

SPECIAL FUTUR

Document exclusif :
**LE DÉVELOPPEMENT
SE TRANSFORME**

2005-2007

Les **PROJETS** de **MICROSOFT** dévoilés

MONO 1.0 : Compatible .net en Open Source

JEUX VIDÉO : Quels outils pour le développeur indépendant ?

SÉCURITÉ

**SECURISER
LE CODE** (suite)

**PRÉVENIR LES FAILLES
OBFUSCATION,
CRYPTAGE...**

LANGAGE

C# 2.0 : quels
changements ?

ObjectSpaces :
la persistance .net

PRATIQUE

**MIGRER DE
DELPHI 7 À 8**

**DES VITAMINES C++
POUR JAVA**



Le BEA eWorld à San Francisco

Le SOA a été le maître mot de la dernière conférence eWorld de BEA. Mais l'éditeur voit déjà plus loin en proposant son nouveau concept : le "Liquid Computing"

La neuvième édition de la conférence eWorld de BEA s'est tenue du 24 au 27 mai à San Francisco. L'éditeur qui fête ses 10 années d'existence, a voulu montrer son dynamisme et son avance technologique à un moment où le doute s'installe (à tort ou à raison) sur l'avenir de l'entreprise. Il fallait donc être rassurant et c'est ce que Alfred Chuang, le CEO, s'est employé à faire durant toute la durée de la conférence. Il y a eu une cascade d'annonces, sur les nouveaux produits tels que Quicksilver et Alchemy, dont nous parlerons plus loin, le lancement d'un nouveau concept baptisé le Liquid Computing, mais il a surtout été question de SOA, le cheval de bataille de BEA.

Déployez SOA maintenant ! Voilà ce qu'a martelé Alfred Chuang, tout au long de son intervention. Pour ceux qui auraient encore pu y échapper, SOA signifie Service Oriented Architecture, ou Architecture Orientée Service en bon français. Sa définition précise diffère quelque peu selon les acteurs, mais tous sont d'accord pour reconnaître qu'il s'agit d'un concept dans lequel une application est considérée comme un fournisseur de services.

SOA est plus simple que les systèmes distribués

Un service est un module généralement intégré dans un composant, chargé d'exécuter une fonction bien définie. L'architecture SOA permet de réduire les couplages entre les modules pour

une meilleure réutilisabilité, le couplage entre l'infrastructure et les plates-formes pour une meilleure interopérabilité et enfin réduire le couplage entre les clients et les services pour une meilleure évolutivité. Cela peut rappeler les systèmes distribués tels que Corba ou DCOM, mais pour Cornelius Willis, Vice-Président en charge de la partie développement de BEA, la grande différence réside dans le fait que dans SOA les transactions ne sont pas forcément synchrones. *"SOA est techniquement bien plus simple, cette architecture offre un niveau de granularité moins fin que les systèmes distribués, ce qui fait que pour les développeurs, il n'y a pas grand'chose qui change. C'est au niveau des architectes que tout le travail doit être effectué"* explique t-il.

Le Liquid Computing, la nouvelle stratégie de BEA

Il y a-t-il une vie après le SOA ? Oui, répond BEA, c'est le Liquid Computing ! L'éditeur ne considère pas le SOA comme une fin en soi, mais plutôt comme une étape, permettant d'atteindre le but suprême, qui est le Liquid Computing. C'est sous ce nom (anciennement connu sous le nom de code de Sierra) que BEA a créé un nouveau concept, dans lequel la réactivité du système informatique ne se compte plus en années, ou en mois, mais en minutes ! De même que le liquide adapte instantanément sa forme en fonction du contenant, le Liquid Computing permet au sys-



Pour Alfred Chuang, CEO de BEA, il faut déployer SOA maintenant !

tème d'information de s'adapter instantanément aux besoins de l'entreprise par le biais d'interactions en temps réel. Il se définit en trois points. Le premier est la compatibilité. Tout le système, qu'il s'agisse des applications, bases de données, matériels, plates-formes, processus métier et Web Services, doivent grâce à cette nouvelle structure, être capable d'échanger des données, à l'intérieur, ou même à l'extérieur de l'entreprise. Peu importe si les composants n'ont pas été créés pour travailler ensemble, cette structure fournira tous les standards et l'interopérabilité nécessaires. De multiples technologies et langages pourront ainsi coexister, et les systèmes qui, au départ n'étaient pas destinés à travailler ensemble, pourront le faire.

