lorsque l'une des étapes du Build retourne une erreur. Les cas de Build cassé les plus courants sont :

- **Le code source ne compile pas.** Cette erreur est généralement due au fait que les développeurs n'ont pas mis à jour leur version locale des sources et résolu les conflits sur leur poste avant de soumettre leurs modifications. La version actuelle du code dans le référentiel n'est donc pas cohérente (fichiers en conflit, différences d'API,...) ;
- **Les tests unitaires ne passent pas.** Ce cas se produit lorsqu'un développeur n'a pas exécuté tout ou partie des tests unitaires avant de mettre ses sources à disposition.

Quelles que soient les règles définies, il convient de s'assurer des points suivants :

- Les développeurs doivent avoir les moyens de s'assurer qu'ils ne casseront pas le Build avant de mettre à disposition leurs modifications dans le gestionnaire de sources ;
- Le Build ne doit pas casser " pour un rien " (nom d'attribut mal formé par exemple), sous peine que l'équipe se désintéresse du résultat du Build ;
- La durée du Build doit rester courte si l'on pratique l'intégration continue. On envisagera le cas échéant d'avoir plusieurs Build :
  - Un Build rapide vérifiant la compilation et les tests unitaires. Ce Build sera exécuté à chaque commit ;
  - Un Build plus long, effectuant des vérifications complémentaires et générant le site projet, qui sera effectué la nuit.

Quelles actions entreprendre à l'issue du Build ?

Lorsque le **Build est cassé,** l'ordonnanceur doit notifier les membres de l'équipe de ce statut anormal (par mail généralement). La résolution de ce problème devient alors la priorité de l'équipe.

En cas de succès, l'ordonnanceur est généralement chargé de marquer la version des sources (et des fichiers de configuration) utilisée. L'étiquette dont on se sert est généralement le numéro du Build. Elle permet à tout moment de reproduire un Build donné. Une livraison est une étape supplémentaire du Build. Après le marquage des sources, elle construit le livrable (MSI, ZIP,...), puis le dépose ou le déploie sur une plate-forme de test.

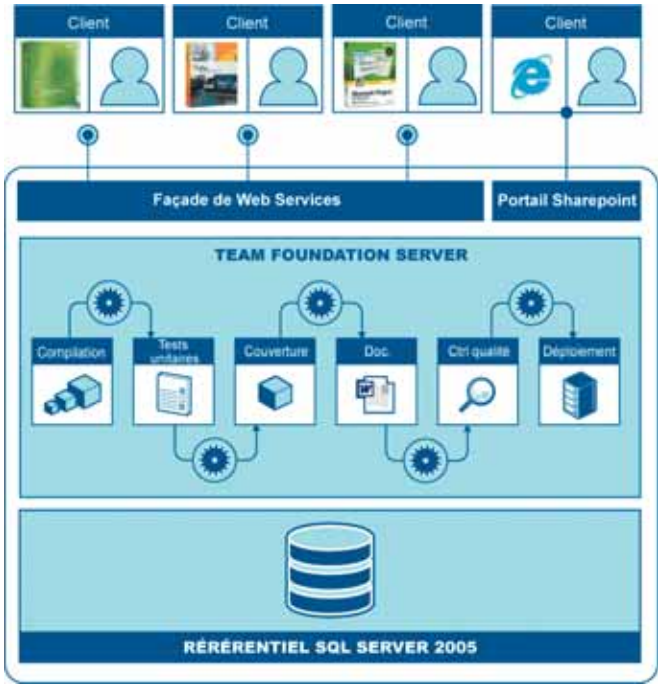


Fig.3 : Pattern " outil intégré " dans Team System

Patterns d'intégration et outils de Build

Nous venons de présenter les étapes et composants du Build. Concrètement, quels sont les outils de Build en .NET ? Comment fonctionnent-ils ?

A la lueur de ce tour du propriétaire des composants du Build, il apparaît qu'un outil de Build procédera lui-même à **l'intégration** de ses différents composants, qui sont : le gestionnaire de source, l'ordonnanceur de Build, le moteur de build, et les sites projet.

En la matière, il existe deux grandes familles d'outils .NET, qui participent de deux patterns de l'intégration.

1. Dans le premier pattern, l'outil de Build est conçu comme un intégrateur de composants du Build : il est relativement léger, et agrège différents composants du Build qui lui sont externes, il les ordonnance et les orchestre dans le pipeline de Build. L'intégration de ces composants externes est souvent effectuée à l'aide du pattern plug-in.
2. Dans le second pattern, l'outil de Build est conçu comme un outil intégré (de composants internes de Build) : il est plus lourd et implémente en interne les composants du Build.

" Intégrateur d'outils " " outil intégré "

Microsoft Team System repose entièrement sur le pattern " outil intégré " : l'outil contient un référentiel centralisé de données (une base SQL Server 2005 avec des cubes OLAP), et un serveur Team Foundation Server. Le serveur TFS implémente les différents composants du pipeline de Build, et présente l'exécution des Build dans un portail SharePoint (Fig.3).

Ce pattern est aussi mis en œuvre dans la

(2) http://ccnet.thoughtworks.com et http://draconet.sourceforge.net
(3) Liste complète : http://confluence.public.thoughtworks.org/display/CCNET/Source+Control+Blocks

29, 30 et **31** janvier 2008

CNIT – Paris La Défense

200 exposants pour trouver des solutions à vos problématiques

Un programme de conférences sans précédent :

- ↘ 6 Keynotes d'ouverture
- ↘ 24 Formations / Tutoriels en accès payant
- ↘ 15 Débats / Tables Rondes en accès libre

10ème édition

Exposition - Conférences

OPENSOURCE solutions Linux

Le Rendez-Vous Annuel Européen dédié à Linux et aux logiciels libres

www.solutionslinux.fr

Vos contacts :

- ↘ Pour exposer : Carole Jardon - cjardon@tarsus.fr - Tél. : 01 41 18 60 52
- ↘ Pour vous inscrire aux Formations / Tutoriels : Lucie Foucher - lfoucher@tarsus.fr - Tél. : 01 41 18 86 42
- ↘ Pour demander votre badge d'accès au salon et réserver vos places aux conférences en accès libre : www.solutionslinux.fr

Un événement :

TARSUS FRANCE

suite Rational Build Forge d'IBM, ainsi que dans Borland Gauntlet.

Le pattern " intégrateur d'outils ", lui, est très utilisé dans le monde Open Source, à l'instar de **CruiseControl.NET** (CC.NET), ou de Draco.NET (2). Il est aussi présent dans des outils commerciaux tels que : Automated Build System, Final Builder, ParaBuild, Visual Build Professional, ou encore Visual Make.

Prenons l'exemple de CC.NET : dans son principe, CC.NET est un intégrateur de composants de Build, suivant un pattern plug-in :

- Il est compatible avec la plupart des gestionnaires de code source : Clear Case, CVS, Perforce, PVCS, Subversion, Synergy, Team Source, Visual Source Safe, …
- Il s'appuie sur NAnt, MSBuild pour la compilation ;
- Il intègre NUnit ou csUnit et NCover pour les tests unitaires instrumentés, ainsi que Fitnesse pour les tests de recette ;
- Il sait lancer VIL ou FXCop pour les contrôles qualité .NET ;
- Il notifie le résultat du Build par email.

CC.NET affiche également le résultat de chaque Build au sein d'un site portail.

Le tableau ci-dessous présente succinctement quelques outils intégrés ou intégrateurs de composants de Build et le pattern d'intégration qu'ils implémentent.

Conclusion

Aujourd'hui, on constate que le Build s'est standardisé autour des composants et des pratiques que nous avons présentés, et que l'on retrouve dans l'intégration continue et les usines de développement .NET.

Les outils de Build modernes implémentent les composants du pipeline de Build suivant deux patterns d'intégration différents : pattern " outil intégré " et pattern " intégration d'outils ". De ce point de vue, le pattern " outil intégré ", qui connaît toutes les données du Build, permet d'offrir facilement des rapports plus complets et de présenter l'évolution des métriques : évolution du nombre de tests unitaires OK/KO, évolution du taux de couverture, évolution des infractions aux règles de développement…

Mieux, lorsque l'outil intègre un gestionnaire de demandes, les rapports peuvent relier les évolutions de code, les métriques qualité avec les demandes de développement, voire le planning projet.

Les outils intégrés nécessitent souvent moins d'effort d'intégration de la part de l'équipe. Ceci n'est pas négligeable car cela induit un délai et des compétences spécifiques dans la mise en œuvre et la mise à jour d'une usine logicielle .NET.

En revanche, les outils intégrés souffrent souvent d'un manque d'ouverture à des composants techniques tiers, ce qui est loin d'être anodin lorsque l'on possède des outils existants.

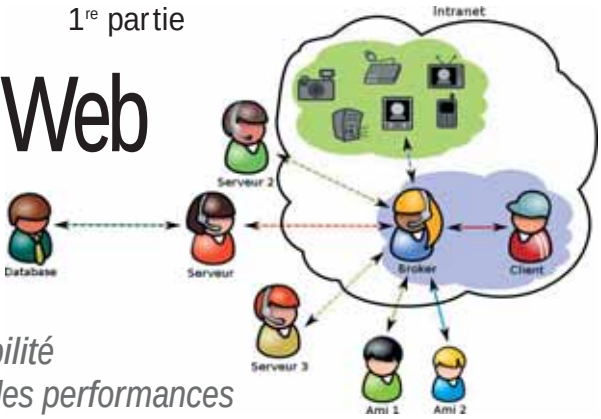
Des remarques ? Des idées ? Venez en discuter sur notre blog ! http://blog.octo.com

■ **Messaoud Oubechou** - meo@octo.com
■ **Frédéric Schäfer** - fch@octo.com
Architectes Logiciels – Responsables du Centre de Compétences .NET
OCTO Technology – www.octo.com

	Pattern d'intégration	Licence Commercial / OSS	Gestionnaire de source supporté	Ordonnanceur	Moteur de Build / Test / Qualité	Site Projet / Notification
Automated Build Studio 1.4 de Automated QA	Intégrateur d'outils	Commercial	StarTeam, CVS, PVCS, VSS, Team Source, Perforce, ClearCase, SourceGear, SVN	Programmé Sur événement déclencheur (check-in, mise à jour de fichier) A la demande par interface web	Compilateurs .NET, C++ (Intel, Borland, GNU), ASP.NET, NAnt, MSBuild, / A. QA Test, JUnit, NUnit, MSTest	Web, technique et orienté Build unitaire (pas de rapport d'évolution en standard) / Net send, MSN, ICQ, mail
CruiseControl.NET 1.3	Intégrateur d'outils	Open Source	CVS, SVN, VSS, Perforce, ClearCase, PVCS, StarTeam, Telelogic, Seapine	Intervalle, programmé, intégration continue	NAnt, MSBuild, Visual Studio, Final Builder / Fitnesse, NUnit, NCover / VIL, FxCop	Web, orienté Build unitaire (pas de rapport d'évolution en standard) / icon tray
Draco.NET 1.6.4	Intégrateur d'outils	Open Source	CVS, VSS, SVN, PVCS, Vault	Service Windows en mode poll et intégration continue	MSBuild, NAnt, VS2005	Site web orienté Build / Notification par email
Final Builder Server de VSoft Technologies	Intégrateur d'outils	Commercial	ClearCase, Source Force, CVS, MKS, PVCS, QVCS, StarTeam, SourceGear, Seapine, SVN, Vault	A la demande, intervalle, intégration continue	C++, .NET, Borland C#, Delphi.NET / Nant, MSBuild / NUnit, MSTest, TestComplete, AQTest	Web site / email, RSS
ParaBuild 3.1 de ViewTier Systems	Intégrateur d'outils	Commercial	Perforce, Subversion, CVS, ClearCase, Serena, StarTeam, Vault, MKS, VSS	A la demande, intervalle, intégration continue	NAnt, MSBuild, VB, MSDEV, make, nmake	Site Web orienté build / email, rss, system tray, IM
Team City 2.1.1 de JetBrains	Intégrateur d'outils	Commercial	ClearCase, CVS, Perforce, SVN, Team Source, VSS	A la demande, intervalle, intégration continue	MSBuild, NAnt	Site web orienté build /
Team System / Team Foundation Server 1.0 de Microsoft	Outil intégré	Commercial	Team Source	A la demande, intervalle, intégration continue	Team Build / MSBuild	Site SharePoint liant Work Items, MS Project lors du Build / email, rss
IBM Rational Build Forge	Outil intégré	Commercial	ClearCase + ClearQuest + CVS, Perforce, StarTeam, VSS, Subversion	Programmé, à la demande, intégration continue	JAVA/.NET, intégration avec Eclipse et VS2005	Web / RSS, mail, tray
Gauntlet de Borland	Outil intégré	Commercial	Borland Star Team	Programmé, à la demande, intégration continue	Intégration native avec Perfect Build Borland Silk Performer Borland Silk Test Manager NUnit	Site et rapports Web, orientés build

Hop, un langage de programmation pour le Web

1re partie



L'arrivée d'Ajax a permis la naissance d'applications Web réactives et pourvues d'interfaces de qualité. Ajax a été rendu possible par la conjonction de plusieurs facteurs: Javascript, CSS2, normalisation du DOM et disponibilité de XMLHttpRequest, sans, bien sûr, oublier l'accroissement des performances des navigateurs et des machines les hébergeant. De nombreuses applications Web telles Google Mail, Flickr, Deezer ou YouTube illustrent ces avancées.

Malheureusement, ces mêmes facteurs imposent, aujourd'hui, de connaître plusieurs langages de programmation ou de formatage (côté serveur ou côté client), de nombreuses bibliothèques, une multitude de styles, de conventions, le tout assorti d'interactions étranges et d'un déploiement de plus en plus délicat. L'écriture d'applications Web de qualité reste intrinsèquement difficile, notamment parce qu'il faut faire coopérer des programmes s'exécutant sur deux lieux distincts: le serveur et le client.

HOP est une nouvelle plate-forme de développement qui a pour but de simplifier l'écriture et la maintenance d'applications Web interactives. Elle permet de réaliser simplement des agendas Web, des blogs, des jeux en réseaux, des outils multimedia sur le Web, etc. Ces applications sont disponibles sur le site de HOP sous la forme des weblets. Dans ce premier article, nous allons présenter HOP, son installation et quelques courts exemples montrant une partie de son extraordinaire potentiel. Un prochain article traitera d'une application non triviale (une galerie de photos) en moins de 50 lignes.

HOP c'est quoi ?

Pour assurer un minimum de sécurité aux ordinateurs utilisés pour surfer, le Web restreint fortement les actions qu'un navigateur Web peut déclencher. En particulier, les programmes s'exécutant dans le navigateur ne peuvent accéder aux ressources physiques de l'ordinateur qui a lancé le navigateur et ils ne peuvent ouvrir des connexions réseaux que vers leur serveur d'origine. Ces mesures, bien que nécessaires, rendent particulièrement complexe l'écriture d'applications qui utilisent les ressources personnelles des utilisateurs. Par exemple, comment écrire un programme Web de visualisation des photographies de votre disque dur si le navigateur Web n'est pas autorisé à lire les fichiers contenant les dites photographies ?

La méthode généralement employée pour contourner ces restrictions imposées par les règles élémentaires de sécurité consiste à étendre les fonctionnalités autorisées du navigateur au moyen de greffons (ou plug-in) spécialisés. Par exemple, le plug-in Flash permet de contrôler la carte son d'un ordinateur et de jouer des vidéos. Le plug-in Google Gear permet d'assurer un mode hors-ligne des applications Google. HOP a adopté une solution différente. Il ne modifie pas les navigateurs mais il repose sur un petit serveur Web léger et malléable qui peut facilement être installé sur tous les ordinateurs classiques ainsi que sur la plupart des périphériques usuels (modems/routeurs, NAS, PDA, etc.). Il peut accéder aux ressources disponibles sur les réseaux locaux, par exemple, pour contrôler des appareils multimédia, mais aussi sur les réseaux globaux, pour, par exemple agréger des données (site mashup). Comme ces serveurs occupent peu de mémoire, il est envisageable d'en lancer plusieurs simultanément sur une même machine, afin que chacun d'entre eux, muni des droits appropriés, prenne en charge une et une seule application. Un premier serveur HOP pourrait, par exemple, exécuter une application de jeu, alors qu'un second pourrait jouer de la musique et qu'un troisième permettrait de lire ses courriels. Ces petits serveurs peuvent être considérés comme des relais (ou proxy) intelligents. Afin d'éviter toute confusion, nous emploierons ici le terme de courtier Web (ou broker).

HOP repose sur un nouveau langage de programmation qui permet de couvrir, dans un formalisme unique, la totalité des besoins d'une application Web. Un programme HOP décrit donc à la fois le comportement du courtier et le comportement du client. Des annotations syntaxiques dans le code permettent de distinguer les expressions dites *courtiers,* c'est-à-dire évaluées par le courtier, des expressions clients évaluées dans le navigateur du client. Mais c'est là la grande force de HOP, les expressions courtiers peuvent référencer les variables et fonctions du code client et vice versa. Le langage HOP repose sur une syntaxe proche de celle d'HTML et une sémantique provenant en droite ligne des langages fonctionnels.

Installation de HOP

Avant d'étudier un premier programme HOP, installons la plate-forme. HOP est un logiciel libre distribué sous licence GPL. Ses sources et des versions pré-compilées sont disponibles sur le site Web de HOP (http://hop.inria.fr). Sous MacOS X et Windows, il est recommandé d'utiliser la version pré-compilée en code octets JVM. Elle est moins performante mais plus portable. Ainsi, pour pouvoir utiliser HOP, sous Windows et sous Mac OS, il faut lancer une JVM (au moins 1.4) avec une commande comme java -jar hop-1.8.0.jar, un simple double clic pourrait aussi suffire. Sous Windows, il se peut que vous soyez prévenu que vous passez en zone Intranet.

Par défaut, HOP attend ses clients sur le port 8080 mais ceci peut être

changé soit au moyen d'une option utilisée sur la ligne de commande (hop --help affiche la totalité des options reconnues), soit au moyen d'un fichier de configuration.
La version JVM peut être utilisée de la même manière sous Linux. Toutefois, sur ce système d'exploitation, il est recommandé de reconstruire une version native de HOP à partir des sources (et de la plus récente version du compilateur Bigloo (au moins 3.0c) que l'on peut trouver sur le site de HOP) avec un classique ./configure && make && make install.
Une fois HOP installé et lancé, il vous faut le sécuriser vis-à-vis des intrus. Orientez votre navigateur (Firefox, Safari et WebKit sont fortement conseillés ; Internet Explorer 7 peut aussi être utilisé) en http://localhost:8080/hop/wizard. Dans ce petit panneau de configuration, indiquez au minimum le mot de passe protégeant le compte d'administration ainsi que le nom et mot de passe que vous emploierez comme utilisateur standard. Sauvegardez avec le bouton de soumission en bas de page, notez éventuellement le nom du fichier où est configuré HOP en cas d'oubli de mot de passe. Arrêtez puis relancez HOP: tous les programmes pré-installés sont désormais accessibles depuis l'URL http://localhost:8080/hop et, en particulier, le programme permettant l'arrêt de HOP (shutdown) ainsi que la documentation du système. Cette documentation est une application HOP interactive: les exemples peuvent être exécutés et modifiés directement depuis le navigateur!