Le deuxième point est la faculté d'adaptation. Le système d'infor-

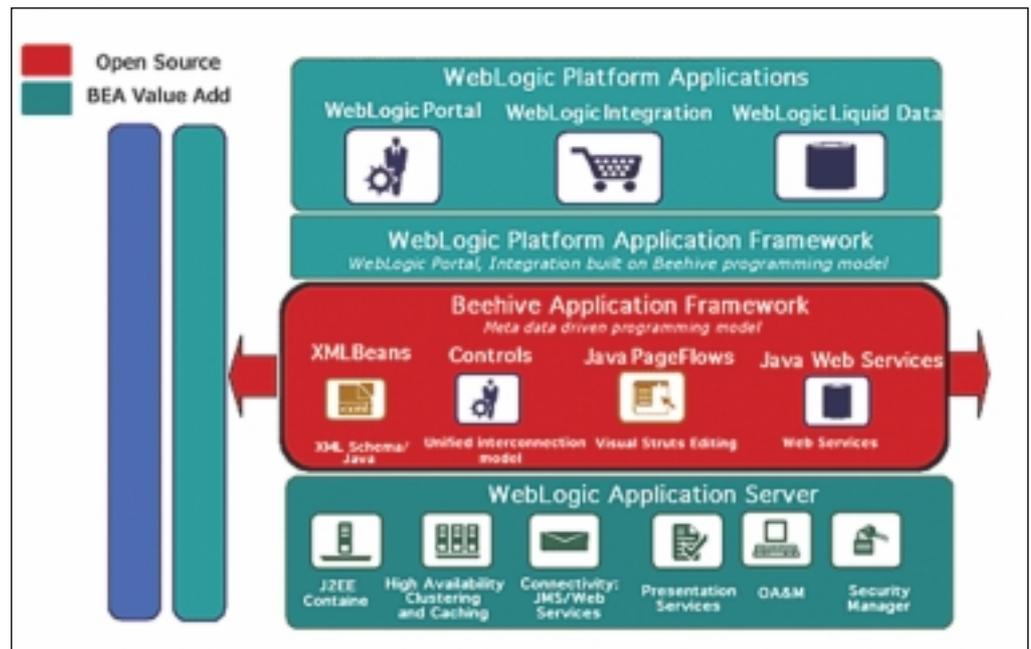
mation doit pouvoir répondre à toutes les demandes et même être en mesure de les anticiper. Il devrait pouvoir le faire de manière dynamique, sans intervention humaine.

Dans le Liquid Computing, les besoins et les réponses sont directement intégrés à l'intérieur du système. Des règles prédéfinies lui indiquent comment réagir, et ces règles sont déclenchées en fonction de certains événements. Cela permet d'une part d'accélérer le traitement des informations et d'autre part de réaliser des économies importantes. En effet, la maintenance et la gestion des applications représentent un coût souvent bien supérieur à celui de leurs acquisitions, ou de leur développement. Ainsi, en s'adaptant de manière automatique, les coûts de maintenance des applications devraient se réduire de

manière importante. Le troisième point enfin est l'augmentation de la productivité. D'après BEA, la durée de vie d'une application est de 10 ans en moyenne. Cela signifie que celles qui sont utilisées actuellement sont basées sur des conceptions et des prédictions qui ont cet âge. Avec une informatique qui réagit plus vite, les applications devraient être parfaitement en phase avec les demandes et ne jamais devenir rapidement obsolètes.

Des objectifs pas faciles à atteindre

Sur le papier, tout cela est évidemment très intéressant, mais reste un peu théorique. Des points importants sont passés sous silence. Le premier concerne l'administration de ces services. Il n'existe actuellement pas de grands moyens de les gérer. Plus grave encore est le problème de la sécurité. Comme il n'est pas question d'attribuer les autorisations d'accès à l'intérieur de chaque composant, il faudra que les développeurs y intègrent des appels à des systèmes externes. On ne sait pas comment seront construits ces systèmes. Les annuaires avec leurs arborescences ne sont pas utilisables pour cette tâche. Comme on le voit, le chemin à parcourir est encore long. Pour arriver au Liquid Computing, BEA préconise son serveur d'applications WebLogic Server Process Edition, ainsi que d'autres produits qui sont en cours de développement. Le premier se nomme Quicksilver. C'est un outil d'intégration ESB (Enterprise Service Bus) et de gestion des services Web. Il ne sera probablement pas disponible avant 2005. Le second, baptisé Alchemy, est destiné à la création de plates-formes pour les clients mobiles. Il sera capable de prendre en compte de multiples appareils avec tous les modes de



La structure d'applications Beehive, s'intègre évidemment dans l'environnement BEA WebLogic, mais peut fonctionner avec les produits d'autres éditeurs

connections possibles on line, off line...). BEA est consciente qu'il y a un virage stratégique à prendre cette année, pour arriver au chiffre prévisionnel de 3 milliards de dollars. Rappelons que le milliard n'a été atteint que sur l'exercice 2003. Le premier trimestre 2004 n'a pas été très encourageant, puisque malgré une petite hausse du chiffre d'affaires, le nombre de licences vendues a légèrement diminué. Beaucoup d'analystes pensent que la solution pourrait venir dans l'achat d'une, ou plusieurs autres sociétés. Pour que cela soit efficace, il lui faudrait acquérir des entreprises d'une taille presque équivalente à la sienne, c'est-à-dire avec des chiffres d'affaires de l'ordre du demi milliard, ce qui n'est pas facile, ni à faire, ni à trouver. Pourtant, Alfred Chuang ne semble pas de cet avis. Il a indiqué qu'il n'était pas question de se "verticaliser". Cela dit, il n'y a pas de raison de se montrer pessimiste. Ce serait oublier les gros atouts que possède l'entreprise. Il y a d'abord une grande capacité à se reconverter, une

technologie reconnue par tous et enfin la qualité des équipes et les compétences des collaborateurs. Nous aurons probablement la

réponse lors du prochain eWorld de 2005, qui se tiendra cette fois encore à San Francisco.