Les premiers pas

Dans cette section, afin d'introduire les principaux concepts de HOP, nous allons montrer comment programmer le sempiternel Hello World!. Les fichiers sources de HOP sont suffixés par .hop. La syntaxe de HOP diffère légèrement de celle de HTML. Les commentaires débutent par le caractère ; et s'achèvent avec la fin de la ligne. Les balises ouvrantes sont précédées d'une parenthèse ouvrante et les balises fermantes sont remplacées par une simple parenthèse fermante. Voyons le contenu de notre premier code source, le fichier hello.hop :

```
1: ;; fichier "hello.hop"
2: (define-service (hello)
3:    (<HTML>
4:       (<BODY> "Hello World!")))
```

Pour charger un programme dans HOP, la façon la plus simple est de lancer HOP en indiquant le fichier à charger sur la ligne de commande. Ainsi écrira-t-on hop hello.hop pour les versions natives et java - jar hop.jar hello.hop pour les version pré-compilées JVM. Il est alors possible d'exécuter notre programme en pointant notre navigateur sur l'URL http://localhost:8080/hop/hello. L'URL /hop/hello a été automatiquement créée par HOP lorsque ce dernier a traité la déclaration (define-service (hello) ...) (ligne 2). L'URL a été produite en concaténant le préfixe hop/ commun à toutes les URLs produites par HOP et le nom du service (ici hello). Les services de HOP jouent le rôle habituellement dévolu aux fonctions dans les langages de programmations classiques. Un service HOP peut recevoir des paramètres, ainsi, nous pouvons écrire la seconde version de notre programme qui affiche cette fois un message variable.

```
1: ;; fichier "hello2.hop"
2: (define-service (hello2 who)
3:    (<HTML>
```


```
4:       (<BODY> "Hello "
5:          (<SPAN> :style "color: red"
6:             who))))
```

Pour exécuter ce nouveau programme, il suffit de redémarrer HOP avec le fichier hello2.hop. Cette fois, comme le service prend un paramètre, il faut utiliser des URL comme http://localhost:8080/hop/hello2?who=Programmez.
Rendons maintenant nos programmes réactifs. Au lieu d'afficher systématiquement la valeur du paramètre who, nous n'afficherons que trois points d'interrogation. La valeur du paramètre who n'apparaîtra que lorsque l'utilisateur cliquera sur ces points d'interrogation. En HOP les parties d'un programme Web destinées à être exécutées par le navigateur ne sont pas écrites en Javascript : elles sont aussi écrites en HOP ! Le caractère ~ préfixe les expressions à exécuter dans le navigateur. À l'intérieur de ces expressions, il est possible d'utiliser les valeurs d'expressions produites par le courtier au moyen de l'échappement inverse qu'introduit le caractère $. Ainsi lorsque l'utilisateur cliquera sur les

points d'interrogation, le navigateur affichera une boîte de dialogue dont le contenu révèlera la valeur de la variable *who* créée par le courtier. Voici la troisième version:

```
1: ;; fichier "hello3.hop"
2: (define-service (hello3 who)
3:    (<HTML>
4:       (<BODY> "Hello "
5:          (<SPAN> :onclick ~(alert $who)
6:             :style "cursor: pointer; text-decoration: underline"
7:             "???"))))
```

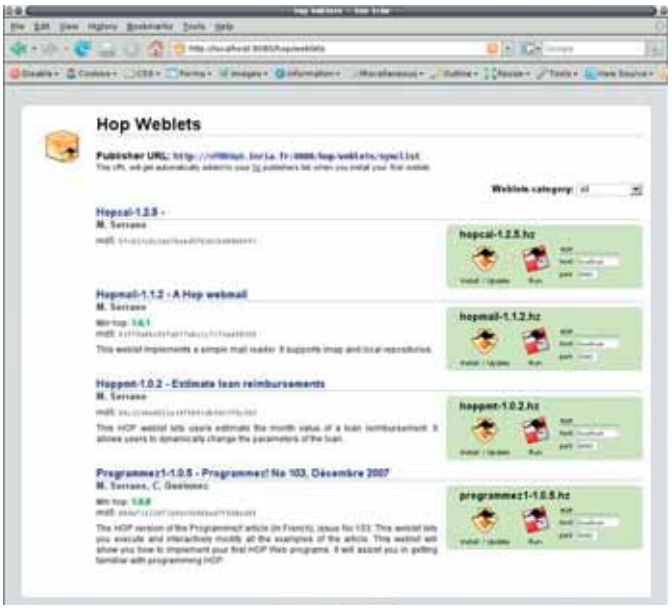
Référencer les variables du courtier depuis le code client simplifie la programmation DHTML. Pour l'illustrer, nous allons faire évoluer notre précédent exemple afin de remplacer le paramètre who du service hello3 par une zone de saisie de texte. À chaque fois que le contenu de cette zone sera modifié, une liste non-ordonnée contenant l'historique des saisies sera mise à jour et affichée dans la page Web. La variable locale *ul* est initialisée avec une liste vide (ligne 3, la construction *let* introduit un bloc local introduisant la variable locale *ul*). Elle est utilisée par le code courtier (ligne 8) pour insérer la liste dans la page Web envoyée au client. Elle est enfin utilisée par le code client (ligne 7) pour les mises à jour de la liste.

```
1: ;; fichier "hello4.hop"
2: (define-service (hello4)
3:    (let ((ul (<UL>)))
4:       (<HTML>
5:          (<BODY> "Hello "
6:             (<INPUT> :type 'text :size 5 :value "???"
7:                :onchange ~(dom-append-child! $ul (<LI> this.value)))
8:             (<DIV> "Welcome: " ul)))))
```

Pour terminer cette première série d'exemples (que vous pouvez aussi expérimenter en parallèle avec deux navigateurs différents), imaginons une dernière modification de notre programme. Au lieu d'accumuler directement dans le navigateur, les valeurs saisies par l'utilisateur, nous allons les stocker, ainsi que leur date d'enregistrement, dans une base de données située sur le courtier. La page Web sera dynamiquement modifiée après chaque nouvel enregistrement afin de refléter l'état courant de la base.

```
 1: ;; fichier "hello5.hop"
 2: (define-service (hello5)
 3:    (let ((table (<TABLE>)))
 4:       (<HTML>
 5:          (<BODY> "Hello "
 6:             (<INPUT> :type 'text :size 5 :value "???"
 7:                :onchange ~(with-hop ($dbase-add! this.value)
 8:                             (lambda (rows)
 9:                                (let ((tbody (map (lambda (r)
10:                                                    (<TR> (map <TD> r)))
11:                                                  rows)))
12:                                   (innerHTML-set! $table tbody)))))
13:             (<DIV> "Database: " table)))))
14:
15: (define dbase '())
16:
17: (define-service (dbase-add! entry)
18:    (set! dbase (cons (list entry (date)) dbase))
19:    dbase)
```

En plus du service principal hello5, ce programme fait intervenir un service supplémentaire dbase-add! (ligne 17) qui ajoute des entrées dans la base de données. Contrairement à l'exemple précédent, lorsqu'une nouvelle valeur a été saisie dans la zone de texte, la page HTML n'est pas directement modifiée. Le code client invoque le service dbase-add! qui est situé sur le courtier (ligne 7), à l'aide de la construction with-hop. Le courtier ajoute alors dans sa base (ici une simple liste mais HOP contient aussi SQLite) la nouvelle entrée (ligne 18) et renvoie au client, une liste contenant toutes les entrées stockées. Cette liste est fournie au second argument de with-hop qui la convertit, dans le navigateur, en une table HTML au moyen de la fonction innerHTML-set! (ligne 12).



Conclusion

Cette série progressive d'exemples montre l'intérêt de concevoir, en même temps, serveur et client. Que les variables et services du serveur soient accessibles du client évite de périlleuses acrobaties avec cookies, objets sessions, attribut id sur balises, codage JSON, etc. On programme en style direct, on parle des mêmes variables et leurs valeurs circulent aisément. En attendant le mois prochain où nous présenterons une véritable application d'une cinquantaine de lignes, vous pouvez vous entraîner en téléchargeant la weblet programmez1 qui vous permettra de disposer, dans votre courtier HOP personnel, du code de tous ces exemples. Votre navigateur sera votre IDE pour les modifier et les exécuter à votre guise!

■ **Manuel Serrano**
Directeur de Recherche, Inria
www.inria.fr/mimosa/Manuel.Serrano

■ **Christian Queinnec**
Professeur, Université Pierre et Marie Curie
www-spi.lip6.fr/~queinnec/

Introduction à la programmation multithread

L'architecture PC que nous utilisons aujourd'hui est celle des origines, à quelques évolutions près. Historiquement, c'est le processeur Intel 386 qui a permis la construction des premiers PC. Il s'agissait alors de micro-ordinateurs performants capables de faire fonctionner plusieurs applications en même temps : le multi tâche grand public était né.

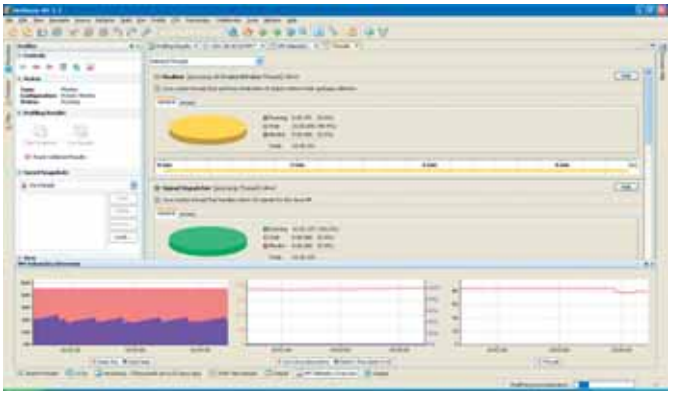
Ce tour de force fut possible grâce à l'intégration d'une unité spécialisée : la MMU. Ce composant permet ainsi d'assurer une gestion de la mémoire et du contexte d'exécution en isolant les programmes avec en tout et pour tout un seul cœur. Afin d'augmenter les performances, le processeur a évolué du point de vue industriel : augmentation de la fréquence ainsi que du nombre de transistors. Le processeur a également vu son architecture évoluer et ce, dans un seul sens : l'amélioration des performances du multitâche. Tout d'abord par un jeu d'instructions étendu : MMX, SSE, HyperThreading et maintenant les instructions de virtualisation. Cependant, la course aux Gh est devenue trop chère et trop compliquée à mettre en œuvre, notamment à cause de la dissipation thermique. C'est pourquoi les grands constructeurs comme Intel et AMD ont commencé il y a quelques années à développer leurs gammes de processeurs double cœur. Aujourd'hui, les configurations avec double cœur sont devenues courantes et déjà le quad core fait son apparition. Cependant, il subsiste un problème pour les développeurs d'applications : la complexité de cette nouvelle organisation de la puissance de calcul nécessite une approche adéquate. Force est de constater que de nombreuses applications n'exploitent pas la technologie dans son entière capacité. Cet article s'attache à présenter un large panorama des différentes techniques.

Différences entre Unix et Windows

Bien que l'architecture multi cœur soit essentiellement une problématique de matériel, le système d'exploitation entre pleinement en action lorsqu'il s'agit de multi tâche. C'est à lui qu'incombe la responsabilité de répartir le temps entre chaque processus en fonction de critères que les concepteurs auront choisis : système temps réel ou en temps partagé. Dans le premier cas, c'est le temps de réponse du système qui est privilégié et dans le second ce sont les performances moyennes. Windows est un système en temps partagé tout comme Linux. Cependant ce dernier peut être modifié pour fonctionner en temps réel via différents patchs sur le noyau. Les systèmes Unix utilisent l'API Posix afin de gérer les threads de la même manière : cycle de vie, communication et synchronisation. En arrière plan, ce sont les appels systèmes du noyau qui sont utilisés. Sous Linux, on connaît bien la commande fork() ou encore execv() qui permettent respectivement de créer un processus fils s'exécutant dans le même contexte et de créer un processus exécutant un programme externe. La famille des systèmes Windows NT supporte l'ensemble de la norme POSIX à l'exception des fonctionnalités optionnelles.

GCC optimise ?

GCC propose depuis sa version 4 la possibilité de mettre en œuvre la vectorisation automatique. Cette technique est une optimisation qui permet de paralléliser l'exécution des instructions. Sur l'architecture PC, cette option du compilateur se fonde sur les technologies SIMD. Ce sont les jeux d'instructions supplémentaires tels que MMX ou SSE d'Intel ou encore 3DNow! du constructeur AMD.



Utilisation de Posix

L'API Posix a pour objectif de " normaliser " l'interfaçage entre les applications et les systèmes d'exploitation. Elle couvre donc de nombreuses techniques telles que la programmation réseau par socket, les entrées/sorties sur fichiers, les signaux ou encore, pour ce qui nous intéresse ici, la gestion des threads et des extensions temps réel. Il est possible de créer des threads de manière simple en programmation système. Les langages les plus appropriés sont historiquement C et C++ mais Python n'a rien à envier au C à ce niveau et permet de le faire aussi bien. Le code source suivant montre la création de deux threads. Celui-ci pourra être compilé sur la majorité des OS Unix dont Linux :

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

void *print_message_function( void *ptr );

main()
{
     pthread_t thread1, thread2;
     char *message1 = "Thread 1";
     char *message2 = "Thread 2";
     int  iret1, iret2;

    /* Création de deux threads qui vont afficher chacun leur message */

    iret1 = pthread_create( &thread1, NULL, print_message_function, (void*)
 message1);
```

```
    iret2 = pthread_create( &thread2, NULL, print_message_function, (void*)
 message2);

     /* On attend que les threads soient prêts  */

     pthread_join( thread1, NULL);
     pthread_join( thread2, NULL);

     /* On affiche la valeur de retour des threads */

     printf("Thread 1 returns: %d\n",iret1);
     printf("Thread 2 returns: %d\n",iret2);
     exit(0);
}

void *print_message_function( void *ptr )
{
     char *message;
     message = (char *) ptr;
     printf("%s \n", message);
}
```

Ce code permet de montrer comment instancier un thread. Il est bien sûr possible d'aller bien plus loin avec l'API Posix et de gérer notamment la communication et la synchronisation des threads. Lorsque l'API est disponible sur le système cible, il est important de privilégier son utilisation plutôt que les appels systèmes du noyau. Voici ce qu'apporte Posix en supplément de la gestion des threads :

- Signaux temps réel
- Planification priorisée des tâches
- Sémaphores
- Entrées sorties asynchrones et synchrones
- Verrous mémoire

Threads sous Windows

La programmation parallèle sous Windows est implémentée via son API. Le nom des fonctions est différent de POSIX, cependant la logique de fonctionnement reste globalement la même. Windows Vista et Windows Server 2008 apportent des améliorations à cette API, notamment sur la gestion des synchronisations. Le nouveau mécanisme appelé Wait Chain Traversal facilite la vie du développeur en apportant une réponse aux problèmes d'étreintes fatales. Ce problème intervient lorsque deux threads s'attendent l'un l'autre. On trouve une documentation importante sur le sujet, et notamment sur la MSDN. Le source suivant, présente la programmation de threads en C# :

```
using System;
using System.Threading;

public class Test
{
    static void Main()
    {
        ThreadStart job = new ThreadStart(ThreadJob);
        Thread thread = new Thread(job);
        thread.Start();
        for (int i=0; i < 5; i++)
        {
            Console.WriteLine ("Main thread: {0}", i);
            Thread.Sleep(1000);
        }
    }

    static void ThreadJob()
    {
        for (int i=0; i < 10; i++)
        {
            Console.WriteLine ("Other thread: {0}", i);
            Thread.Sleep(500);
        }
    }
}
```

Ce code montre le fonctionnement des threads. Ici, la différence principale avec le code C Posix est l'utilisation du paradigme objet qui simplifie quelque peu la tâche. Le code suivant met en œuvre une technique de synchronisation applicable sous Windows avec les fonctions de son API.

```
CRITICAL_SECTION verrou_file;
CONDITION_VARIABLE file_pas_vide;
CONDITION_VARIABLE file_pas_pleine;

extern bool File_Pleine();
extern bool File_Vide();
extern void Inserer_Item_Dans_Liste(Item i);
extern Item Extraire_Item_De_Liste();

DWORD WINAPI Thread_Producteur(PVOID param)
{
  while (true)
  {
    Item i=ProduireItem();
    EnterCriticalSection(&verrou_file);

    while (File_Pleine())
    {
     SleepConditionVariableCS(&file_pas_pleine, &verrou_file, INFINITE);
    }

    Inserer_Item_Dans_Liste(i);

    LeaveCriticalSection(&verrou_file);

    WakeConditionVariable(&file_pas_vide);
  }
}

DWORD WINAPI Thread_Consommateur(PVOID param)
{
  while (true)
  {
    EnterCriticalSection(&verrou_file);
```

```
        while (File_Vide())
        {
          SleepConditionVariableCS(&file_pas_vide, &verrou_file, INFINITE);
        }

        Item i=Extraire_Item_De_Liste();

        LeaveCriticalSection(&verrou_file);

        WakeConditionVariable(&file_pas_pleine);

        ConsommerItem(i);
    }
}
```

Problématiques du multithread

Lors de l'exécution d'une application mono thread, le processus léger a accès aux ressources de l'application : variables, sockets, fichiers… Etant seul, il peut y accéder à volonté et sans restriction. En revanche, dans un contexte multithreadé, chaque processus est susceptible d'accéder simultanément à une même ressource. Il est donc nécessaire de mettre en œuvre des verrous et autres mécanismes de synchronisation. Le plus utilisé est sans aucun doute le sémaphore. Le principe est simple : il s'agit d'un objet permettant d'autoriser un nombre limité d'utilisateurs à une ressource. Lors de son initialisation, on attribue le nombre de threads pouvant utiliser la ressource. La fonction P() permet d'effectuer la demande d'une ressource. Elle reste en attente tant que la ressource n'est pas disponible. La fonction V(), à l'inverse, libère la ressource et permet ainsi à un autre thread d'effectuer ses traitements. L'utilisation des sémaphores permet ainsi d'empêcher les accès simultanés à une ressource avec les effets indésirables et imprévisible que cela peut engendrer. Cependant, mal utilisée elle peut provoquer des situations d'interblocage. La nécessité de rendre le code atomique, c'est-à-dire exécuté en une seule fois sans qu'aucun thread puisse intervenir est implémentée par programmation. En Java, on utilisera le mot clef *synchronized* sur les blocs de codes à verrouiller. L'API Posix offre de nombreuses solutions pour gérer l'ensemble des problématiques de manière uniforme. Windows les couvre et apporte ses propres fonctionnalités supplémentaires.

Rôle de la machine virtuelle en Java

La machine virtuelle Java permet l'exécution d'applications Java sur de nombreuses plates-formes : Windows, Linux, Solaris… Afin d'implémenter une librairie portable et fournir ainsi une interface commune de programmation, le JDK propose des classes et des interfaces de base ainsi que des classes plus évoluées, permettant de gérer les problématiques de synchronisation et de communication.

Créer un thread

En Java, il y a deux méthodes pour développer des Threads. La première est la plus simple mais également la moins souple : elle consiste à utiliser l'héritage. La classe Thread est l'implémentation directe du concept de processus léger. Ainsi, il est possible d'hériter de cette classe et de redéfinir la méthode run() pour créer un thread :

```
public class SimpleThread extends Thread {
public SimpleThread() {
super.setName("SimpleThread");
}
@Override
public void run() {
while(true) {
try {
System.out.println("Hello");
Thread.sleep(1000);
} catch (InterruptedException ex) {
System.err.println(ex);
}
}
}
}
```

Ce processus va afficher le message " hello " toutes les secondes. Nous avons pris soin de nommer le thread. Dans la pratique, java n'impose pas le nommage explicite des threads à leur construction, cependant, cela nous sera utile par la suite pour faire du monitoring et du profiling. Lançons le Thread depuis une classe Lanceur :

```
public class Main {

public static void main(String[] args) {
new SimpleThread().start();
}

}
```

Le lancement du thread se fait en une seule ligne. Cependant faites attention, il ne faut surtout pas appeler la méthode run(), sans quoi la méthode sera exécutée dans le même thread que Main. L'appel de la méthode start() est obligatoire, c'est cette méthode qui créera le Thread. Cette façon de faire n'est pas la plus répandue, bien qu'étant la plus simple. En effet, Java ne supporte pas l'héritage multiple, utiliser l'héritage revient à empêcher la classe d'hériter d'une autre classe. Pour pallier ce problème, la seconde technique utilise une interface. Ainsi on peut faire hériter notre thread d'une classe tierce.