■ Alain COUPEL

Beehive, un projet Open Source pour J2EE

Annoncé en mai, le projet Beehive Open Source est un outil destiné à faciliter le développement d'applications J2EE et d'applications orientées services. Le BEA Workshop est composé de deux parties, un outil de développement et l'Application Framework, et Beehive est basé sur ce dernier. Il en reprend de nombreuses fonctions telles que les annotations, les contrôles Java, les Java Web Services et les Page Flow. " Tous les contrôles de WebLogic seront utilisables dans Beehive. De plus, les développements pourront être réalisés sous d'autres environnements que BEA Workshop, comme les outils de Borland par exemple " explique Cornelius Willis. Beehive pourra également être utilisé avec d'autres serveurs d'applications. Les deux plates-formes Beehive et WebLogic évolueront simultanément. Pour BEA, le but du projet est d'encourager les développeurs à travailler sur J2EE en leur donnant des outils aussi simples à utiliser que la plate-forme .NET. Le modèle Open Source ayant été choisi pour cela, mais surtout parce que les nouveaux standards sont pris en compte beaucoup plus rapidement qu'avec les JSP. Beehive devrait être disponible dans le courant de l'été, gratuitement, sous la licence Apache 2.0. Le choix d'Apache a été fait pour sa notoriété et pour s'assurer que cette technologie sera utilisée dans d'autres projets Apache supportés par BEA, tels que Tomcat ou les XML Beans.

InterSystems s'étend du côté de l'intégration

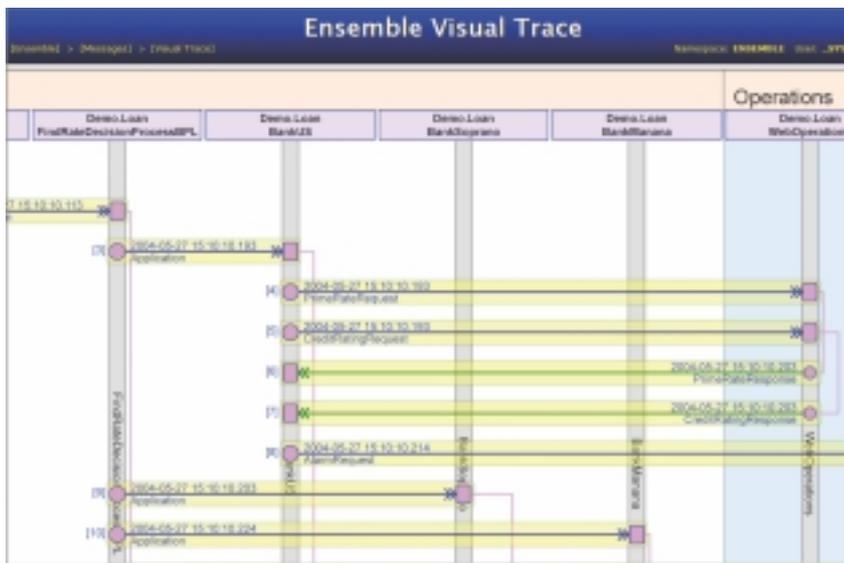
Connu pour sa base post-relationnelle Caché, InterSystems ajoute désormais l'EAI à son catalogue, avec son outil d'intégration Ensemble.

Le symposium, organisé par la filiale française d'InterSystems s'est déroulé les 3 et 4 juin à Cannes en présence de ses clients et partenaires. La différence entre eux étant assez floue, puisque l'éditeur entretient avec ses clients des rapports aussi étroits qu'avec ses partenaires. InterSystems est une société privée, créée en 1978 et qui emploie environ 500 personnes pour un chiffre d'affaires annuel de 145 millions de dollars. La maison mère se trouve dans le Massachusetts et le bureau français est basé à Sophia Antipolis. Pendant longtemps, sa gamme a été limitée à un seul produit, la base de données post-relationnelle Caché. C'est une base qui dispose d'une véritable technologie objet ainsi que d'un environnement de développement. Bien qu'elle ne soit pas très connue, elle est classée n° 5 par le Gartner Group, derrière Microsoft, IBM, Oracle et Sybase. Cette base connaît un important succès, particulièrement dans le domaine médical, d'où elle tire 80 % de ses revenus.

Un outil orienté messages

Baser toute son activité sur un seul produit est toujours risqué, aussi InterSystems a décidé de se tourner vers l'EAI en proposant son outil Ensemble. " Les clients demandaient plus qu'une simple base et il y avait surtout un important marché à saisir " déclara

le chef produit Joe Gallant. Ensemble représente également une bonne opportunité pour de nombreux éditeurs de logiciels spécialisés (finance, santé...) en leur permettant de s'étendre. Actuellement, il n'existe pratiquement plus de développement " pur " à partir de rien. Tous les projets doivent tenir compte de l'existant et la moitié des ressources passe dans l'intégration. Tout nouveau projet d'intégration nécessite du développement et tout développement nécessite un projet d'intégration. "Etre nouveau sur le marché de l'EAI est un avantage, puisqu'il permet de prendre des leçons de tout ce qui a été fait auparavant" poursuit-il. Il est toujours intéressant de connaître les origines des différents éditeurs. Beaucoup viennent du Workflow ou du Middleware, et proposent des produits plus centrés sur les flux de données. Au contraire, parce qu'il est architecturé autour d'une base de données objet, Ensemble est centré sur le stockage et la gestion des messages. Il se distingue en cela des architectures orientées composants comme le sont celles de WebLogic ou de .NET. Ensemble est le résultat de 3 ans de travail, mais le produit n'a été dévoilé qu'en novembre



2003. "Nous ne parlons d'un produit que lorsque nous avons au moins 30 sites en production. Actuellement, nous rencontrons de beaux succès avec des projets qui étaient en panne" nous confie Dominique Jourda, directeur des ventes en France.

Une formation rapide

Ensemble n'est pas une compilation d'outils disparates, mais une plate-forme complète d'intégration d'application conçue pour les architectures SOA (Service Oriented Architecture). Elle est constituée de nombreux adaptateurs (données, applications, émulation, systèmes transactionnels, protocoles...), d'un serveur d'intégration, d'un serveur d'applications, d'une base de données objet, ainsi que d'un outil de gestion et de développement. Ensemble est simple à programmer. Ainsi, la formation ne dure que 5 jours. Cette simplicité, permet de mettre l'outil aussi bien à la disposition des techniciens que des fonctionnels, ce qui n'est pas le cas avec certaines plates-formes Java qui nécessitent de

grandes compétences dans ce langage. Comme sur la quasi-totalité des outils récents, Ensemble permet de développer aussi bien de manière graphique, que par lignes de code, toute modification effectuée d'un côté étant répercutée de l'autre. Le langage utilisé peut-être du Basic (un langage très proche de VBScript), ou ObjectScript (un langage en Open Source proche de Java). Ensemble est compatible avec J2EE, .NET, BPEL 4WS (Business Process Engineering Language), les Web Services, JMS Par rapport à la base post-relationnelle Caché, Ensemble est bien complémentaire. Caché suffit pour les développements seuls, et Ensemble sera utilisé pour les développements et pour l'intégration.

La plate-forme Ensemble est entrée dans la partie visionnaire du Magic Quadrant de Gartner, six mois seulement après sa sortie. Toutefois, malgré sa reconnaissance par les plus grands analystes, InterSystems manque encore de notoriété par rapport à ses principaux concurrents.

■ Alain COUPEL

Le Centre Hospitalier de Besançon mise sur XML pour les échanges d'applications médicales.

Pour permettre aux professionnels de la santé et aux patients d'accéder rapidement aux dossiers médicaux, le CHU de Besançon a fait appel à la plate-forme d'intégration de Software AG.

À la clé, le bus d'intégration XML Entire X, qui assure une vision unifiée des interactions entre les différentes applications.

La loi du 4 mars 2002 relative au droit des malades et à la qualité du système de santé, baptisée aussi Loi Kouchner, a totalement changé la donne : désormais, les patients peuvent accéder librement à l'ensemble des informations les concernant. Conséquence : les établissements de santé doivent adapter leur système d'information. Le CHU de Besançon (300 000 patients, 400 médecins et 4 500 agents) n'échappe pas à la règle. Pour Serge Adam, DSI du CHU de Besançon, " Il s'agit de faire évoluer le système d'information tout en conservant l'existant. Il s'agit aussi d'établir une communication avec les réseaux de soins de l'Agence Régionale d'hospitalisation (ARH) de Franche Comté. L'objectif est simple : alimenter un gisement de données pour concentrer et organiser l'ensemble des informations sur les patients, à partir de documents provenant de sources diverses ". Le chantier est de taille, d'autant qu'il faut s'appuyer sur une architecture basée sur des normes et des standards informatiques utilisés dans la santé (IHE, HL7, etc.). Par rapport à tout cela, le Centre Hospitalier de Besançon est confronté à des problèmes d'intégration d'applications. "Qu'il s'agisse de la carte vitale, des feuilles de soins, etc.,

les données sont de plus en plus dématérialisées. Désormais, nous devons revoir toute notre architecture pour placer le patient au centre de l'information, afin qu'il puisse partager ses dossiers avec les professionnels médicaux ", déclare Serge Adam. Octobre 2002, le CHU lance un appel d'offres. Le cahier des charges est si vaste qu'il est découpé en cinq lots. Le premier concerne les dossiers cliniques, le second comprend la fourniture d'outil EAI et d'un système de GED (plus orienté gisement de données) avec la maîtrise d'œuvre. Le troisième lot a pour mission la mise en place d'un bureau virtuel (portail d'accès et serveur d'application). Le quatrième concerne l'intégration de l'imagerie médicale et enfin le dernier lot, se rapporte à la fourniture et l'installation de composants et de matériels. Software AG est retenu sur le second lot. "Cet éditeur a été choisi pour son expertise technologique en XML. Ce langage représente l'espéranto de l'information. C'est un langage évolutif, extensible et interopérable au niveau de l'intégration des applications. Trois critères incontournables pour notre projet. Nous lui avons aussi confié la maîtrise d'œuvre et l'intégration de l'ensemble de la solution" résume Serge Adam.