```
public class GoodThread implements Runnable {
public void run() {
String message = new String("Hello");
while(true) {
try {
System.out.println(message);
Thread.sleep(10);
} catch (InterruptedException ex) {
System.err.println(ex);
}
}
}

}
```

Comme pour la classe Thread, c'est la méthode run() qui contient le code applicatif. A ce niveau, seule la déclaration de l'implémentation de l'interface Runnable a changé. En revanche, le code nécessaire pour lancer le thread est légèrement plus complexe :

```
public class Main {
public static void main(String[] args) {
Thread t = new Thread(new GoodThread(););
t.setName("GoodThread");
t.start();
}
}
```

Une fois le thread instancié, il n'y a en revanche plus de différences entre les deux façons de faire. C'est pour cette raison, que l'interface Runnable est la plus utilisée. Le delta de complexité est largement compensé par l'apport en souplesse sur le code.

Manager les threads

Si votre application doit gérer de nombreux threads, il sera important de bien architecturer votre code en prévoyant notamment l'implémentation d'une fabrique de Threads qui masquera la complexité de leur instanciation. Une telle classe pourra conserver les références dans une *Liste d'objets* et aura le contrôle sur le cycle de vie des threads. Elle pourra ainsi gérer les *Singleton* de manière rigoureuse. Les méthodes permettant de manager les threads se trouvent toutes dans la classe Thread. La méthode *sleep()* permet de faire dormir un processus, ainsi il ne consommera plus de temps CPU pendant une durée définie. Si vous souhaitez faire dormir un processus non pas sur une durée déterminée, mais sur un événement déterminé, vous devez appeler la méthode *wait()*. Le processus sera inactif jusqu'au prochain appel à la méthode *notify()*. Pour réveiller tous les threads en sommeil, vous pouvez également utiliser *notifyAll()*. Attention à son utilisation cependant. Il existe de très nombreuses méthodes dont l'utilisation nécessite une lecture attentive de la javadoc. Nous avons vu ici l'essentiel, les autres méthodes servent surtout à faire de la synchronisation entre threads, que nous allons voir plus en détail maintenant.

Problématique de concurrence

Lorsque de nombreux threads s'exécutent, des problèmes peuvent survenir. Il est alors important d'isoler les ressources afin de contrôler l'exécution des threads et éviter tout accès concurrent qui peut avoir des conséquences catastrophiques dans une application. Java propose un package dédié à cette problématique. En fin d'article un lien vers le tutoriel officiel vous permettra de prendre pleinement connaissance de ses fonctionnalités. On y retrouve des solutions bien connues tels que les sémaphores. Des structures de données y ont été réécrites pour les rendre " Thread safe ", c'est-à-dire qu'elles ne comportent pas de risque de compromission en utilisation multithread. La plupart des opérations étant rendues atomique. L'exemple suivant illustre la mise en œuvre de cette API :

```
import java.util.concurrent.atomic.AtomicInteger;

class AtomicCounter {
    private AtomicInteger c = new AtomicInteger(0);

    public void increment() {
        c.incrementAndGet();
    }

    public void decrement() {
        c.decrementAndGet();
    }

    public int value() {
        return c.get();
    }

}
```

Cette implémentation présente l'intérêt de ne pas utiliser le mot clef *synchronized* qui provoque des blocages inutiles, puisque trop large. La classe *AtomicInteger* ne bloque que la portion de code critique. Dans cet exemple, ce n'était pas nécessaire, mais pour les classes plus complexes, il est préférable d'avoir systématiquement recours à ce type de classe disponible dans le package java.util.concurrent.atomic.

Monitoring et profiling avec Netbeans 6

Déjà présent sous forme de plug-in dans la version 5, le profiler sera intégré de base dans l'environnement Netbeans avec un lot de nouvelles fonctionnalités et d'optimisations. La sortie de Netbeans 6, qui sera effective quand vous lirez ces lignes, va être accompagnée de nombreux modules déjà disponibles en bêta. L'outil permet la détection des goulots d'étranglements ainsi que des bugs et autres fuites mémoires. L'utilisation du profiler prend encore plus de sens en programmation multithread puisque celui-ci permet la visualisation en temps réel de la part de ressources consommée par thread (temps cpu, mémoire ...). Une bonne utilisation de ces outils permet ainsi la correction et d'optimiser les performances globales de votre application. La surcharge générée par l'outil est conséquente si vous faites le choix d'analyser le comportement de l'application dans les moindres détails. A vous de choisir les options qui vous conviennent au démarrage d'une session de profiling. La surcharge peut ainsi atteindre 50% des temps CPU. Il faudra donc prendre du recul par rapport aux performances de l'application lancée dans ce contexte.

Conclusion

La programmation multithread permet d'optimiser les performances de nos applications sur les architectures matérielles dont nous disposons aujourd'hui. La complexité des développements tend à être mieux gérée par des environnements de développement et des compilateurs plus efficaces. Les concepteurs des systèmes d'exploitation emboîtent le pas en proposant des API plus robustes. Le développeur à donc plus que jamais intérêt à se former à ces techniques. Cet article avait pour objectif de vous présenter l'étendue des possibilités. Il serait intéressant d'aller plus loin, par exemple, sur l'étude de POSIX ou des mécanismes apportés par Java et .NET.

Loïc Guillois

Et les serveurs d'applications ?

On aurait tort de croire que seules les applications lourdes sont impactées par la problématique de programmation multithread. Les serveurs d'applications font déjà grand usage des threads pour répondre aux requêtes de chaque client. De plus, il est fort probable qu'il vous soit nécessaire de mettre en œuvre des tâches de fond. Le faire avec les techniques traditionnelles n'est pas envisageable. Un outil efficace et dédié à ce travail existe : Quartz et sa variante .NET. Il permet ni plus ni moins de disposer d'un planificateur de tâches au sein du serveur d'applications Java EE.

RIA pour téléphone mobile

1re partie

Les applications internet riches pour les téléphones mobiles se heurtent à des contraintes de portabilité largement plus grandes que celles du Web. J2ME propose des interfaces pour construire ces applications qui possèdent l'avantage d'être basées sur une standardisation et d'être disponibles sur une large gamme de téléphones. Cette première partie explore la construction sous J2ME d'interfaces utilisateur pour le téléphone mobile.

Le téléphone offre à l'utilisateur une disponibilité permanente de la connexion Internet. A ce titre, il doit être considéré comme un moyen privilégié d'accéder aux services en ligne.

Pour construire des applications internet riches, les mobiles "haut de gamme" disposent de plusieurs solutions de type " Flash ". Ces outils sont très similaires à ceux existant sur le Web et fournissent des applications à la fois graphiques et utilisant les fonctionnalités du mobile.

Pour la majorité des autres téléphones, l'interface pour les applications en ligne est le browser. Mais l'ergonomie du browser est d'autant plus insuffisante qu'il ne dispose en général ni des plug-in, ni même du JavaScript. J2ME offre un compromis pour la réalisation des applications riches. Il possède une gestion de l'écran des mobiles, une gestion des interactions utilisateur et un accès à Internet et a l'avantage d'être largement répandu sur l'ensemble des mobiles. De plus, J2ME a été conçu dans un but de portabilité et de standardisation, les applications riches ainsi bâties pourront offrir un large niveau de portabilité. Son point faible est l'absence d'outils de haut niveau pour la réalisation d'interfaces graphiques.

J2ME pour la gestion des interactions utilisateurs

J2ME est le terme qui définit les environnements Java pour les équipements mobiles. Pour les téléphones mobiles, J2ME a une configuration: CLDC 1.1 qui représente une spécification d'interface de la machine virtuelle et des classes de base, et un profil: MIDP 2.0 qui correspond à un ensemble de classes permettant d'écrire des applications. Pour ce profil, le package qui gère les interactions avec les utilisateurs est le LCDUI (Liquid Cristal Display User Interface). Ce package possède deux modèles d'écran *(Displayable)*, un modèle pour les applications graphiques et les jeux *(Canvas)* et un modèle pour l'échange d'information avec l'utilisateur *(Screen)*. L'utilisation de *Canvas* permet de réaliser des applications offrant une très forte interactivité comme des jeux. Elle nécessite une programmation complexe contrôlant les éléments graphiques au niveau du pixel et prenant en compte les contraintes temps réel.

La classe *Screen,* au contraire, est conçue pour simplifier le développement des classes gérant les interactions entre l'utilisateur et le téléphone. Elle propose la gestion de :

- Menu : *List*
- Entrée de texte : *TextBox*
- Formulaire : *Form*
- Message d'erreur: *Alert*

List permet de gérer un système de menus qui peuvent être parcourus et déclenchés avec une utilisation du clavier.

TextBox permet la saisie texte avec une gestion du scrolling.

Alert permet l'affichage temporaire d'un message d'erreur et l'enchaînement sur un exemple de traitement. *Form* permet la composition de formulaires complexes.

Elle utilise l'agrégation d'éléments *Item*. Les spécialisations de la classe *Item* offrent tous les éléments nécessaires pour réaliser un formulaire de configuration ou de requête :

- ***StringItem, ImageItem :*** affichage d'une chaîne de caractères ou d'une image, ces éléments pouvant être dans un bouton.
- ***DateField, TextField :*** édition d'un champ date ou texte
- ***ChoiceItem :*** gestion d'une boîte de sélection
- ***Gauge :*** affichage d'une jauge

Les *Items* possèdent un attribut *layout* permettant de contrôler leur alignement et leur disposition dans le formulaire.

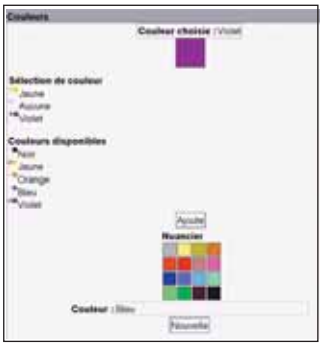


Diagramme UML simplifié de l'interface utilisateur J2ME

L'exemple 1 présente la mise en oeuvre de ces *Items* avec certaines options. L'application est un formulaire de choix de couleur qui permet de choisir une couleur dans un *ChoiceItem* de type " bouton radio ", les couleurs étant elles-mêmes choisies dans un panel qui est un autre *ChoiceItem* mais à choix multiples. Les actions sur les *Items* sont gérées par un mécanisme de traitement des événements par délégation :

```
buttonUpdateItem.setItemCommandListener(new ItemCommand
Listener() {
public void commandAction(Command c, Item item) { // Add
appuyé on ajoute les couleurs
....
}
}
```

La classe anonyme permet, quand le bouton est appuyé, de transférer les couleurs du panel à la boîte de sélection.

Un *Item* particulier est le *CustomItem,* il permet d'enrichir les champs disponibles par des fonctionnalités avancées. L'exemple 2 présente un tel *CustomItem* complexe implémentant un nuancier. Un CustomItem permet d'utiliser les capacités " bas niveau " de gestion de l'affichage et du clavier qui sont réservées au *Canvas* et de les intégrer à l'intérieur d'une *Form*. Et il conserve les mêmes capacités de portabilité liées à l'utilisation des *Forms*.

J2ME offre plusieurs extensions pour afficher des éléments plus élabo-

rés mais elles vont être restreintes à une classe limitée de mobiles et entraîner la limitation de la diffusion des applications :

- La visualisation de vidéo (3GP)
- La visualisation de graphiques 2D animés (SVG)
- La visualisation d'objets 3D (M3G)

Ce tour d'horizon donne un aperçu des capacités de l'application embarquée d'interagir avec l'utilisateur, la seconde partie de l'article s'intéressera à la communication avec le serveur d'application.

Olivier Théry

L'exemple 1 est disponible sur le site

```
package exemple2;

import javax.microedition.lcdui.Canvas;
import javax.microedition.lcdui.CustomItem;
import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.game.Sprite;

public class ColorSelectorItem extends CustomItem {
    private int[] colors = {
   0xbfbfbf,  0xffff80, 0xc0c000,  0xc08000, 0xc04000, 0xc00000,
0xC08080, 0xff40c0,
   0x4000c0, 0x6060e0, 0x80c0ff, 0x80ffc0, 0x80ff40, 0x00c000,
0x402020, 0x202020 };

    private int posx; / / position du curseur sur l'écran
    private int posy;

    private int preferredWidth;
    private int preferredHeight;

    private int minimalWidth = 80;
    private int minimalHeight = 80;


    public ColorSelectorItem(String title) {
        super(title);
        preferredWidth = minimalWidth;
        preferredHeight = minimalWidth;
    }


    protected int getMinContentWidth() {
        return minimalWidth;
    }

    protected int getMinContentHeight() {
        return minimalHeight;
    }

    protected int getPrefContentWidth(int arg0) {
        return preferredWidth;
    }

    protected int getPrefContentHeight(int arg0) {
        return preferredHeight;
    }

    public int getColor() {
        return colors[posx + 4 * posy];
    }

    protected void keyRepeated(int keyCode) {
        keyPressed(keyCode);
    }


    protected void keyPressed(int keyCode) {
        switch(keyCode) {
            case Canvas.KEY_NUM2 :
                posy-;
                if(posy < 0)
                    posy = 0;
                repaint();
                break;
            case Canvas.KEY_NUM8 :
                posy++;
                if(posy >= 4)
                    posy = 3;
                repaint();
                break;
            case Canvas.KEY_NUM4 :
                posx-;
                if(posx < 0)
                    posx = 0;
                repaint();
                break;
            case Canvas.KEY_NUM6 :
                posx++;
                if(posx >= 4)
                    posx = 3;
                repaint();
                break;
        }

    }

    public void paint(Graphics g, int w, int h) {
        int tx, ty;
        for(int i = 0; i < 16; i++) {
            tx = i % 4;
            ty = i / 4;
            g.setColor(colors[i]);
            g.fillRect(tx * 20 + 2, ty * 20 + 2, 16, 16);
            g.setColor(((tx == posx) && (ty == posy)) ? 0xff0000 : 0x0);
            g.drawRect(tx * 20 + 2, ty * 20 + 2, 16, 16);
        }
    }