L'intégration, clé de voûte du système hospitalier

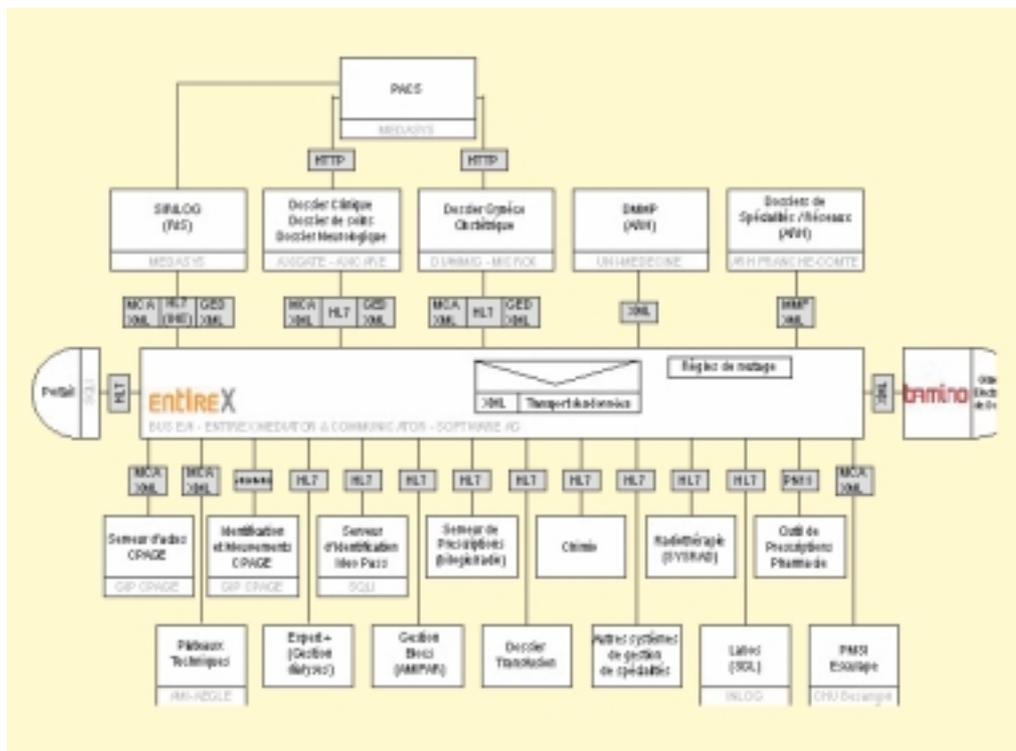
La plate-forme est déclinée selon deux grands axes. Le premier concerne la plate-forme intrahospitalière, pour l'échange d'informations entre les applications hétérogènes du Système d'Information Hospitalier (SIH) et l'alimentation du dossier légal du patient. Le deuxième axe porte sur la plate-forme régionale, pour l'échange d'informations et le partage du dossier patient entre les établissements hospitaliers, les médecins libéraux, les laboratoires et le patient. " Le premier projet explique Joël Damas, responsable solution chez software AG, consiste à connecter les dossiers médicaux spécialisés (gynécologie, obstétrique, urologie, etc.) avec les applications existantes. " Pour intégrer les dossiers et échanger avec les applications du CHU (des applications métiers), la direction informatique est confrontée à deux possibilités : soit faire des liaisons point à point, mais ce type de schéma est très vite ingérable, ou bien mettre en place un bus applicatif de données, pour intégrer les applications développées sous Windows et sous Unix. C'est cette dernière solution qui est

retenue : le bus applicatif natif XML. "Ce format permet de véhiculer des données, comptes rendus, radios, etc. et de les restituer quels que soient les formats d'origine" souligne Serge Adam.

Une plate-forme native XML

Le Bus XML (EntireX Mediator) est chargé d'orchestrer les échanges d'informations entre les systèmes intégrés. Comme les applications propriétaires du CHU ne produisent pas du XML, il faut intégrer un composant, un connecteur applicatif entre le bus de données et les applications. De leur côté, les connecteurs santé sont des composants ouverts qui peuvent prendre en compte les flux normés et les flux propriétaires. "Cette approche permet de vérifier à chaque instant le document (le valider, identifier l'émetteur, connaître ses propriétés, etc.)" précise Joël Damas. Au niveau du bus de données EntireX XML Mediator, tout est automatique. Un exemple ? L'application émettrice envoie un document, ce document est ensuite stocké puis envoyé vers le bus XML, ce dernier le prend en charge, agrège l'information et la désagrège, et route, en fonction du contenu et de la structure du document. Tous

les messages et documents XML sont stockés dans la base de données native XML Tamino de Software AG. "L'intérêt d'une base de données native XML est simple. Tamino peut stocker les documents XML dans leur format original, ce qui supprime le processus de liaison des données et des tables et améliore les performances. Le moteur est combiné à une base organisée de manière hiérarchique, module X-Node pour le mapping des tables en XML. Nous implémentons les requêtes XQL, XPath, et Soap. A titre indicatif, un SGBDR dispose juste d'un index de tri, alors que Tamino intègre trois types d'index : texte, tri, et un index de structure " explique Joël Damas. Mais Tamino joue aussi le rôle de GED. La base de données intègre en effet, tous les dossiers légaux des patients, les comptes-rendus, etc. Tamino est alimenté par le bus EAI et prend en charge tous les messages des applications. "L'avantage de cette plate-forme est de générer en XML l'ensemble des flux, depuis les dossiers cliniques et les différents comptes-rendus médico-techniques (radiologie, biologie, etc.) vers le dossier fédérant les documents du



patient" indique Joël Damas. Lorsque le dossier du patient est transformé au format XML, il sera accessible sous plusieurs formes. Il pourra être délivré sous forme d'un CD-Rom avec une clé de verrouillage, ou bien accessible, via un navigateur, ou encore être hébergé chez un tiers de confiance avec la recommandation IHE profil XSD (format informatique

utilisé dans la santé) pour être interopérable avec les hébergeurs agréés. Fin 2005, le projet devrait être finalisé. "Nous avons fourni EntireX XML Médiateur avec des connecteurs spécifiques tels que : Cpage, et HL7. Aujourd'hui, la direction informatique du CHU réalise une étude d'adaptation pour s'assurer que les applications dialoguent bien entre elles.

Parallèlement à cela, nous attaquons les spécifications des prochains connecteurs, à savoir : Fast pharmacie et DMPP (dossier médical minimum partagé)" déclare Joël Damas.

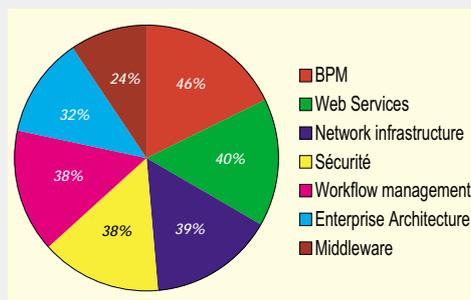
Une plate-forme évolutive et pérenne

Selon Serge Adam, les bénéfices d'une telle plate-forme sont de plusieurs ordres. "En premier lieu, la communication entre professionnels de la santé et patients est essentiellement améliorée, grâce à un accès immédiat à l'ensemble des informations produites par les différentes applications. En outre, la mise en place d'une architecture d'intégration XML permet d'étendre le capital fonctionnel du système d'information à l'extérieur. De plus, cette architecture permet de relier entre elles les anciennes et les nouvelles applications. Dernier avantage, le bus d'intégration XML Entire X assure une vision complète et unifiée des interactions entre les différentes applications."

■ Annie Lichtner

Les enjeux de l'Intégration

Selon le cabinet de conseils Meta Group, l'intégration est au cœur de toutes les initiatives informatiques : avec en premier chef, le BPM 46 %, les Web Services et SOA (Architecture Orientée Service) 40 %, le Network Infrastructure 39 %, la sécurité 38 %, le workflow 38 %, l'Enterprise Architecture 32 % et le Middleware 24 %. Un marché qui se porte bien, avec une croissance de l'ordre de 5 à 7 %. Pour Christophe Toulemonde, directeur des programmes Integration & Development Strategies au Meta Group, "Le marché de l'intégration se décline en trois grandes catégories : la standardisation et les Web Services associés, la Commodisation avec les outils d'infrastructure tels que les bases de données et la



Convergence, avec trois leaders : IBM, Microsoft et BEA." Toujours d'après le Meta Group, les entreprises recommencent à mettre en place des chantiers d'importance, avec une vision plus stratégique. "Les directions informatiques souhaitent réutiliser les structures, les documents, mettre en place un centre d'excellence" précise Christophe Toulemonde.

2005 - 2007 : Les projets de Microsoft dévoilés

Durant l'été, les éditeurs travaillent encore et toujours. Même si nous vous avons déjà parlé des futurs produits de Microsoft, nous vous proposons dans cet article d'en savoir un peu plus sur les évolutions produits, les nouveautés autour du framework et des technologies, étant bien entendu qu'il est impossible ici de parler de tout. D'ici à la fin 2007, Microsoft aura terminé entièrement sa mue autour de .NET.

La liste des nouveautés de Microsoft est impressionnante. Si 2004 reste relativement calme, 2005 sera une année importante pour l'éditeur, avec une déferlante de nouveaux outils et bibliothèques. Whidbey est l'arbre qui cache la forêt et Longhorn modifiera en profondeur le modèle Windows tel qu'on le connaît actuellement.

Visual Studio .NET 2005 alias Whidbey (sortie: 2005)

VS 2005, plus connu sous le nom de code Whidbey, arrivera dans quelques mois. Il s'agit d'une étape importante, la plus importante depuis l'apparition de Visual Studio.NET. Cette version travaille sur les priorités suivantes : améliorations de l'IDE, nouveautés sur les langages et sur le framework .NET, développement Office, intégration avec SQL Server 2005 (Yukon), et enfin une meilleure prise en compte du cycle de vie, via des fonctions dédiées. VS.NET 2005 est prévue lors de la sortie du .NET framework 2.0. Le projet est donc ambitieux. Microsoft veut réduire de moitié le temps que l'on passe à développer une application. De plus, on doit bénéficier du support du code 64 bits (Win64). Outre les diverses améliorations autour des langages, plusieurs autres domaines subissent un lifting. Pour XML



Le travail de recherche sur l'interface 3D de Microsoft

et les données, VS.NET 2005 offrira un meilleur modèle de conception d'applications orientées données. L'implémentation d'ADO.NET sera facilitée avec une réduction par deux du code nécessaire, tout en ayant une maintenance et un déploiement réduit. Le développeur bénéficiera de nouveaux assistants pour les DataSources et DataSet (qui seront plus visuels). Pour les Smart tags, on notera qu'ils supporteront le développement itératif. Sur XML, les améliorations concernent l'intelligence, pour XSD et DTD.