}
```

Etendre la console de Windows Home Server

Microsoft a fait sa renommée et les débuts de son succès grâce à un système d'exploitation, célèbre mais maintenant délaissé, MS-DOS, destiné aux PC IBM en 1981. Depuis cette époque Microsoft a conquis bien des marchés et domaines différents. Avec son nouveau système d'exploitation Windows Home Server, Microsoft espère aujourd'hui se positionner sur le marché des réseaux domestiques, domaine où il était encore absent.

Ce nouveau système d'exploitation associe la puissance et la robustesse du système serveur – Windows Server 2003 – à la simplicité d'une interface accessible à toute personne, y compris les non techniciens. Néanmoins, derrière cette simplicité se trouve une véritable plate-forme serveur que tout bon technicien ou développeur pourra exploiter à sa façon.

Qu'est–ce que Windows Home Server ?

Windows Home Server propose d'unifier, de sauvegarder et de diffuser les données des appareils multimédia (Il est compatible *Windows Media Connect*) à travers votre réseau domestique, que ce soit celles de vos ordinateurs, de votre appareil photo, votre lecteur MP3, etc. Windows Home Server cache la complexité et la puissance d'un système d'exploitation robuste derrière une interface – appelée la console Windows Home Server - qui rend extrêmement simple, même pour une personne novice, l'accès et l'administration de son réseau.

Windows Home Server offre cinq points clés :

- Administration simplifiée grâce à la console Windows Home Server,
- Centralisation du stockage : flexibilité, grâce à Drive Extender et sécurité, grâce à la duplication,
- Sauvegarde planifiée des ordinateurs du réseau (sauvegardes incrémentales),
- Accès à distance aux ordinateurs de votre réseau depuis Internet,
- Streaming de vos fichiers multimédias (Windows Media Player, XBOX 306, etc.)

Derrière la console Windows Home Server se trouvent les raffinements de Windows Server 2003, vous allez donc pouvoir utiliser cette plate-forme pour héberger vos sites web, votre SharePoint, un Virtual Server 2005, … Sachez que pour rendre les opérations de maintenance plus simples à vos utilisateurs, vous pouvez également étendre l'interface de la console Windows Home Server. C'est d'ailleurs sur cet aspect que la suite de cet article se focalisera.

Développement sur Windows Home Server

Vous pouvez étendre Windows Home Server de deux façons : étendre la plate-forme (site web, services, etc.) ou étendre Windows Home Server (notifications, partages, sauvegardes, etc.). Dans les deux cas, vous pourrez étendre la console pour faciliter l'accès aux fonctionnalités et aux paramètres.

Dans le premier cas, vous avez la possibilité de créer des applications de toutes sortes en vous appuyant sur les API de WHS vous donnant accès à différentes informations cruciales : les ordinateurs clients connectés, les partages, les disques du serveur, les sauvegardes, les notifications, etc. Ainsi vous pourrez développer vos propres solutions web, Web Services ou applications WSS si vous le souhaitez. Dans le second cas, vous allez pouvoir rajouter des onglets à ceux déjà existants dans la console WHS et/ou dans le panneau de paramètres de la console pour permettre une exécution ou un paramétrage aisé.

Pour vous aider dans votre tâche de développeur, Microsoft a mis à votre disposition un SDK composé de :

- Deux assembly .NET : *Microsoft.HomeServer.SDK.Interop.v1 et HomeServerExt.dll*
- Une documentation située à cette adresse : http://msdn2.microsoft.com/en-us/library/bb425866.aspx

Dans cet article, nous allons nous focaliser sur les possibilités d'extensibilité de la console d'administration de Windows Home Server qui vous permettra de rajouter des onglets (et donc des fonctionnalités) et une fenêtre de configuration. Le développement d'un complément à la console Windows Home Server nécessite un minimum de rigueur, comme vous le verrez, sur les conventions de nommage. Cet article n'est pas limitatif des possibilités offertes par l'extensibilité de la console WHS, et se veut être une courte introduction au développement et aux possibilités d'extension de la console.

La solution type d'une extension Windows Home Server

Nous allons utiliser Visual Studio 2005 pour créer une solution qui sera composée de deux projets distincts :

- Un projet de bibliothèque de classes (une extension WHS est une DLL) qui vous permettra de créer l'onglet et les classes nécessaires pour la console,
- Un projet de déploiement WiX (Windows Installer XML) qui vous permettra de créer un exécutable d'installation MSI pour déployer l'extension WHS sur le serveur.

Création du projet de déploiement

Il y a deux manières de réaliser le MSI nécessaire au déploiement de l'add-in dans WHS : utiliser WiX pour créer directement le MSI désiré, ou alors utiliser un projet de déploiement dans Visual Studio 2005 puis l'outil *ORCA.exe* pour modifier les propriétés du MSI (mettre la valeur de la propriété *WHSLogo* à *1*) pour que Windows Home Server l'identifie comme un exécutable de déploiement d'extension WHS. Dans cet article nous allons traiter uniquement de la méthode avec WiX. Pour cela, vous devez installer la dernière version de l'add-in pour Visual Studio 2005 nommé Votive afin d'avoir à votre disposition les modèles de


Fig.1

projet WiX (Fig.1). Ajoutez un projet à votre solution existante en choisissant un projet de type *WiX Project* afin de générer un installeur qui effectuera les tâches suivantes :

- Vérifier que l'ordinateur cible est bien un Windows Server 2003 ou Windows Home Server,
- Ajouter une propriété *WHSLogo* avec une valeur à *1*,
- Copier les fichiers nécessaires à votre application dans le répertoire *%systemdrive%\Program Files\Windows Home Server*.

Pour avoir l'intégralité du script WiX, reportez vous aux sources associées à l'article.

Création et configuration du projet d'add-in WHS

Dans Visual Studio 2005, ajoutez un projet de type bibliothèque de classes que vous pouvez nommer *WHSProgrammezAddin.* Afin de déployer l'extension sur le serveur, WHS demande à la DLL d'extension de suivre la convention de nommage suivante : ***HomeServerConsoleTab****.LeNomDeVotreOnglet****.dll.***

Pour cela, changez dans les propriétés du projet, le nom de l'assembly de la bibliothèque et nommez la ***HomeServerConsoleTab.OpenXMLDemo.*** En plus de cette convention portant sur le nom de la DLL, tous vos fichiers source doivent utiliser un espace de noms respectant cette forme : ***Microsoft.HomeServer.HomeServerConsoleTab.****LeNomDeVotreOnglet.* Afin de vous faciliter la tâche par la suite, il est d'une bonne pratique de changer l'espace de nom par défaut dans les propriétés du projet en spécifiant ***Microsoft.HomeServer.HomeServerConsoleTab.Programmez*** : (Fig.2).

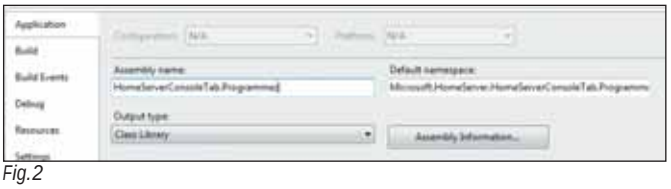


Fig.2


Fig.3

Un autre élément nécessaire à un onglet est l'image 32x32 qui lui sera associée. Ajoutez une ressource de type Image à votre projet qui sera utilisée comme icône de l'onglet (Fig.3).

N'oubliez pas d'ajouter les dépendances vers les assembly de Windows Home Server, soit les DLL ***HomeServerExt.dll et Microsoft.HomeServer.SDK.Interop.v1.dll*** se trouvant dans le répertoire ***%ProgramFiles%\Windows Home Server*** de votre serveur.

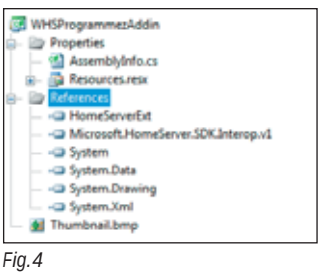


Fig.4

Récupérez donc ces DLL sur votre serveur Windows Home Server (les sources associés à cet article n'incluent pas ces DLL) et référencez les dans le projet : (Fig.4).

Pour résumer les différentes étapes de configuration du projet, il vous faudra :

- Configurer le nom de l'assembly de votre bibliothèque de classe ainsi que l'espace de nom par défaut (ou ne pas oublier de le changer par la suite dans vos classes) selon les conventions de nommage indiquées plus haut,
- Ajouter une image 32x32 qui sera associée à votre onglet,
- Ajouter les dépendances vers les assembly de Windows Home Server.

Fig.5

Si vous jetez un coup d'œil dans le répertoire *c:\Program Files\Windows Home Server,* vous remarquerez que les DLLs par défaut de la console ne dérogent pas à la règle de nommage des assembly (Fig.5).

Création de l'onglet pour la console WHS

Les projets sont créés et configurés, attaquons nous maintenant à la réalisation de notre onglet. Un onglet pour la console Windows Home Server se compose au minimum de trois éléments principaux :

- Une image associée au titre de l'onglet,
- Une classe implémentant l'interface *IConsoleTab,*
- Un contrôle utilisateur qui représente la surface graphique/utilisateur de l'onglet.

Pour créer, vous devez nécessairement implémenter une classe nommée ***HomeServerTabExtender*** qui implémente l'interface ***Microsoft.HomeServer.Extensibility.IConsoleTab :***

```
...
using Microsoft.HomeServer.Extensibility;
using Microsoft.HomeServer.SDK.Interop.v1;

namespace Microsoft.HomeServer.HomeServerConsoleTab.Programmez
{
    public class HomeServerTabExtender : Microsoft.HomeServer.Extensibility
.IConsoleTab
    {
        private ucInfoPanel nPanel;
        public HomeServerTabExtender(int width, int height, IConsoleServices
 consoleServices)
        {
            ...
        }

        public bool GetHelp() { return false; }

        public Guid SettingsGuid { get { return Guid.Empty; } }

        public string TabText { get { return "Programmez!"; } }

         public Bitmap TabImage { get { return Properties.Resources.Thum
bnail; } }

        public Control TabControl { get { return myPanel; } }
    }
}
```

Voici un descriptif rapide des différentes méthodes de l'interface *IConsoleTab :*

- La méthode *GetHelp()* spécifie si vous fournissez ou non une aide à l'utilisateur pour cet onglet.
- La propriété *SettingsGuid* spécifie l'identifiant unique du panneau de paramètres associé à cet onglet. Si vous ne souhaitez pas mettre de panneau de configuration – comme dans le cadre de cet article – vous pouvez renvoyer un Guid vide.
- La propriété *TabText* retourne le texte qui sera affiché dans l'onglet de la console en dessous de l'image.
- La propriété *TabImage* retourne l'image qui sera affichée dans l'onglet au dessus du texte de TabText.
- La propriété *TabControl* retourne le contrôle utilisateur qui sera affiché dans l'onglet.

La classe WHSInfoClass

Les API d'extensibilité de la console Windows Home Server - principalement concentrées dans les deux DLL que nous référençons – permettent de récupérer un certain nombre d'informations. Même si leur utilisation n'est pas toujours des plus pratiques, les informations retournées par la méthode WHSInfoClass, vous permettront de réagir et d'interagir avec Windows Home Server de façon relativement simple.
Voici un extrait du code utilisé dans la solution de cet article permettant de lister les dossiers partagés :

```
StringBuilder sb = new StringBuilder();
WHSInfoClass whsInfo = new WHSInfoClass();
foreach (IShareInfo shareInfo in whsInfo.GetShareInfo())
{
sb.AppendLine(String.Format("Le partage '{0}' est localisé dans '{1}' : {2} - Duplication {3}",
shareInfo.Name, shareInfo.Path, shareInfo.Description, shareInfo.IsDuplicated == 1 ? "activé" : "désactivée"));
}
```

Création d'un panneau de paramétrage

La création et le fonctionnement d'un panneau dans le menu de *Paramètres* de la console de Windows Home Server est très similaire à celle d'un onglet de console :

- Nommer la classe *HomeServerSettingsExtender,*
- Implémenter l'interface *Microsoft.HomeServer.Extensibility.ISettingsTab,*

Les seules différences se trouvent au niveau des méthodes à implémenter :

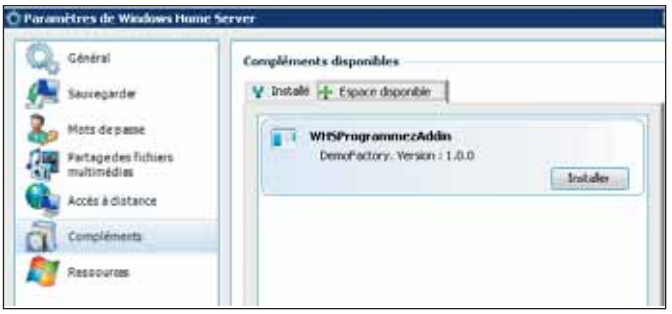
```
public bool Commit() { /* Actions ici */ return true; }

public Guid SettingsGuid { get { return new Guid("{53D21B83-C36A-4b11-8262-E8DA3165FD5D}"); } }
```

La méthode *Commit()* permet d'appliquer les modifications à votre logique. Cette méthode est appelée lorsque l'utilisateur clique sur 'OK' ou 'Appliquer'. La propriété *SettingsGuid* spécifie l'identifiant unique de cet onglet de paramètres. Cette référence sera utilisée pour ouvrir le panneau de paramètres avec la méthode :

```
IConsoleServices.OpenSettings(new Guid("{53D21B83-C36A-4b11-8262-E8DA3165FD5D}"));
```

Déploiement d'une extension Windows Home Server

Pour déployer l'add-in dans Windows Home Server, il vous suffit de copier le fichier de déploiement MSI généré par WiX dans le répertoire **\\NomServeur\Logiciels\Add-ins** de votre serveur WHS. Une fois le fichier MSI copié dans le répertoire, rendez-vous dans le panneau des paramètres de Windows Home Server, onglet vertical *Compléments* puis onglet *Espace disponible* dans lequel vous devriez trouver votre add-in. Il ne vous reste plus qu'à cliquer sur *Installer* pour installer l'add-in.



Une fois l'opération effectuée la console se fermera. Authentifiez-vous à nouveau et vous devriez avoir un résultat similaire au suivant :

Conclusion

Etendre la console de Windows Home Server ne représente pas une tâche compliquée même pour un novice en programmation. Au-delà de son apparence de produit *out-of-the-box,* Windows Home Server représente une véritable plate-forme serveur pour le développement de solutions destinées à des usages domestiques. Il faudra juste tenir compte du public visé pour faire de vos créations des succès.

Références

- Documentation du SDK : http://msdn2.microsoft.com/en-us/library/bb425862.aspx
- Projet WiX : http://wix.sourceforge.net/
- Addin Visual Studio 2005 Votive : http://wix.sourceforge.net/votive.html
- Outil ORCA : http://msdn2.microsoft.com/en-us/library/aa370557.aspx
- Téléchargement du .NET framework 3 : http://www.microsoft.com/downloads/details.aspx?FamilyID=10CC340B-F857-4A14-83F5-25634C3BF043&displaylang=fr
- Spécifications Open XML ECMA-376 : http://www.ecma-international.org/publications/standards/Ecma-376.htm

Julien Chable - *Wygwam*

Concours Home Server du 1er au 31 janvier 2008
http://msdn2.microsoft.com/fr-fr/coding4fun/

LANGAGE : VB
DIFFICULTÉ : ●●●○○
APPLICATION : MIGRATION
SOURCE : non

Migrez vos codes VB sur Linux et Mac OS X avec REALbasic

1re partie

Avec le lancement de Visual Basic .Net, de nombreux développeurs Visual Basic se sont sentis abandonnés. Pour la première fois, une nouvelle version de VB n'assurait pas de compatibilité ascendante. Donc, que cela vous plaise ou non, si vous êtes un développeur VB qui souhaite faire évoluer son application, vous serez dans l'obligation de porter votre code.

Pour conservez vos connaissances et valoriser votre savoir-faire Visual Basic, vous pouvez migrer votre code vers REALbasic (RB), outil de développement cross plate-forme et compiler vos applications pour Linux, Mac OS X et Windows, y compris Windows 98, NT, Me, 2000, XP et Vista.

Options de portage

Porter son code implique l'utilisation de nouveaux mots clés qui ne fonctionnent pas comme vous en aviez l'habitude. Dans la mesure où vous allez apprendre un nouveau langage et une nouvelle IDE, il semble logique de regarder autour de vous. REAL Software propose son atelier de développement REALbasic, orienté objet, assez proche de VB. Il peut utiliser la majorité de votre code sans le modifier, et lire vos formulaires. Vous pouvez porter votre code pour obtenir une application fonctionnant sous tous les Windows 32-bit (98/NT/ME/2000/XP/Vista) sans utiliser le framework .Net; Linux (tout Linux Intel utilisant GTK2.0+ comme Novell Linux Desktop, RedHat Desktop, Ubuntu ou Mandriva) et Mac (Mac OS X, UB, Intel et Classic). De plus, l'outil ne nécessite pas de runtime ou de framework et votre application ainsi portée inclut des objets d'interface natifs rendant son graphisme très agréable.

Pour vous le prouver, voici ci-contre les dessins d'écran d'une application compilée avec REALbasic.

Considérations pour le portage

Avant de migrer vers RB, il vous faut considérer l'ensemble du travail à effectuer. Les systèmes d'exploitation Linux et Mac sont très différents de Windows. Par ailleurs, l'intégralité de votre code ne sera pas transposable, mais au terme de votre travail avec RB vous disposerez d'une application vraiment cross plate-forme!

Mots clefs du langage

Vous serez agréablement surpris de voir combien de mots clefs RB et VB sont identiques, par exemple : *Left, Right, Asc, Str*. La grande majorité de vos connaissances sont directement utilisables dans RB. Bien sûr, certains mots clefs ne fonctionnent pas de la même manière. Par exemple : Mid ne peut pas utiliser une chaîne de caractères :

```
Mid(x,2,1) = "!" 'ne fonctionne pas en REALbasic
```

Mid est une fonction (retourne une sous-chaîne) et une expression (modifie une chaîne) en Visual Basic. En RB, Mid est uniquement une fonction, donc il vous faut une expression de substitution pour Mid. Vous pouvez avoir des fonctions et des méthodes qui portent le même nom, ce qui est logique puisque leurs fonctionnalités sont différentes. Par exemple :

```
Sub Mid(ByRef Text As String, StartPos As Integer,_
Length As Integer = -1, Assigns SubString As String)

 //replacement exact pour l'expression Mid dans VB
 // paramètre de longueur optionnel (comme VB)

Dim max as Integer

 // gestion du paramètre de longueur optionnel
   max = Len(Text) 'maximum size allowed

 // extrait & concatène la chaîne

Text = Left(Text, StartPos) + Left(SubString, _
      Length) + Mid(Text, StartPos + Length + 1)

End Sub
```

se substitue à Mid, donnant à RB une expression Mid en complément de la fonction Mid existante. Une fois écrit, vous utilisez Mid comme actuellement. C'est très simple.

Vous ne pouvez pas non plus utiliser une variable sans déclaration (Dim) explicite. Donc, avant de porter votre code, l'une des choses les plus importantes à faire est de le nettoyer! Enlevez les transtypages variables ($, #, %, etc.), utilisez l'Option Explicit pour localiser toutes les variables dynamiques non déclarées, et déclarez-les.

Comme dans tout langage, les mots réservés peuvent entrer en conflit. RB comprend une classe String, et le nom String est réservé, rendant le mot clef String de VB incompatible. Dans ce cas, il vous faut changer

votre code après l'avoir ouvert dans RB. Puis, vous devez écrire une fonction de remplacement pour String, et enfin chercher et remplacer tous les mots clefs VB String avec la fonction de remplacement.
RB et VB ont également chacun leurs propres mots clefs. Par exemple, VB utilise Space alors que RB ne l'utilise pas, mais ceci peut facilement être reproduit comme suit :

```
Function Space (n as integer) As string
   dim s as String
   for i as Integer = 1 to n
      s = s + " "
   next
   Return s
End Function
```

D'un autre côté, RB utilise les mots Min, Max & CountFields. Dans certains cas, des mots clefs différents ont la même fonction. Par exemple, REALbasic utilise UpperCase, et non Ucase.

Il est assez simple d'écrire un programme enveloppant (wrapper) autour de l'implémentation RB afin de ne pas modifier votre code. Cependant, la performance sera meilleure si vous remplacez Ucase (ou Lcase) par UpperCase et LowerCase.

Syntaxe du code

En comparant les syntaxes, l'utilisation des mots clefs est parfois différente. Par exemple, l'expression ci-après est valide pour les deux outils :

```
Dim X, Y As String
```

Mais à cause de l'expression VB Def XXX (qui n'existe pas en RB), la déclaration X ci-dessus peut ne pas être une string! En RB, cette syntaxe déclare X et Y comme String.
Cependant, il y a des différences intéressantes qui demandent à ce que vous modifiiez un peu votre code. Par exemple, quand vous effectuez une conversion entre une virgule flottante et un entier, VB arrondit le nombre à convertir alors que RB le tronque. En Visual Basic :

```
Private Sub Command1_Click()
   Dim d As Double
   Dim i as Integer

   d = 1.5
   i = d
   MsgBox Str(i) 'retourne "2"

End Sub
en revanche, en REALbasic:
Sub Action()
   Dim d as Double
   Dim i as Integer

   d = 1.5
   i = d
   MsgBox Str(i) 'retourne "1"
End
```

Connaître ces différences à l'avance rend le portage et le debugging beaucoup plus rapides. Dans cette situation, il vous faudra remplacer l'attribution directe de la variable par une opération Round afin de produire le même résultat. Alors que VB et RB disposent d'opérateurs logiques (Or, And, Not) qui peuvent être utilisés comme suit,

```
If A or B and Not C Then Beep
```

Visual Basic autorise l'utilisation des opérateurs (Or, And, Xor, Not) également comme des expressions. Par exemple, en Visual Basic vous pouvez écrire :

```
A = 2 Or B
If A Or B Then _
   A = A And B
- OU -
A = 2 Or B
If A Or B Then
   A = A And B
End If
- OU -
A = 2 Or B
If A Or B Then A = A And B
```

Note: En RB les deux dernières lignes fonctionneront parfaitement, sous réserve que A et B soient tous les deux des booléens. Et il n'autorise pas l'utilisation des entiers comme des booléens.

RB, d'un autre côté, met à disposition un support logique intégral et des opérations logiques sur les bits utilisant la classe BitWise. Par exemple :

```
i = BitWise.BitAnd(A, B)
i = BitWise.BitOr(A, B)
i = BitWise.BitXor(A, B)
i = BitWise.BitOnesCompliment(A, B)
i = BitWise.ShiftLeft(A, B)
i = BitWise.ShiftRight(A, B)
```

En résumé, il est recommandé de supprimer toutes les expressions VB DefXXX et d'utiliser des variables et fonctions de nommages partout, avant de commencer le portage. Supprimez les typages variables, et typez chaque variable explicitement en utilisant Dim. Supprimez $, %, #, @ etc. de toutes vos déclarations, variables, fonctions et constantes. Vérifiez et identifiez vos boucles conditionnelles. Vous pourrez les réparer APRES le portage en RB.
Dans la seconde partie de cette article, nous étudierons les similitudes et différences du point de vue des données, de la structure du langage, des Entrées/Sorties, de l'interface afin de faciliter le portage de vos applications.

Hank Marquis

OpenLaszlo : une alternative à Flex

2e partie

La première partie de cet article a permis de détailler les concepts principaux d'OpenLaszlo ainsi que de réaliser une application type "helloWorld". Dans cette deuxième partie, nous allons réaliser un front-end pour Google Reader.

Avant toute chose, et pour ceux qui ne connaîtraient pas encore Google Reader, il s'agit d'un lecteur de feed RSS gratuit et disponible à l'adresse http://www.google.com/reader. La seule chose nécessaire pour pouvoir l'utiliser est de posséder un compte Google. L'intérêt de Google Reader, comme pour la plupart des outils fournis par Google, est d'exposer une API REST. Afin de comprendre le fonctionnement de l'API utilisée ici, je vous conseille la lecture des URLs citées en fin d'article.

L'authentification

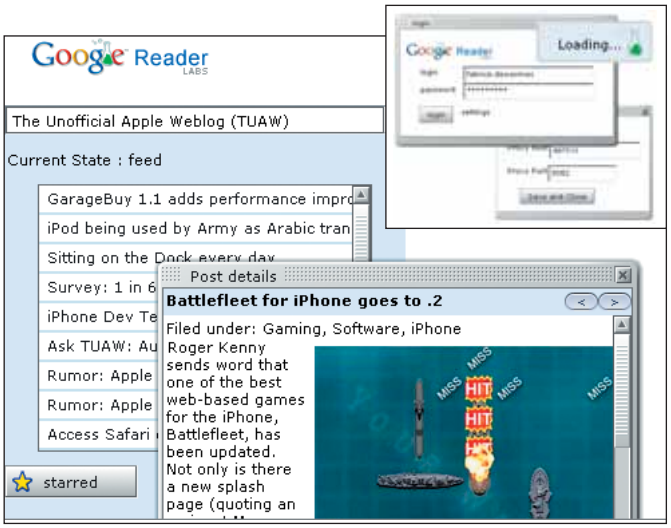
Si vous avez réalisé l'application HelloWorld de la première partie de cet article, nous allons juste créer dans le répertoire *my-apps* d'OpenLaszlo un nouveau répertoire nommé *reader* à l'aide d'Eclipse. Afin de structurer notre programme, nous allons décrire les éléments d'interface dans un fichier source nommé reader.lzx et créer un répertoire "data" dans lequel nous mettrons la partie modèle de notre application.
Commençons d'abord par nous authentifier sur Google Reader. Pour cela, nous allons créer une petite boîte de dialogue qui va permettre de recueillir les informations utilisateurs. Créez le fichier reader.lzx à la racine de votre répertoire reader et ouvrez-le à l'aide de l'éditeur spket. Comme pour Swing, il est nécessaire avant toute chose de créer un canvas sur lequel nous allons ensuite pouvoir "dessiner" les composants graphiques :

```
<canvas bgcolor="0xffffff" title="READER"></canvas>
```

Dans une application lzx, chaque composant peut être identifié soit par son attribut name qui a une portée locale, soit par son attribut id, qui a une portée globale.
Créons à présent la boîte de dialogue à l'aide du composant window :

```
<window id="loginView" x="474" y="160" title="login">
    <simplelayout axis="y" spacing="10" />
    <hbox spacing="10">
        <text y="10">login</text>
        <edittext id="login" width="200" />
    </hbox>
    <hbox spacing="10">
        <text y="10">password</text>
        <edittext id="password" password="true" width="200" />
    </hbox>
    <button>login</button>
</window>
```

Lancez votre navigateur et allez à l'URL http://localhost:8080/lps-4.0.5/my-apps/reader/reader.lzx. Après compilation, vous verrez une fenêtre intitulée 'login' comprenant deux champs de saisie précédés par un label et suivis par un bouton.



La mise en page des composants se base sur des positions absolues de composants (comme on le voit pour window) mais aussi sur des layout comme dans swing. Ici, nous avons utilisé une combinaison de *simplelayout* qui empile juste les composants soit sur l'axe des x, soit sur celui des y en intercalant une marge, et *hbox* qui empile les éléments qu'il contient de façon horizontale.
Affectons maintenant un comportement au bouton. Pour cela, créons d'abord un nouveau fichier lzx nommé *data.lzx* et situé dans un répertoire *classes*. Ce fichier va être une librairie d'accès aux données et à ce titre nous allons faire en sorte qu'elle soit incluse une et une seule fois dans l'application, au moment de l'application, en utilisant le tag library. Nous allons ensuite définir un jeu de variables globales pour disposer des URLs d'appel de l'API Google. Pour cela, nous utilisons un tag *node* avec un attribut *id*. Le tag node se trouve au sommet de la hiérarchie d'objets d'OpenLaszlo : il est l'équivalent de Object. Voilà donc ce que nous donne le fichier data.lzx pour le moment :

```
<library>
    <node id="gGlobals">
        <attribute name="SID" type="string" value="" />
        <attribute name="google_url" type="string" value="http://www.
google.com" />
        <attribute name="reader_url" type="string" value="${google_url
+'/reader'}" />
        <attribute name="login_url" type="string" value="${google_url +'/
accounts/ClientLogin'}" />
        <attribute name="state_url" type="string" value="${reader_url +'/
atom/user/-/state/com.google/'}" />
    </node>
</library>
```

Notez l'utilisation des *constraints* OpenLaszlo par l'usage de la syntaxe *${}*. Nous arrivons à la partie la plus intéressante : il nous faut à présent créer un *dataset,* c'est-à-dire un jeu de données qui sera récupéré d'une source de données atteinte en http. Pour cela, rien de plus simple :

```
<dataset name="tokends" type="http" ondata="gDataMan._handleToken()" onerror="gDataMan.handleError()" />
```

Implicitement, dataset va créer un objet *datasource* qui sera chargé de réaliser la requête http et stocker les données retournées dans le dataset. Un *datapointer* est ensuite lié au dataset, permettant ainsi de se promener dans les données afin de récupérer la donnée souhaitée. Dès que les données sont récupérées, la méthode spécifiée dans l'attribut *ondata* est appelée permettant d'enchaîner un traitement à la suite de la récupération des données. Mais avant de les traiter, réalisons la méthode qui va lancer la récupération des données, dans notre cas une requête d'authentification.

```
<method name="getSID" args="ds, login, password">
    <![CDATA[
    // clear query params
    ds.setQueryParams(null);
    ds.setQueryType("POST");

    ds.setSrc(gGlobals.login_url);

    ds.setQueryParam("service", "reader");
    ds.setQueryParam("Email", login);
    ds.setQueryParam("Passwd", password);
    ds.setQueryParam("source", "scroll");
    ds.setQueryParam("continue", "http://www.google.com");
    // make actual request
    ds.doRequest();
    ]]>
</method>
```

Plusieurs choses sont importantes ici :

- tout d'abord nous avons fait usage du tag *method* qui permet de définir une méthode dont le nom est précisé par l'attribut *name* et dont la liste des arguments se trouve dans l'attribut *args*. Parmi les attributs passés, le premier est une référence sur un dataset.
- Nous avons précisé à la datasource de faire la requête en POST en utilisant la méthode *setQueryType* de l'objet dataset.
- Nous avons positionné les paramètres de la requête en utilisant la méthode setQueryParam de dataset
- Nous avons lancé la requête

Mais comment connaître toutes ces méthodes utilisées ici ? Simplement en consultant la documentation d'OpenLaszlo : http://localhost:8080/lps-4.0.5/docs/reference. Par ailleurs, notez aussi l'usage des sections CDATA car rappelons-nous que nous travaillons dans du XML et que potentiellement certains caractères doivent être échappés. A présent, il nous faut traiter le résultat de la requête en créant la méthode *_handleToken* que va appeler le dataset sur réception de données :

```
<method name="_handleToken">
    <![CDATA[
    content=tokends.getPointer().xpathQuery("/text()");
    tokens = content.split("\n");
    for (i=0;i<tokens.length;i++) {
        var start = tokens[i].indexOf("SID=");
        if (start == 0) {
            SID = tokens[i].substring(start+4);
            break;
        }
    }
    ]]>
</method>
```

Ici, nous récupérons le datapointer associé au dataset afin de récupérer une donnée précise. Dataset étant un objet permettant de gérer des données XML, il est possible d'en récupérer d'un endroit précis du flux XML à l'aide d'une expression *xpath*. Attention cependant à deux choses à ce niveau là : d'une part OpenLaszlo ne fournit un support que partiel de xpath et il n'est pas rare de se trouver coincé lors de la navigation dans un arbre un peu complexe, d'autre part, le lancement d'une requête xpath retourne un résultat sans changer la position du pointeur de données. Ainsi, si vous souhaitez naviguer dans l'arbre de données, il vous faudra utiliser des méthodes de datapointer telles que *selectChild, selectNext, selectPrev,* etc...
Voilà ! Il ne nous reste plus qu'à connecter cette mécanique à l'interface utilisateur. Pour cela, modifions le code du bouton de la fenêtre de login comme suit :

```
<button onclick="gDataMan.getSID(tokends,login.getValue(),password.getValue())">login</button>
```

Notez comme les variables globales peuvent être accédées directement, notamment dans le cas du login et du password. Mais aussi, n'oublions pas d'inclure notre librairie dans le fichier reader.lzx en ajoutant juste après *canvas* :

```
<include href="classes/data.lzx" />
```

En route vers le feed !

Jusqu'à présent, nous n'avons fait que la partie qui nous permet de récupérer un SID Google, c'est-à-dire un jeton qui permettra de s'identifier auprès de Google pour les appels suivants. Créons donc la méthode qui va permettre d'émettre les requêtes à l'API. Pour cela, il faudra que chaque requête passe un cookie contenant le SID récupéré par la phase d'authentification :

```
<method name="gRequest" args="ds,url">
    <![CDATA[
    // clear query params
    ds.setQueryParams(null);
    ds.setQueryType("GET");
    ds.setHeader("Cookie", "Name=SID;SID="+SID+";Domain=.google.com;Path=/;Expires=160000000000");
    ds.setSrc(url);
    // make actual request
    ds.doRequest();
```

Directeurs, Responsables informatiques,
recevez gratuitement
le NUMERO 1

Un nouveau mensuel paraît ***fin janvier*** *:*

- *Un magazine centré sur les logiciels en entreprise :*
- *Comment les choisir, les déployer, les exploiter…*
- *Toutes les questions que se posent les décideurs : directeurs informatiques et responsables.*

Car les entreprises, quelle que soit leur taille, reposent en partie sur les applications informatiques.

Cela méritait bien un magazine !

Si vous êtes Responsable ou Directeur Informatique, demandez GRATUITEMENT le NUMERO 1

Répondez en ligne sur www.solutions-logiciels.com

par fax : 01 55 56 70 20 ou en envoyant ce coupon à :

Solutions-Logiciels, service Diffusion, 22 rue rené Boulanger, 75472 PARIS

☐ ***Oui****, envoyez-moi gratuitement le N°1 – réservé à la France métropolitaine (écrire en lettres capitales)*

NOM ..... Prénom .....

Fonction ☐ Directeur informatique ☐ responsable informatique

Titre ..... Société .....

N° ..... rue .....

Complément .....

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

Adresse mail ..... @ .....

c'est aussi en ligne sur www.solutions-logiciels.com

Editeur : K-Now sarl au cap de 8000€ siège social : 6 rue Bezout, 75014 Paris

```
    ]]>
</method>
```

Cette méthode est suffisamment générique pour permettre tous les autres appels à l'API de Google. En revanche, il est nécessaire de créer un dataset pour stocker les différents types de résultats reçus. Dans le cadre de l'article, nous allons nous contenter d'afficher les articles marqués d'une étoile mais le code téléchargeable va un peu plus loin, permettant de consulter n'importe quel feed auquel vous avez souscrit dans Google Reader.

```
<dataset name="starred" type="http" />
```

Revenons à présent à notre fichier définissant notre interface et ajoutons un composant qui affichera la liste des articles. Pour cela, nous allons utiliser le composant *list* qui permet d'afficher une liste d'autres composants et notamment des composants de type texte : les *textlistitem.* Comme mentionné dans l'article précédent, une des grandes forces d'OpenLaszlo est sa mécanique de binding de données aux composants. Voyons donc comment attacher notre composant *list* aux données récupérées depuis Google : chaque composant supporte l'attribut *datapath* qui permet de manipuler un *datapointer* sur une source de donnée.

```
<list id="feedList" width="250" height="400">
    <textlistitem id="itemList" datapath="starred:/feed/entry/" text="$
path{'title/text()'}" />
</list>
```

Ici *textlistitem* pointe sur la datasource *starred* avec le chemin xpath "/feed/entry". Comme ce chemin retourne plusieurs occurences, un *textlistitem* va être répliqué pour chaque résultat et sur chaque résultat le texte affiché est défini par le chemin xpath " title/text() ", qui est relatif au datapath du parent.
Mais pour l'instant, la datasource starred ne contient aucune donnée, c'est pourquoi il faut l'alimenter, et pourquoi pas le faire tout de suite après l'identification de l'utilisateur ? Ajoutons donc une ligne à la fin de notre méthode _handleToken :

```
gDataMan.gRequest(starred,gGlobals.state_url+'starred');
```

Ainsi dès l'authentification réussie, une nouvelle requête est émise, les données sont rapatriées et attachées au composant *list* qui se chargera de les présenter.
Il ne nous reste plus qu'à afficher le contenu de chaque article. Pour cela, nous utilisons toutes les connaissances accumulées jusqu'à présent pour ajouter un composant window :

```
<window id="detailView" title="Post details" x="400" width="500"
height="400" y="50" bgcolor="white" resizable="true" visible="false"
closeable="true">
    <datapath id="dpath" />
    <view id="scrollView" width="100%" height="${detailView.height -
detailView.inset_top - detailView.inset_bottom}" clip="true">
        <text datapath="summary/text()" resize="true" multiline="true" id
="details" width="100%"></text>
        <scrollbar id="sc" mousewheelactive="true" focusview="${parent.
parent}" usetargetsize="true"/>
    </view>
</window>
```

Cette fois nous avons utilisé un composant datapath indépendant, pour l'identifier de façon unique et pouvoir ainsi modifier dynamiquement les données pointées par la fenêtre. Le texte de la fenêtre est quant à lui toujours le résumé de l'article et sera toujours au même emplacement *xpath,* par conséquent le *datapath* du composant *text* est positionné à *summary/text()*. Cela pré-suppose bien entendu que la fenêtre ne fera qu'afficher des articles provenant d'un flux en XML toujours formaté de la même façon ! Affectons enfin un appel de méthode en réponse à une sélection d'un article dans la liste en associant au listener *onmouseup* de notre composant *textlistitem* l'appel à la méthode que nous allons écrire juste après :

```
onmouseup="detailView.showDetails()"
```

Il nous faut donc maintenant créer la méthode *showDetails* dans notre composant window.

```
<method name="showDetails"><![CDATA[
    p = feedList.getItem(feedList.getValue()).datapath.p;
    dpath.setPointer(p);
    //trick to force the text size to be recomputed
    details.setHeight(scrollView.height);
    if (!detailView.visible){
        detailView.setVisible(true);
    }
]]></method>
```

Ici, nous récupérons d'abord la valeur sélectionnée par l'utilisateur dans la liste pour pouvoir récupérer le textlistitem concerné ainsi que le datapointer. Nous donnons ensuite ce pointer au datapath de la fenêtre de présentation des détails de l'article. Ainsi le *datapath* de la fenêtre est devenu identique au *datapath* du *textlistitem* sélectionné par l'utilisateur, ce qui permet au champ *text* contenu dans la fenêtre de pouvoir pointer sur le chemin xpath " summary/text() " de l'entrée sélectionnée dans le feed.
Ouf ! Nous en avons fini mais notez comme l'interface reste sommaire et bien sûr il nous faut encore nettoyer quelques petites choses comme la mise en page ou encore l'affichage de la fenêtre de login. Bien sûr, pour rendre l'interface plus attractive nous pouvons ajouter quelques animations aux composants. Tout ceci est fait dans le code téléchargeable à l'adresse http://fdewasmes.free.fr. A vous maintenant d'enrichir cela.

Quelques liens :

http://www.niallkennedy.com/blog/archives/2005/12/google_reader_a.html
http://code.google.com/p/pyrfeed/wiki/GoogleReaderAPI
http://gpowered.blogspot.com/2007/08/google-reader-api-functions.html

Fabrice Dewasmes
Java & Open Source Dept. Manager
PragmaConsult SA
www.pragmaconsult.lu
http://fdewasmes.free.fr

Introduction à la programmation fonctionnelle avec Haskell

1re partie

Dédaignée, pour ne pas dire méprisée en France, la programmation fonctionnelle n'en est pas moins un outil puissant dans l'arsenal du programmeur. Examinons quelques uns des ses atouts à travers un langage à la pointe du progrès : Haskell

Haskell est le plus élégant, le plus beau, de tous les langages. L'affirmation est osée, et bien sûr, subjective. Et puis comment un langage informatique peut il être élégant ou beau ? Nous allons tenter de répondre à cette interrogation. Mais intéressons nous d'abord à une autre question: qu'est-ce qu'un langage informatique ? Rappelons-nous l'aube des temps informatiques. Pour programmer, il fallait perforer manuellement des cartes. Ou bien entrer un à un des octets de code machine dans une mémoire, calculer manuellement les adresses de sauts, etc. Bref programmer n'était pas vraiment programmer, mais plutôt s'empêtrer dans les mécanismes internes de l'ordinateur. Invivable... Alors on a inventé Fortran pour ne plus perforer manuellement les cartes. Pour l'anecdote, aujourd'hui encore, des particularités de la syntaxe de Fortran ont pour raison d'être la constitution physique de ces cartes :) Pour ne plus calculer manuellement les adresses de saut on a inventé l'assembleur. Assembleur et Fortran ont un but commun: s'abstraire des mécanismes internes de la machine. Programmer en assembleur reste incontestablement laborieux. Alors on a inventé le langage C (et d'autres). Avec C le niveau d'abstraction est plus important et le programmeur travaille plus confortablement. Dans une certaine limite toutefois. C reste très proche de la machine. Programmer en C c'est encore beaucoup subir la dictature des mécanismes internes. Alors on essaie d'élever le niveau d'abstraction. Par exemple, on dote des langages tels que Java ou C# d'un ramasse-miettes pour abstraire le programmeur de la fastidieuse gestion de la mémoire. Les machines deviennent "virtuelles". Virtuelle ou pas, les langages dont nous avons parlé, ainsi que tous ceux de la famille des langages dits impératifs (y compris objets), pilotent peu ou prou une machine, et pour ce faire découpent un programme en séquences et modifient des états.

Les langages fonctionnels, une autre famille de langages

Les langages fonctionnels ont pour intention d'abstraire complètement de la machine. Avec eux, un programme est un ensemble de fonctions au sens mathématique du terme et c'est le système qui s'occupe de l'évaluation d'un ensemble d'expressions, pas le programmeur. Pour ce faire, ces langages s'appuient sur un runtime évolué. C'est ainsi que Lisp, le premier des langages fonctionnel, et créé il y plus de cinquante ans, a inventé le ramasse-miettes que Java a aujourd'hui repris à son compte...

Haskell, un langage à la pointe du progrès

Dans la lignée des langages fonctionnels, Haskell est encore tout récent. Ses spécifications ont été fixées en 1998. C'est un langage à la pointe du progrès, doté d'une syntaxe épurée, fonctionnel pur (sans aucun effet de bord), totalement paresseux et doté à la fois d'un polymorphisme ad hoc et d'un polymorphisme paramétrique. Voilà pour la définition technique. Nous n'entrerons pas dans ces détails abscons. Disons plus simplement que tout avec Haskell est fonction, y compris les entrées/sorties qui sont effectuées dans un contexte purement fonctionnel, la Monade. Haskell c'est l'homogénéité. Les constructeurs de types sont fonctionnels, les types intégrés et les types utilisateurs se comportent de la même façon, etc. Cette homogénéité et cette absence d'effet de bord apportent un niveau d'abstraction incroyable ainsi que nous le verrons plus loin. Encore ignoré en France, pays où l'on a toujours un train de retard, Haskell est de plus en plus utilisé à l'étranger par des industriels qui ont compris les avantages qu'ils pouvaient en tirer en terme de temps de développement et de sûreté du code. C'est que comme avec beaucoup de langages fonctionnels, la correction d'un programme Haskell peut être prouvée. En comparaison les tests en C, Java, C++ ne font que mettre des bugs en évidence, mais hélas ne prouvent pas leur absence... Et plus généralement les défauts inhérents à tous les langages impératifs font du langage lui-même une partie du problème à résoudre plutôt qu'une partie de la solution.

Haskell n'est pas un langage à tout faire.

Doté de librairies standard riches: réseau, graphiques, OpenGL, concurrence, etc., Haskell n'a toutefois pas la prétention de savoir tout faire. S'il est question de piloter un périphérique c'est évidemment C (ou C++ ou Pascal ou Ada, etc.) que l'on retiendra puisque c'est sa vocation. Par contre, s'il est question d'écrire une application scientifique, bancaire, réseau, exemples non limitatifs, Haskell convient. On peut trouver des défauts à Haskell. Ses performances excellentes sont en dessous de C en raison notamment du mécanisme d'évaluation paresseuse. Il a tout ce qu'il faut pour supporter Unicode mais les librairies sont encore en cours d'écriture. Le débogage peut s'avérer difficile en raison de l'évaluation paresseuse. Mais Haskell est tellement sûr qu'un programmeur n'éprouve le besoin de déboguer que très exceptionnellement. Le problème majeur est l'apprentissage. Ce n'est pas que Haskell soit difficile à apprendre ou à manipuler par lui-même. La difficulté réside dans les habitudes prises par les programmeurs rompus aux langages impératifs. Aborder Haskell à la manière d'un langage impératif, c'est se heurter à un mur. Il est nécessaire de l'aborder avec l'esprit ouvert et d'en saisir la philosophie. J'ai commencé ma carrière en programmant des automates et de l'assembleur. Alors ces difficultés je les connais bien :) Elles ne sont finalement pas si difficiles à surmonter. Et la pratique m'a montré que dès qu'on a acquis un peu de savoir-faire, on se demande comment diable on a pu travailler avec ces tas de silex que sont C, C++, Java, et les autres. Je vous propose maintenant de voir ce que abstraction veut dire avec Haskell.

Un peu de code, beaucoup d'abstraction

Pour bien évaluer un langage, rien de tel que d'écrire un peu de code et comparer avec le code équivalent écrit avec d'autres langages. Posons un problème simple. Nous voulons une fonction qui recevant une liste (ou équivalent selon le langage), construise et retourne une liste telle que ses éléments soient ceux de la liste reçue en argument, additionnés à une valeur également reçue en argument. L'idéal est que la fonction soit générique. Nous choisissons C++ et nous écrivons d'abord un premier jet :