Microsoft en profite pour renforcer les applications ASP.NET, ainsi que pour l'ensemble d'un développement Web. Whidbey permettra, sans configuration des applications Web, d'éditer des sites. On dispose des protocoles FPT, UNC et FPSE. Surtout, pour les sites à distance, on pourra faire des copies et remonter directement, via une interface graphique. Bref, un déploiement largement simplifié. Ici aussi, l'intelligence sera disponible. VS.NET se transforme ainsi également en éditeur Web... Toutes les options de validation seront disponibles. Microsoft précise que le module design ne modifiera pas le code. Précaution utile. Sur les

anciens projets (ASP.NET 1.x), la compatibilité sera assurée.

La construction de packages d'installation / d'application et le déploiement sont devenus des contraintes fortes. VS.NET 2005 n'oublie pas d'inclure de nouvelles possibilités. L'architecture de versionning de fichiers et d'assemblies avait déjà réglé un certain nombre de conflits. Demain, on disposera d'une architecture unique autour de MSBuild. MSBuild est ce que l'on nomme le build engine. Il s'agira du modèle de Longhorn. Il s'appuie sur XML et se présente comme le concurrent de Ant, sauce Microsoft. Il supportera l'ensemble des langages présents dans la CLR (ainsi que le C++). Pour créer des builds visuellement, on peut utiliser BuildStudio. MSBuild va au-delà du simple build. MSBuild permet aussi de faire de l'obfuscation (de code). Whidbey permet aussi d'utiliser le contrôleur de source de son choix sans à avoir besoin de bidouiller la base de registre.

Côté déploiement, c'est la technologie ClickOnce. Deux axes essentiels à retenir : déploiement rapide et facile, mise à jour automatique de l'application. Tout se déroule dans

la fonction Publish. Un assistant est là pour vous guider. Plusieurs modes de déploiement sont disponibles : sur le Web, FTP, en file share ou en local. Quand vous préparez le publish, vous spécifiez si l'application prend en compte ou non la mise à jour. À chaque lancement de l'application par l'utilisateur, le programme vérifie si une version plus récente existe. Bien entendu, on peut avoir accès à la fonction désinstaller. ClickOnce ne nécessite pas non plus d'être en compte admin ou root pour pouvoir installer une application et il n'est pas intrusif (No Touch Deployment). Le développement devra passer par le composant Bootstrapper pour vérifier les prérequis et pour exécuter les composants redistribués.

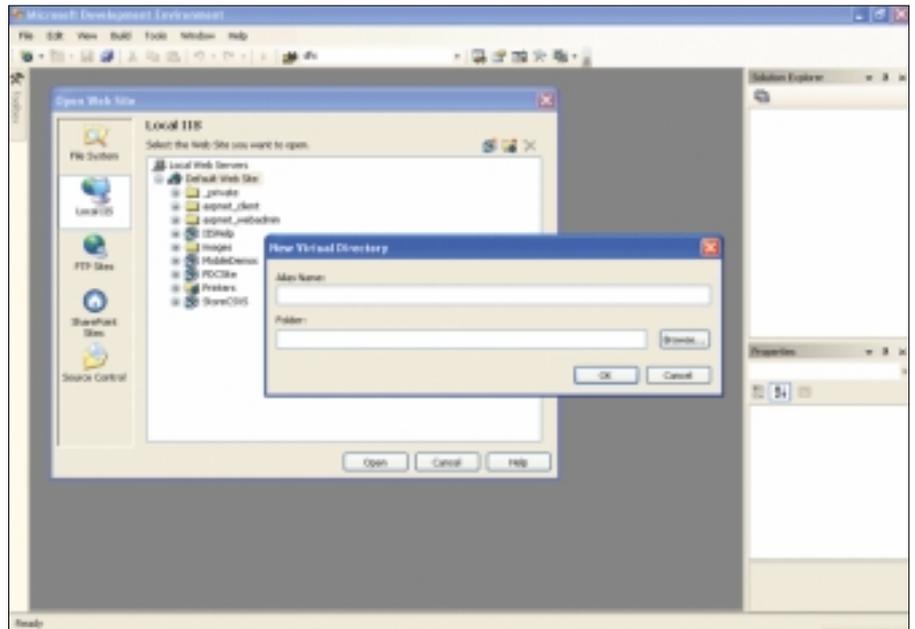
Parmi les nombreuses autres nouveautés / améliorations, on voit l'arrivée du refactoring. Plus besoin d'arrêter l'exécution de son code pour corriger les bugs et erreurs, les corrections se feront à la volée (la fonction est pour le moment utilisable avec VB.NET). Dans le module debug, il y a le support du 64 bits (sur processeur AMD et Intel et en code managé et non managé) et de Yukon (avec support debugging SQL et CLR). Vous pouvez même personnaliser l'affichage des données. Le debugger de Whidbey sera extensible (en partie). Dans la partie C++, le développeur bénéficie d'un debug plus souple sur les types C++ STL. Pour faire le debug, on aura le choix entre le local et le distant. On aura aussi droit au projet temporaire. Cela signifie que la sauvegarde se fait uniquement à la demande du développeur.