```
#include <vector>
#include <algorithm>
#include <functional>

std::vector<int> f(const std::vector<int>& v, int valeur)
{
   std::vector<int> nv(v.size());
   std::transform(v.begin(), v.end(),
         nv.begin(),
         std::bind1st(std::plus<int>(), valeur));
   return nv;
}
```

Nous avons 10 lignes de code pas vraiment faciles à lire. Ne parlons pas de concision. Voici maintenant la fonction écrite en Haskell :

```
f valeur liste = map (+ valeur) liste
```

Une seule ligne de code. Et encore, fort verbeuse... :) Car un haskeller averti, appliquera la règle du currying et écrira seulement :

```
f valeur = map (+ valeur)
```

Nous pouvons maintenant parler de concision :) Le code C++ fait-il au moins ce qu'on lui demande ? Rien n'est moins sûr. Nous y reviendrons, mais écrivons d'abord des fonctions génériques. En C++:

```
#include <vector>
#include <algorithm>
#include <functional>

template <class T> std::vector<T> ft(const std::vector<T>& v, T valeur)
{
   std::vector<T> nv(v.size());
   std::transform(v.begin(), v.end(),
         nv.begin(),
         std::bind1st(std::plus<T>(), valeur));
   return nv;
}
```

Le code n'a pas gagné en clarté. En Haskell maintenant:

```
f valeur = map (+ valeur)
```

Et oui, c'est exactement la même chose que plus haut. Notre ligne de code, toute simple, était déjà générique. Essayons un peu notre code. En C++ :

```
int data[] = {1, 2, 3};
std::vector<int> v(data, data+3);

std::vector<int> nv = f(v, 10.5);
std::vector<int> nvt = ft<int>(v, 10.5);
```

A l'issue de quoi les vector nv et nvt contiendront tous deux les valeurs 11, 12 et 13. Seulement voilà 1+10.5 cela ne fait pas 11. Que s'est il passé ? Nous avons dit à C++ que nous travaillons avec des int. Or nous lui passons une valeur de type double (10.5). Qu'à cela ne tienne, tout générique qu'il est, C++ fait entrer "au chausse-pied" la valeur dans le type int. Peut être le compilateur émettra-t-il un avertissement selon les options de compilations sélectionnées, mais rien, dans le standard C++, ne l'oblige à le faire. Peut-être que le résultat de l'opération est bien conforme à ce que nous souhaitions, ou peut-être avons-nous simplement perdu des données à la suite d'une erreur de programmation. Ce genre d'erreur d'inattention se produit fréquemment et C++ valide sans état d'âme. Bon admettons le, nous avons triché. Travaillons donc avec des valeurs entières :

```
unsigned int data[] = {1, 2, 3};
std::vector<unsigned int> v(data, data+3);

std::vector<unsigned int> nvt = ft<unsigned int>(v, 4294967294U);
```

Cette fois nvt contient les valeurs 4294967295, 0 et 1. En raison d'un débordement de capacité d'un registre 32 bits, notre calcul est encore faux! Les programmeurs expérimentés que vous êtes savent bien que ce type d'erreur est hélas lui aussi très fréquent. En raison de son manque de sûreté, nous concluons que C++ fait partie de notre petit problème, pas de sa solution. Voyons avec Haskell :

```
f 10.5 [1, 2, 3]
[11.5,12.5,13.5]
```

Nous n'avons rien dit à Haskell quant au type. De lui-même il infère le bon type, le type Float. Nous n'avons pas perdu de valeur. Mais peut être voulions nous travailler avec des Int. Voyons comment cela fonctionne: notre fonction est générique. Voici sa signature :

```
f :: (Num a) => a -> [a] -> [a]
f valeur = map (+ valeur)
```

Cela signifie que f fonctionne pour tout type de la classe Num(érique), c'est-à-dire ici et concrètement tout type, y compris un type utilisateur, supportant l'opération d'addition. Mais nous pouvons souhaiter spécialiser la fonction à des valeurs entières :

```
f :: (Integral a) => a -> [a] -> [a]
f valeur = map (+ valeur)
```

A partir de maintenant l'appel f 10.5 [1, 2, 3] sera rejeté par le compilateur. Et si nous donnons :

```
f 4294967294 [1, 2, 3]
```

nous obtenons le résultat correct :

```
[4294967295,4294967296,4294967297]
```

Nous constatons qu'en plus d'être concis Haskell dispose d'un système de types extrêmement sûr. En béton armé même :) Nous concluons que Haskell fait, lui, partie de la solution à notre petit problème :) Quand nous écrivons notre fonction f en Haskell, nous n'avons à nous soucier

ni du déroulement de son code ni d'éventuelles pertes de valeurs. Nous sommes affranchis des tracas inhérents à la machine.

Toujours plus d'abstraction

Varions les déplaisirs et écrivons des fonctions factorielles en Java. Une fonction est récursive et l'autre non. Pour ne pas alourdir le discours, nous admettons que les fonctions seront appelées avec un entier strictement positif en argument.

```
public class Facs {
    public static int facRec(int valeur) {
        if(valeur==1)
            return 1;
        else
            return valeur*facRec(valeur-1);
    }
    public static int fac (int valeur) {
    int result = 1;
    for(int i=2; i<= valeur; i++)
        result *= i;
    return result;
    }
}
```

Ce code verbeux fonctionne-t-il ? Nullement. L'appel facRec(15000); provoque une exception StackOverFlowError. L'appel fac(15000); retourne la valeur zéro ce qui ne semble pas très exact. Fondamentalement nous avons des erreurs dans l'esprit de celles rencontrées plus haut avec C++. La tracasserie du débordement de pile est due à la machine, que celle-ci soit affublée du nom de virtuelle n'y change rien. Quant à la valeur fausse, c'est le même débordement de capacité que celui rencontré précédemment. Comparons maintenant notre code Java avec Haskell d'un autre point de vue. Dans notre code Java, nous sommes extrêmement dirigistes. En plus d'exprimer l'algorithme proprement dit, nous avons codé la séquence d'exécution dans le détail. Voyons comment cela peut se passer avec Haskell :

```
facrec valeur =
    if valeur == 1 then 1
    else valeur* facrec (valeur-1)
```

Ce code, extrêmement maladroit, a au moins le mérite de fonctionner et est capable de retourner le résultat de facrec 15000, nombre astronomique certes, mais calculé. Pourquoi ce code est il maladroit ? Parce qu'il est écrit dans un style impératif, comme si nous voulions avoir la main mise sur le flux d'exécution, ce qui est contraire à la philosophie du langage. Haskell est un langage déclaratif. Déclarons donc que dans le cas où valeur vaut 1 notre fonction s'évalue à 1 et que sinon elle s'évalue à valeur multipliée à elle-même invoquée avec valeur-1 en argument :

```
facrec 1 = 1
facrec valeur = valeur * facrec (valeur-1)
```

Plus de if ni de else. Nous nous sommes concentrés uniquement sur la définition de la fonction et nous laissons au compilateur le soin de construire la séquence d'exécution. Il y a toutefois bien mieux à faire.

Quand, humainement, on pense "factorielle n", on pense au produit de (la liste de) tous les entiers de 1 à n. Tel on le pense, tel on l'écrit en Haskell :

```
fac n = product [1..n]
```

Nous avons là un grand moment d'informatique. Aucune contrainte due à la machine, aucune gestion de la séquence d'exécution, rien. Nous avons simplement exprimé la chose telle qu'elle est en une ligne et c'est réglé. Techniquement, cela est possible grâce à l'évaluation paresseuse de Haskell qui ne construira jamais la liste des entiers de un à n mais évaluera ceux-ci au fur et à mesure des besoins. A nouveau le langage fait partie de la solution au lieu de faire partie du problème. Il est aussi, accessoirement, une solution au problème des barrettes de Ram ;)

Infiniment plus d'abstraction

Haskell permet d'aller infiniment plus loin, c'est le cas de le dire, que ce que nous venons de voir avec notre fonction fac. En effet grâce à l'évaluation paresseuse, nous pouvons manipuler des structures de données infinies. Pour illustrer cela, posons nous un autre petit problème. Nous voulons travailler avec une liste d'entiers tels que ceux-ci soient tous les carrés d'entiers impairs. Cette liste est donc infinie. Nous voulons être capables d'afficher par exemple les 5 premiers éléments de cette liste, de tester si une valeur quelconque fait partie de cette liste et nous voulons pouvoir afficher, par exemple, le 1000e élément de cette liste. Voici le code, simple, qui fait tout cela :

```
ensembleN = [0,1..]

impairs = filter odd ensembleN

impairs_carres = [x*x | x <- impairs]

est_impair_carre n (x:xs)
    | n < x = False
    | n == x = True
    | otherwise = est_impair_carre n xs


main = do
    putStr $ (show $ take 5 impairs_carres) ++ "\n"
    putStr $ (show $ est_impair_carre 121 impairs_carres) ++ "\n"
    putStr $ (show $ est_impair_carre 83 impairs_carres) ++ "\n"
    putStr $ (show $ impairs_carres !! 999) ++ "\n"
```

A l'exécution ce code affiche :

```
[1,9,25,49,81]
False
True
3996001
```

Bien qu'il puisse être facilement réduit de moitié, ce code n'en est pas moins fascinant en l'état. La première ligne déclare rien moins que l'ensemble des entiers naturels. D'ailleurs la déclaration ensembleN = [0..] aurait suffit :) Toujours est il que ensembleN est un ensemble de données infini. Pourtant nous pouvons passer ensembleN en argument à une autre fonction comme si de rien n'était! C'est d'ailleurs ce que nous

faisons dans la fonction "impairs" qui filtre, au moyen du prédicat odd tous les entiers impairs. "impairs" est donc l'ensemble, infini lui aussi, des entiers impairs. Enfin nous déclarons un troisième ensemble, lui aussi infini, celui des carrés des nombres impairs. Pour ce faire, nous utilisons une liste compréhensive qui construit la liste de tous les carrés des valeurs prises dans l'ensemble des valeurs impaires. Nous manipulons ainsi, l'air de rien, pas moins de trois ensembles infinis. Enfin nous déclarons une fonction "est_impairs_carres" qui teste si un nombre passé en argument appartient à la liste impairs_carres également passée en argument. Nous procédons en comparant le nombre à la tête de la liste. Pour cette fonction, nous déclarons simplement que si le nombre est inférieur à la tête de la liste, il n'appartient pas à cette liste. Nous déclarons que si le nombre est égal à la tête de la liste, il appartient à la liste, et dans tous les autres cas nous déclarons devoir tester notre nombre sur la queue de la liste.

Ce qui est remarquable, avec tout ce code, qu'un Haskeller expérimenté pourrait écrire bien mieux encore, c'est qu'à aucun moment nous ne nous sommes inquiétés de contraintes imposées par la machine. Nous ne nous sommes pas inquiétés le moins du monde du déroulement de ce programme. Totalement affranchis, libres, nous avons directement manipulé des concepts, des idées. On objectera sans doute qu'en informatique industrielle "pratique" on travaille rarement avec des ensembles de nombres comme ici. Imaginons simplement un nombre astronomique de possibilités que l'on manipulera comme une seule entité, que l'on filtrera pour examen dans un logiciel d'aide à la décision, dans un jeu de stratégie, etc. Avec Haskell, l'ordinateur s'efface et la créativité du programmeur s'exprime à volonté. Voilà pourquoi votre serviteur pense que Haskell est le plus élégant, le plus beau de tous les langages, et que le paradigme fonctionnel mérite vraiment d'être employé.

Un langage pour des tâches concrètes

Aussi fascinantes que soient l'évaluation paresseuse et la manipulation de types infinis, il serait dommage d'en déduire que Haskell est un langage de mathématiciens, ou limité à l'intelligence artificielle et somme toute assez loin de la programmation "quotidienne". C'est tout le contraire. Au quotidien Haskell apporte le gain de temps de développement et la facilité de maintenance du code par son expressivité et sa lisibilité. Prenons un nouvel exemple pour illustrer cela: le quicksort, ou tri rapide. Voici ce que cela donne en C (code pris sur http://www.akropolix.net/):

```
qsort(int a[], int lo, int hi )
{
  int h, l, p, t;

  if (lo < hi) {
      l = lo;
      h = hi;
  p = a[hi];

  do {
      while ((l < h) && (a[l] <= p))
          l = l+1;
      while ((h > l) && (a[h] >= p))
          h = h-1;
      if (l < h) {
          t = a[l];
          a[l] = a[h];
          a[h] = t;
      }
  } while (l < h);

  t = a[l];
  a[l] = a[hi];
  a[hi] = t;

  qsort( a, lo, l-1 );
  qsort( a, l+1, hi );
  }
}
```

Voici maintenant une implémentation possible en Haskell :

```
qsort []     = []
qsort (x:xs) = qsort elts_lt_x ++ [x] ++ qsort elts_gt_x
        where
            elts_lt_x   = [y | y <- xs, y < x]
            elts_gt_x   = [y | y <- xs, y >= x]
```

Le code C ne brille pas par sa concision et n'est pas facile à relire dans la mesure où l'algorithme de tri ne saute pas aux yeux. La version Haskell en revanche est la transcription directe de l'algorithme. Même sans pratiquer le langage on devine immédiatement que x est le pivot. Ou est le pivot dans le code C ? Au fait le code C ne sait trier qu'un tableau d'entier. Le code Haskell est générique ipso facto et triera toute liste de types comparables.

J'espère que vous avez eu plaisir à découvrir Haskell et que cet article vous a donné envie de vous intéresser à ce langage hors du commun, incroyablement expressif et puissant. Le mois prochain nous nous retrouverons pour quelques notions et maniements de base. Et si ces deux articles reçoivent un accueil favorable de votre part, nous irons plus loin.

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

Quelques ouvrages de références

Il existe quelques bons livres, malheureusement tous en anglais pour s'initier à Haskell et acquérir des notions avancées :

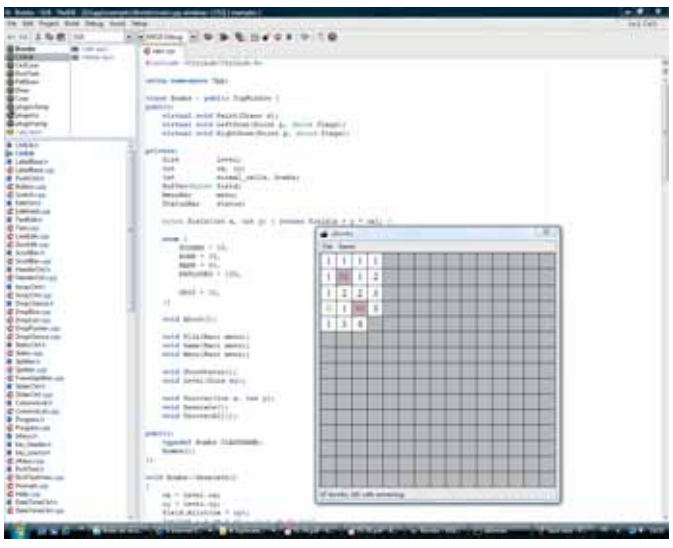
- **The haskell School of Expression,** Paul Hudak – Cambridge University Press
- **Haskell, The Craft of Functionnal Programming,** Simon Thompson – Addison Wesley
- **Algorithms, A Functionnal Programming Approach,** Rabhi & Lapalme - Pearson Education
- **Purely Functionnal Data Structures,** Chris Okasaki - Cambridge University Press

Ultimate++ : l'ultime IDE multi-plate-forme pour le développement d'applications C++

En raison du caractère natif de C++, développer des applications graphiques multi- plates-formes est un problème récurrent. Disposer d'un outil de développement efficace et multi plate-forme est également un souci. A tout cela Ultimate++ apporte une solution percutante, à base de beau code C++

Le développement d'applications graphiques C++ multi-plates-formes trouve sa solution dans des librairies conçues à cet effet. Par exemple, la librairie Qt est une solution excellente. Reste le problème de l'outil de développement multi-plate-forme. En ce qui concerne le travail avec C++, Eclipse ne donne pour l'instant, et de l'humble avis de votre serviteur, rien de vraiment bon. Des environnements tels que Code::Blocks sont sans doute prometteurs, mais ont encore besoin de maturité. En outre, ils sont de facture classique, pas vraiment innovante, et basés sur de l'existant lui-même non innovant. Ainsi, Code::Blocks, pour reprendre cet exemple, est construit sur wxWidgets, librairie certes multi plate-forme et qui a eu son heure de gloire, mais écrite dans du C++ "à la papa", tout rempli de macros et peu élégant, voire lourd. Ultimate++ se démarque catégoriquement de tout cela. A l'exception de l'inévitable Gtk+ qui fait office de librairie graphique native sous Linux, tout pour Ultimate++ a été pensé et écrit de zéro. Le code est du C++ ultra moderne à la manière de Boost ou de Blitz++, cette dernière étant d'ailleurs utilisée pour diminuer les temps de construction des projets.

Le résultat final est un outil de développement de conception originale, ultra performant et réactif, plein de trouvailles pratiques, et permettant de développer avec rapidité des applications graphiques C++ portables, au code extrêmement concis. Je vous invite à vous faire une idée de cela en visitant la page de comparatifs http://www.ultimatepp.org/www$uppweb$comparison$en-us.html. L'outil vient avec un riche choix d'exemples qui vous permettront de vous faire une idée des possibilités offertes par ses librairies. Ultimate++ est désormais parfait sous Windows. Il l'est presque, aux dires de ses développeurs, sous Linux. Je l'utilise cependant sous cette plate-forme sans le moindre ennui.



Un démineur écrit et exécuté sous Ultimate.

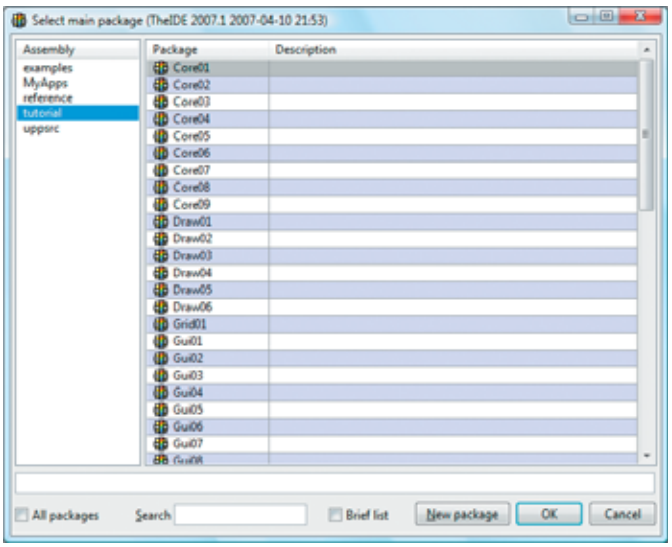


Fig.1 - La boîte de dialogue de sélection de package et aussi de création/configuration des assemblies.

Il le sera bientôt sous Mac et peut être plus tard sous d'autres plates-formes encore. Le projet Ultimate++ est remarquablement actif et accompagné d'un forum. Nous allons par cet article nous familiariser avec cet outil et son organisation, faire quelques manipulations non décrites dans la documentation, et nous allons essayer de comprendre quelques uns des remarquables mécanismes internes des librairies sous-jacentes.

1 Organisation générale

L'installation d'Ultimate++ est une formalité pour les informaticiens avertis que sont les lecteurs de Programmez!. Sous Windows, et si aucun compilateur C++ n'est présent sur votre système, choisissez l'archive d'installation tout-en-un qui intègre GCC pour Windows. A tout moment vous pouvez demandez à Ultimate++ de reconnaître et utiliser un ou plusieurs autres compilateurs. Lorsqu'Ultimate démarre pour la

première fois, il vous demande de définir un répertoire de travail, et le cas échéant de choisir un ou plusieurs compilateurs C++. Ultimate++ reconnaît et utilise parfaitement le compilateur de Visual Studio 2005. Lorsqu'Ultimate++ démarre normalement, il vous demande de sélectionner un package dans un assembly. Parmi les assemblies, vous reconnaîtrez le nom du répertoire de travail précédemment spécifié.
Le terme package rappelle Java, le terme assembly rappelle C# et les deux n'ont rien en commun ni avec Java ni avec C# :) Sous Ultimate++ un assembly est simplement un répertoire à la racine d'une arborescence. Un assembly Ultimate++ présente donc une forte analogie avec une solution Visual Studio ou un Workspace Eclipse.
Le package quant à lui présente une forte analogie avec un projet Visual Studio ou Eclipse. Les packages sont autant de répertoires sous le répertoire racine qu'est l'assembly. Le package contient tous vos sources (.cpp, .h et autres) ainsi qu'un fichier .upp qui contient la définition du package et ses dépendances, c'est-à-dire les autres projets/packages que votre projet/package utilise, par exemple le corps d'une application principale dépendant de plusieurs librairies partagées.
Ce fichier .upp est relu à la volée à chaque fois que vous cliquez sur un nom de package dans l'IDE, et vous avez ainsi un accès immédiat aux sources des packages dont vous dépendez. Mieux que cela, si votre application dépend de packages "systèmes", c'est-à-dire des librairies fournies par Upp, vous avez, en un seul clic de souris, accès à tous ces sources, ce qui, à l'utilisation, se révèle formidable.

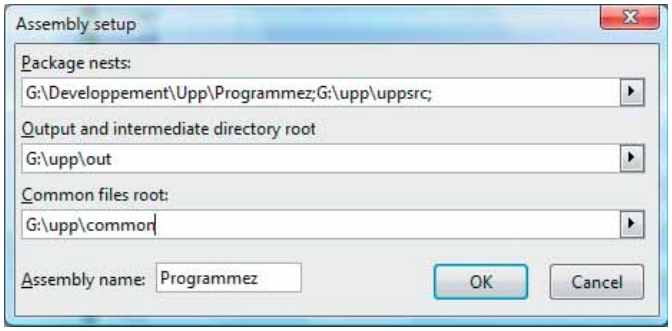


Fig.2 - Configuration de notre assembly

2 Un projet multi-plate-forme

Pour mieux comprendre tout cela, nous allons faire ensemble un petit exemple. Quand on utilise les librairies fournies par Ultimate++, le code de l'application est directement compilable sur n'importe quelle autre plate-forme. Cependant, on peut très bien imaginer qu'une application ait à utiliser une fonctionnalité spécifique à Windows d'un côté et spécifique à Linux de l'autre côté. La documentation n'explique pas en détail comment faire dans ce cas. Quand on a compris la philosophie d'Ultimate++ c'est toutefois très facile. Nous commençons le travail sous Windows. Pour faire les choses bien, nous allons d'abord créer un assembly qui contiendra tous nos essais. Ultimate++ étant lancé, allez dans le menu "File|Set main package...". Une boîte de dialogue apparaît comme illustré (Fig.1). Dans la partie gauche figurent les assemblies existants, dans la partie droite les packages rangés sous l'assembly sélectionné dans la partie gauche. Cliquez avec le bouton droit de la souris sur le panneau de gauche, et choisissez "New assembly" dans le menu surgissant. S'ouvre alors un second dialogue qu'il faut renseigner

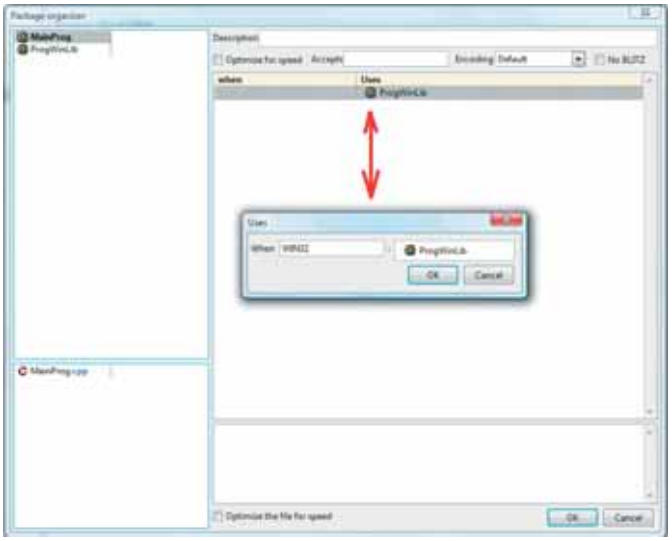


Fig.3 - Configuration de l'utilisation conditionnelle d'un package

avec le plus grand soin, comme illustré (Fig.2). Tout en bas du dialogue (champ "Assembly name") nous donnons le nom de notre assembly, soit Programmez. Pour les deux champs du milieu nous gardons les valeurs par défaut et nous concentrons toute notre attention sur le champ supérieur, "Package nests". Ce champ doit contenir en premier le répertoire associé à l'assembly. Si ce nom de répertoire ne figure pas en premier, vous pourrez avoir des difficultés à accéder à votre package au sein de l'IDE. Ensuite, doivent venir les noms des répertoires racines d'autres assemblies. Autant dire que "Package nests" aurait été mieux nommé "Assemblies nests". Si vous utilisez des libraires Ultimate++ dans vos projets, ce qui sera a priori souvent le cas, vous devez impérativement ajouter c:\upp\uppsrc à la liste, en supposant que Ultimate++ soit installé sous c:\upp.
Ce répertoire contient les sources des librairies systèmes. Si ce répertoire est absent, vous ne pourrez pas ajouter des packages systèmes à votre projet, ce qui en clair signifie que vous ne pourrez pas travailler correctement, ni peut être même compiler. Validez la création de votre package. Vous revoilà dans la première boîte de dialogue. Cliquez sur le bouton "New package". Dans le dialogue qui apparaît, donnez le nom de votre package (autrement dit de votre projet) Par exemple MainProg, puis sélectionnez le modèle "Console application (no U++)" dans le panneau de gauche.
Remarque : sous Linux il est possible que les modèles soient absents, créez alors simplement un projet vide. Validez, vous arrivez enfin dans l'IDE. Son interface se divise en trois parties. La partie à droite concerne évidemment l'édition de code. La partie gauche de divise en deux panneaux. Celui du haut liste les packages dont dépend le projet. Pour l'instant il n'y en a qu'un, celui que nous venons de créer. Et son contenu, à savoir un seul fichier .cpp, est listé dans le panneau inférieur. Lorsqu'un projet dépend de plusieurs packages, cliquer sur l'un de ceux-ci dans le panneau supérieur affichera tout son contenu dans le panneau inférieur. Il est temps d'écrire un peu de code :

```
#ifdef flagWIN32
   #include <ProgWinLib/ProgWinLib.h>
#endif
```

```
#ifdef flagLINUX
   #include <ProgLinuxLib/ProgLinuxLib.h>
#endif


int main(int argc, const char *argv[])
{
   helloworld();
   return 0;
}
```

Il s'agit d'un helloworld non portable :) Nous supposons que la fonction helloworld dépend de fonctionnalités propres au système hôte. Nous la reléguons dans une librairie partagée dont nous ferons une version pour chaque système. Ultimate++ est suffisamment intelligent pour compiler le tout selon la plate-forme, moyennant un tout petit peu de configuration. Le code d'exemple montre une directive de compilation conditionnelle pour l'inclusion de l'en-tête de la librairie partagée selon la plate-forme hôte.

La question est: où et comment sont définies les macros flagWIN32 et flagLINUX ? Avant d'y répondre, créons nos librairies partagées. Pour cela, sélectionnez "File|Set main package..." dans le menu de l'IDE et créez un package sous le nom de ProgWinLib avec le modèle Win32 DLL project (no U++). Rien de nouveau ici. Après, écrivez le code. Celui-ci, totalement sans intérêt, n'est pas donné ici. Nous renvoyons le lecteur au Cd-Rom. Revenez à MainProg. Pour cela, passez à nouveau par le menu "File|Set main package..." et sélectionnez 'MainProg' qui figure maintenant dans la liste du dialogue. Nous allons maintenant ajouter le package de la librairie partagée à notre projet et configurer la dépendance conditionnelle.

Dans l'IDE cliquez avec le bouton droit sur le nom du package courant (MainProg) et sélectionnez "Add package to MainProg...". Nous revoilà dans notre dialogue bien connu. Sélectionnez-y 'ProgWinLib'. De retour dans l'IDE, cliquez à nouveau sur MainProg avec le bouton droit, et cette fois choisissez "Package Organizer". Dans le dialogue qui s'ouvre, vous voyez que MainProg dépend de ProgWinLib.

Avec le bouton droit de la souris demandez l'édition de ligne qui décrit la dépendance et ajoutez WIN32 comme condition, comme illustré (Fig.3). Le WIN32 ici, est vu comme flagWIN32 dans le code C++. On procède de la même façon pour la librairie Linux. Nous renvoyons le lecteur au site pour le projet complet.

Nous en avons terminé, notre projet peut être porté tel quel sous Linux, repris par l'IDE, compilé et exécuté. En outre rappelons bien que ces manipulations, qui ont eu l'intérêt de faire manipuler et naviguer dans les packages, ne sont justifiées que parce que nous avons émis l'hypothèse de l'usage d'une fonctionnalité spécifique à une plate-forme. Sinon le portage des sources est direct, sans directive de dépendances conditionnelles.

Fig.4 - Une application graphique de démonstration.

Enfin, signalons que les flags de construction ne concernent pas que les plates-formes. Il en existe aussi des relatifs aux compilateurs connus. Nous renvoyons le lecteur à la documentation de l'IDE, chapitre "Builds flags".

3 Un aperçu de la qualité de la librairie graphique.

Cette librairie est écrite dans code du "C++ spirit" qui exploite les fonctionnalités les plus avancées du langage. Toute remplie de templates à la Alexandrescu et de surcharges d'opérateur, elle permet d'écrire un code client à la fois concis est clair, ce qui fait figure d'oiseau rare dans le monde des librairies graphiques C++.

Voici à titre d'exemple, un code écrit à partir du modèle "CtrLib application with main window", notre fenêtre étant sans layout, c'est-à-dire non conçue dans le concepteur graphique. Le résultat est illustré (Fig.4).

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;

class demowin2 : public TopWindow {
private:
   EditField pr, ab;
   Button bab, ex;
public:
   typedef demowin2 CLASSNAME;
   demowin2();
   void OnBab();
   void OnEx();
};

#include "demowin2.h"

demowin2::demowin2()
{
   SetRect(0, 0, 300, 200);
   CenterScreen();
   pr.SetText("Programmez!");
   pr.LeftPos(10).TopPos(10);
   ab.SetText("");
   ab.LeftPos(10).TopPos(40);
   bab.SetLabel("Clic");
   bab.LeftPos(10).TopPos(70);
   ex.SetLabel("Quitter");
   ex.LeftPos(100).TopPos(70);
   *this << pr << ab << bab << ex;
   bab <<= THISBACK(OnBab);
   ex <<= THISBACK(OnEx);
}

void demowin2::OnBab()
{
   ab.SetText("Abonnez vous!");
}

void demowin2::OnEx()
{
   Break(0); // 0 -> Sortie normale du programme
}

GUI_APP_MAIN
```

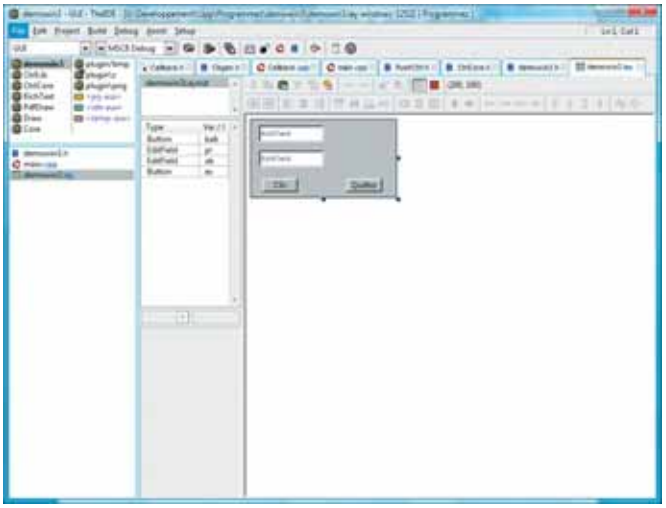


Fig.5 - Utilisation du concepteur graphique d'Ultimate++

```
{
   demowin2().Run();
}
```

Ce code parle de lui-même, même si on ne connaît pas la librairie et c'est ce qui est remarquable. Dans la déclaration de la classe nous avons en guise de composants 2 instances de EditField (zone de texte) et 2 instances de Button, comme membres de la classe, sans autre forme de procès. Les méthodes OnBab et OnEx seront les fonctions de rappels, ou Callbacks des boutons. Rien qui ne semble pas anodin dans cette classe. On notera cependant la déclaration d'un synonyme (instruction typedef) dont le rôle sera expliqué plus loin. Le corps du constructeur est également anodin, à l'exception des trois dernières lignes. Vient d'abord:

```
*this << pr << ab << bab << ex;
```

A ce stade, les classes Button et EditField sont bien sûr instanciées. Mais les composants ne sont pas encore visibles à l'écran. Le rôle de cette première ligne est d'injecter les composants graphiques (non les classes) dans l'interface.
Là où d'autres librairies n'ont que des méthodes "Add", qui existent d'ailleurs aussi sous Ultimate++, la surcharge de l'opérateur << apporte une concision incomparable. Vient ensuite :

```
bab <<= THISBACK(OnBab);
```

Cette ligne connecte le bouton à sa fonction de rappel. Les classes encapsulant les composants ont comme membres, des instances de la classe CallBack qui est un objet fonction généralisant les pointeurs de fonctions. La macro THISBACK instancie une classe Callback autour de notre méthode Bab. La macro est ainsi définie dans l'en-tête Core/Callback.h :

```
#define THISBACK(x) callback(this, &CLASSNAME::x)
```

Ce qui éclaire quant à la nécessité de la déclaration du synonyme mentionnée plus haut. Après quoi la surcharge de l'opérateur <<= du composant permet d'affecter son instance de Callback par la valeur instanciée par la macro. A l'utilisation, c'est tout simple. Le code de la librairie l'est un peu moins... mais très instructif. Or, comme on peut naviguer dedans à volonté il ne faut surtout pas s'en priver pour approfondir les mécanismes internes.

4 Conception graphique

Notre dernière illustration (Fig.5) montre la même application, mais avec un layout. Cela signifie que l'interface n'est pas écrite à la main, mais définie par un concepteur graphique qui génère les informations nécessaires dans un fichier .lay dit fichier de layout. Là encore le mécanisme interne de la libraire est remarquable. Voici le nouvel en-tête de la classe

```
class demowin3 : public Withdemowin3Layout<TopWindow> {
public:
   typedef demowin3 CLASSNAME;
   demowin3();

   void OnBab();
   void OnEx();
};
```

La déclaration des composants a purement et simplement disparu! Par contre l'ancêtre de notre classe est profondément modifié. Il s'agit cette fois d'une classe patron dont le nom intègre le nom du layout défini dans le fichier.lay. Si vous consultez les sources vous comprendrez (en simplifiant un peu) que le preprocessing du fichier de layout va générer la déclaration d'une classe template (insistons sur le template) qui aura comme nom le nom du layout préfixé par With et que dans ce template sont déclarés les composants de l'interface utilisateur. Ensuite à la compilation normale, ce template est instancié avec comme paramètre TopWindow dont il dérive. Cette façon de dériver un template d'un de ses arguments génériques, bien que connue depuis longtemps, a été popularisé par Andrei Alexandrescu dans son ouvrage "Modern C++ Design, Generic Programming and Design patterns Applied". Ce mécanisme fait que la classe de base TopWindow se trouve étendue avec une classe surgie de nulle part mais qui y injecte les composants. Du coup le code du constructeur est à notre niveau réduit à trois fois rien.

```
demowin3::demowin3()
{
   CtrlLayout(*this, "Programmez!");
   CenterScreen();
   pr.SetText("Programmez!");
   bab <<= THISBACK(OnBab);
   ex <<= THISBACK(OnEx);

}
```

Cerise sur le gâteau changer le nom du layout dans le nom de notre classe est seul suffisant pour avoir une fenêtre avec une autre interface (c'est-à-dire un autre layout issu du concepteur) automatiquement! Magie des templates C++, et grand coup de chapeau au concepteur du mécanisme. Nous devons arrêter ici cet article, mais sachez que tout sous Ultimate++ est de cette veine. Découvrez Ultimate++, vous ne le regretterez pas.

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

Abonnez-vous

1 ÉCO

Recevez le magazine chaque mois

économisez 20 €

11 Numéros

45 €

Au lieu de 65,45 €
(Prix au numéro)

(Prix France métropolitaine)

-40%

2 Étudiant

Vous devez justifier de votre statut d'étudiant

Economisez 26 €

11 Numéros

39 €

Au lieu de 65,45 €

(Prix au numéro)

(offre réservée France métropolitaine)

-30%

3 Numérique

Lisez chaque mois le magazine

Format PDF
(téléchargement)

11 Numéros

30 €

Tarif Monde entier

Inscription en ligne uniquement

www.programmez.com

2,73€ le numéro

+ 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 ! ou abonnement en ligne : www.programmez.com

☐ ABONNEMENT 1 an ECO au prix de 45 € TTC. *Tarif France métropolitaine.*
Tarifs hors France métropolitaine : CEE et Suisse : 51,83 € - Algérie, Maroc, Tunisie : 55,95 € - Canada : 64,33 € - Tom : 79,61 € - Dom : 62,84 € - Autres : nous consulter

☐ ABONNEMENT 1 an ETUDIANT (11 n°) : 39 € TTC. *Offre limitée à la France métropolitaine. Photocopie de la carte d'étudiant obligatoire*

☐ + SUPPLEMENT ABONNEMENT 1 an INTEGRAL au prix de lancement de 12 € TTC. (s'ajoute à une des formules d'abonnement)

MONTANT TOTAL DE L'ABONNEMENT : ☐ €

☐ 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 - 75010 Paris.
abonnements.programmez@groupe-gli.com

Programmez!
LE MAGAZINE DU DÉVELOPPEMENT

Offre limitée, valable jusqu'au 31 janvier 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 104

ACTUS

Une clé au parfum

Complètement indispensable : la clé USB parfumée ! Avec au choix fraise, citron, orange, ou menthol (et la jolie couleur fluo correspondante) pour embaumer vos données en 1, 2, 4 ou 8 Go. Notez qu'avec plusieurs exemplaires, et un classement rigoureux, vous pouvez ainsi repérer la clé et les

fichiers dont vous avez besoin même dans l'obscurité la plus totale. Vous pouvez aussi utiliser la clef comme désodorisant en la passant sous vos aisselles (pas du côté connecteur de préférence). Chez Memup, de 12 à 75 € la clef environ, selon la capacité.

Footballeur de salon

Un tapis sensitif avec un ballon fixé dessus : voici le dernier cri en matière de contrôleur virtuel pour les simulations de foot sur PC et consoles Playstation, de Pro Evolution Soccer à FIFA (toutes versions) en passant par bien d'autres. Le tapis Shootpad se branche sur le port jeu habituel. Il gère les passes, dribbles et tirs, tandis qu'une manette sans fil tenue à la main, à détection distante, comparable à celle de la Wii, gère les déplacements et sélections de joueurs. Un logiciel permet d'affiner les réglages de contrôle et même de définir certains " combos ". Evidemment, c'est épuisant et, vu par un témoin extérieur, vous offrirez un spectacle déplorable… Mais ça marche très, très bien ! Importé par Big Ben Interactive, de 52 à 60 € environ.

Wii Sports : contrôle total

Le jeu Wii Sports fourni avec la console a beaucoup fait pour son succès, avant tout les simulations de golf et de tennis, les plus

réussis. On attendait donc avec impatience des contrôleurs virtuels grandeur nature, ils arrivent : la raquette pour le tennis, le club pour le golf. Le procédé est simple, par intégration dans l'accessoire de la manette Wii d'origine, mais fonctionne parfaitement. Le tout est réuni dans un pack commun et éminemment sportif, chez Big Ben bien sûr. 2008 verra aussi l'arrivée de la batte de baseball et de la boule de bowling virtuelles, dédiées aux autres Wii Sports, mais uniquement disponibles aux USA pour l'instant. Enfin, pour finir, vous pourrez protéger vos manettes avec ces caches fluo de diverses couleurs (ici le pire, le rose). Gadget ? Pas tant que ça, car le contact est excellent et la prise en main bien plus ferme : pour joueurs sérieux… ou brutaux !

Le vrai chic du geek

Vous ne serez plus considéré comme un vrai geek en 2008 si vous n'arborez pas, pour tenir votre pantalon flasque, troué et taché de café, cette magnifique ceinture faite tout simplement d'une belle nappe IDE ! Le dernier cri néo-high-tech-décadent, pour 17 $ à peine chez fractalspin.com. A s'offrir en combinaison avec le porte-cartes circuit imprimé, tout aussi " cheek ".

DS : les petits plats dans les petits

Vous devrez patienter jusqu'en février pour vous adonner à la Cuisine en Délire, nouvelle adaptation de la license Ratatouille sur Nintendo DS. L'ensemble combinera rapidité et réflexion, votre mission étant de confectionner à la demande les plats les plus fous pour faire passer votre rat favori au rang de Grand Chef. Si le jeu est plutôt destiné aux " plus jeunes ", la vogue des jeux d'entraînement cérébral pour les " plus vieux " se poursuivra l'année prochaine : pas moins de quatre titres annoncés pour agiter les neurones des adultes, et même des seniors ! Bientôt remboursable pas la sécu ?

Un MMO heureux

Encore en bêta-test fin décembre, ElfOnline devrait être la bonne surprise des MMO online de 2008. Le jeu joue à fond la carte de l'originalité avec des graphismes manga très colorés, une structure de progression des personnages rapide, des règles de comportement vraiment étonnantes, et un humour inhabituel. On peut ainsi créer son équipement et ses " armes " à partir des objets les plus extravagants, depuis la spatule en bois

offensive jusqu'à la brouette maléfique de téléportation. Les premiers tests ont déclenché des réactions enthousiastes comme on en a rarement vu. Mais il faudra sans doute patienter encore un peu pour avoir la version française. En attendant, suivez l'évolution du développement final sur www.happymmo.com, ou dans les forums, ça vaut le coup.

PAR JEAN-MICHEL MAMAN

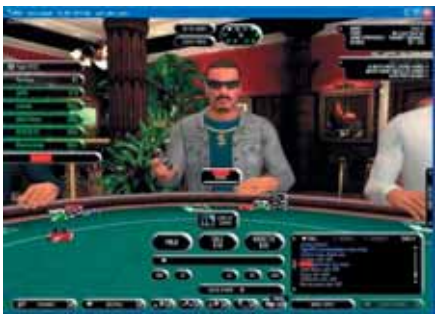
Poker online : carré d'as en 2008

2007 a marqué le triomphe mondial du poker, variante Texas Hold'em, et avant tout, version online. En 2008, les sites qui hébergent les parties vont devoir aller encore plus loin : vers le réalisme total, une sorte de " poker life " sur le web. Addiction totale ?

Avec plus de 30 millions de passionnés dans le monde, c'est la toute dernière folie Internet… et un incroyable pactole ! Car les sites de poker, basés pour la plupart en Amérique du Sud ou dans des paradis fiscaux, raflent une mise énorme : pas besoin de tricher, il leur suffit de prélever un droit minimal sur chaque partie (souvent à peine 1 $) pour gagner à tous les coups. Du côté du joueur, ce n'est pas cher (près d'une heure de jeu en étant stratégiquement prudent) et ça peut rapporter gros : voyages, entrées dans des tournois réels prestigieux, etc. Bref, bénéfices des sites comme addiction des joueurs ont grimpé en flèche. Mais comment maintenir le phénomène ou atteindre de nouveaux sommets en 2008 ?

Une bonne main

La sécurité n'est pas en jeu, et les joueurs, malgré une ou deux affaires suspectes, ont pleinement confiance. Avant tout parce que les sites n'ont pas besoin de truquer les cartes pour gagner : l'accumulation des parties suffit très largement à leur bonheur. Ensuite, parce que des procédures très strictes sont utilisées pour garantir l'intégrité du jeu. Ainsi, les cartes des autres joueurs ne sont transmises depuis le serveur vers chaque client qu'à la fin d'un coup. Impossible donc de les hacker au passage pour connaître à temps le jeu de ses adversaires. Ensuite, parce que la procédure de tirage aléatoire est incroyablement rigoureuse. Chez Pokerstars par exemple, leader actuel du marché, l'algorithme de répartition des cartes s'appuie sur deux données variables comme imprévisibles, la température interne de plusieurs composants du serveur, plus les mouvements de souris combinés de tous les joueurs. Il est en outre crypté dans toutes ses phases. On n'est donc pas loin du fameux " parfait hasard " ! Enfin, les problèmes de serveurs sont bien maîtrisés. Certes, certains sites hébergent souvent plus de 200 000 joueurs et de 12 000 tables simultanément, mais les données échangées restent légères.



PKR

Pok3D

Améliorer son jeu

En 2008, les développeurs de poker online vont donc s'attaquer surtout au traitement graphique des parties. Pour augmenter l'addiction, l'innovation va se porter sur un environnement plus réaliste, avec la 3D en tête de liste. Quelques sites proposent déjà ce type de graphisme, et sont en train de l'améliorer. Dans ce domaine, les français avaient été les initiateurs : dès fin 2006, les développeurs de Mekensleep lançaient déjà Pok3D, conçu à partir de logiciels open source et disponible sur Debian, Red-Hat et autres. Malgré des décors plaisants, et des avatars assez sommaires mais sympathiques, le succès est cependant modeste aujourd'hui : face aux grosses multinationales sud-américaines, difficile de tenir, même en allant se baser à Chypre…

PKR, en revanche, est un sérieux outsider. Son traitement 3D, superbe pour les tables comme pour les joueurs, a toutes les chances de rafler le pot. Le prix à payer (non, pas pour les pertes au jeu !) est une installation un peu lourde, sujette à quelques problèmes avec DirectX, et encore impossible sous Vista. Mais c'est magnifique, et l'équipe de développeurs annonce des miracles pour l'année prochaine. A suivre de près. Le très respectable site LadbrokesPoker mise lui aussi son année 2008 sur la 3D. Tables en perspective, plusieurs angles de caméra, et surtout des avatars de joueurs qu'il est possible de customiser au cheveu près. En outre, des animations d'expression et d'attitude viennent s'intégrer au jeu, pré-programmées et utilisables directement comme des emoticons de luxe.

L'immersion totale ?

Mais le poker online devrait pousser bientôt au-delà de la 3D. Le site de poker Dusk Till Dawn vient juste d'ouvrir la voie en créant, conjointement à ses jeux online, un club de poker bien réel, de 46 tables tout à fait concrètes, à Nottingham en Angleterre. Les deux mondes, réels et virtuels, vont interagir en permanence par des écrans et des tournois de qualification. Mais il faut attendre l'innovation la plus significative de la part d'une start-up californienne, ActSynCo. Ses développeurs préparent des tables de poker online en images réelles par streaming proche de celui des vidéo-conférences, les joueurs utilisant webcam et micro pour se faire voir (impassibles, bien sûr). En même temps, au milieu des vidéos des joueurs, la gestion de la partie et la représentation de la table, des jetons, et des cartes, restent pleinement informatiques et graphiques. Ce projet, au carrefour du réel et du virtuel, devrait être lancé en mai prochain.

Ladbrokes

En attendant la loi

Enfin, développeurs français avides de surfer sur la vague du Hold'em, sachez que les deux grandes chaînes de casinos français, Partouche et Barrière, n'attendent qu'une évolution de la législation des jeux de notre beau pays pour lancer chacun leur site online de poker. Le bruit court que cette libéralisation pourrait se faire dès juin 2008, et que Barrière a même déjà engagé des développeurs indies… Faites vos jeux !

Collection webmaster

- ***Difficulté : **/******
- ***Editeur : Micro Application***
- ***Auteur : divers***
- ***Prix : 25 €***

Gadget et widgets

Ces petites applications font désormais partie du quotidien du développeur et des utilisateurs, que ce soit sur son bureau système ou sur le web. L'ouvrage débute par un rappel technique de ce qu'est un widget et des différents types de gadgets (Windows Live, Google, etc.). L'auteur propose de travailler sous différents IDE Web pour faciliter le développement. Puis on entame le développement de gadgets Google, Netvibes, Yahoo, Vista.

Lacune de taille : l'absence de MacOS X et son Dashboard ! Un livre original et sympathique pour égayer ses développements.

Flex & Air

Les technologies Adobe pour le RIA et le RDA intéressent de plus en plus le développeur. Ce livre est une plongée dans les univers Flex et Air. L'ensemble des accès sont abordés : ActionScript 3, MXML, les données, la création d'interface, Flex Charting, comment passer de Flex à AIR. À la fin de l'ouvrage, on trouve des informations sur les tests unitaires ou encore les Flex Data Services.

Une bonne introduction pour ceux qui souhaitent démarrer dans la RIA !

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…

PHP 5.2

- ***Difficulté : ******
- ***Editeur : Eni édition***
- ***Auteur : Olivier Heurtel***
- ***Prix : 27,14 €***

Si la version 5.3 vient de sortir, il n'est pas inutile de revenir sur les modifications, assez profondes, de la v5.2 de PHP. Le but est de concevoir un site web dynamique avec PHP. La v5.2 n'est donc pas spécifiquement abordée par l'auteur. L'ensemble du langage est présenté et décrit : variables, formulaires, accès aux données, gestion des sessions, etc. De très nombreux exemples ponctuent l'ouvrage. Dans la partie données, l'auteur aborde le modèle PDO de PHP et le support des principales bases de données du marché. Dommage qu'un chapitre sécurité n'ait pas été inclus.

Le système d'information durable : la refonte progressive du SI avec SOA

- ***Difficulté : ******
- ***Editeur : Lavoisier***
- ***Auteur : collectif***
- ***Prix : 60 €***

Avant que les informaticiens à l'origine des systèmes existants partent à la retraite, il est nécessaire d'engager des projets de refonte progressive. Cet ouvrage étudie l'architecture orientée services ou SOA (donnant les bases d'un système informatique durable, plus agile, capable de s'adapter aux évolutions et aux processus de gestion avec les tiers. Cette architecture prend le meilleur des approches objets, d'urbanisation, de BPM, etc. Elle s'appuie sur les systèmes de gestion de règles (Business Rules Management System), de gestion des données de référence et de paramètres (Master Data Management), et l'ingénierie des modèles avec MDA (Model Driven Architecture). Les niveaux de mise en œuvre de la SOA de surface, de refonte et étendue sont développés. Des exemples permettent de mieux comprendre les enjeux de la SOA et sa mise en place.

Programmation concepts, techniques et modèles

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

Finalement, c'est quoi la programmation ? Pour revenir aux fondamentaux mêmes du code et de ses philosophies, 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 !

Programmer en Java

- ***Difficulté : ******
- ***Editeur : Eyrolles***
- ***Auteur : Claude Delannoy***
- ***Prix : 35 €***

Autre classique des livres de programmation, Programmer en Java ! Cette 5e édition couvre Java 5 et 6. Claude Delannoy applique au langage Java la démarche pédagogique qui a fait le succès de ses livres sur le C et le C++. Il insiste tout particulièrement sur la bonne compréhension des concepts objet et sur l'acquisition de méthodes de programmation rigoureuses. L'apprentissage du langage se fait en quatre étapes : apprentissage de la syntaxe de base, maîtrise de la programmation objet en Java, initiation à la programmation graphique et événementielle avec la bibliothèque Swing, introduction au développement Web avec les servlets Java et les JSP. Complet et progressif, de nombreux codes ponctuent le propos de l'auteur. Bon point : un CD-Rom incluant Eclipse Europa et la JDK 6 et l'ensemble des sources du livre !


SERVEUR ECO D'AMEN
UNE TECHNOLOGIE
EN HARMONIE
AVEC L'ECOSYSTÈME.
D'ÉMISSION
-55%
DE CO2*
amen
IN WEB WE TRUST
A partir de
49 € HT/mois **
soit 58,60 € TTC/mois
NE CHOISISSEZ PLUS ENTRE
PERFORMANCE ET ÉCOLOGIE.
SERVEURS ECO D'AMEN :
DE 1 À 3 Go DE RAM,
55 % D'ÉMISSION DE CO2
EN MOINS PAR AN.
http://eco.amen.fr
AMD Athlon X2 3400+ ou BE-2350
De 2x1,8GHz à 2x2,1GHz
Interface Plesk 8.2 – 10 domaines
OS : Fedora Core 7, Ubuntu 6, Debian 4, Windows Server 2003
De 1 à 3 Go de RAM
Disque dur de 80 Go à 160 Go
Trafic mensuel : 1 To
1 à 2 adresses IP
Inclus : Reboot, Restore et Recovery
En option : Amen DataBackup 10 Go (Inclus sur ECO 3000)
AMD 64 Athlon X2
PARTENAIRE INDUSTRIEL
entrepreneurs, faites le choix de l'économie numérique
www.econumerique.pme.gouv.fr
NUMÉRO GRATUIT : 0800 740 935 ou www.amen.fr
NOMS DE DOMAINE - EMAIL - HÉBERGEMENT - CRÉATION DE SITE - E-COMMERCE - RÉFÉRENCEMENT
* comparé à un serveur équivalent de la gamme concurrente - ** pour un engagement annuel - AMEN RCS PARIS - B 421 527 797 - IN WEB WE TRUST - Nous croyons au web. Conditions Générales de Vente sur le site www.amen.fr - REFLEXION FAITE