Team System (sortie : 2005)

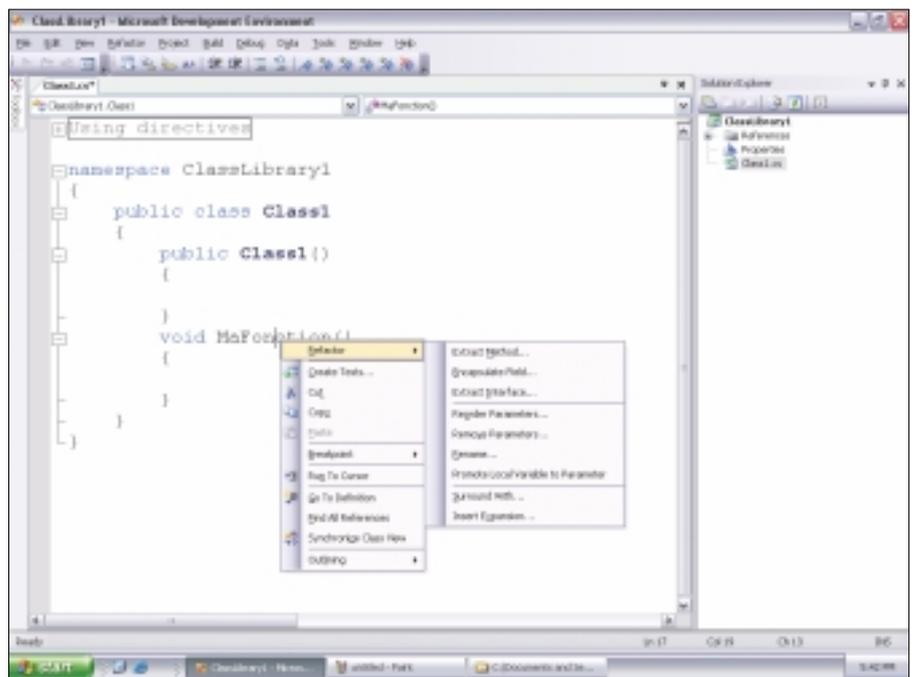
Microsoft reprend le chemin tracé par d'autres éditeurs : le cycle de vie de l'application. Cela passe par des outils de tests, d'optimisation, de travail en équipe et de modélisation. Il s'agit aussi de renforcer la qualité du code et la sécurité de celui-ci. Visual Studio 2005 Team System doit répondre à ces problématiques, un peu comme le définit Borland.

Trois axes sont définis par Microsoft : réduire la complexité, faciliter la collaboration et le travail en équipe, personnaliser et étendre. Cette nouvelle architecture de l'IDE s'utilisera sur une plate-forme serveur Windows.

Avec le Team System apparaissent plusieurs éléments. Tout d'abord, le Team Foundation, avec Work Item Tracking, un gestionnaire de projet, un gestionnaire de sources, ou encore un Integration Services. Puis, les nouveaux



Les nouveaux modules Web de VS.Net 2005.

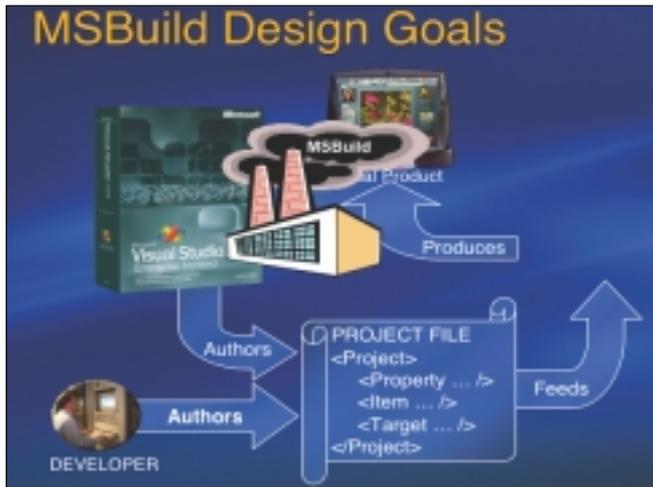


Fonction refactoring de VS.Net 2005.

outils proprement dits, autour de l'architecture et du design, du build, du test et de l'analyse de code. En pratique, le Team Foundation est la couche basse de Team System.

La partie test (et d'analyse) devient une pièce maîtresse du futur VS 2005. La Team Test Edition introduit toute une série de nouveaux outils, éprouvés en interne par Microsoft. Pour mieux intégrer le test au développement, on voit apparaître un nouveau type de projet, le Test Project. On retrouve les fonctions habi-

tuelles pour les tests : tests unitaires, tests Web, tests génériques, tests automatiques (ex. : test de montée en charge) et les tests manuels. Bien entendu, ils seront accessibles directement, via l'IDE et son interface. On trouvera notamment la fenêtre Test Explorer pour y gérer les tests. Très pratique, quand on doit créer des batteries de tests. De plus, une autre fonction permet de découvrir le code utilisé et non utilisé (= code coverage). Cela permet de réduire la taille du code et les bugs potentiels.



Le desktop 3D selon Sun

Visual Source Safe 2005 (sortie : 2005)

On parle de Whidbey, mais qu'en est-il de SourceSafe ? L'outil de versionning de Microsoft n'est pas abandonné, du moins pour le moment. Son système de contrôle doit, lui aussi, s'appuyer sur le moteur de SQL Server 2005 (Yukon). Visual Source Safe 2005 bénéficiera donc de nombreux ajouts et modifications : XML, Unicode, performances réseaux, localisation, remote access, etc. Cet outil vise avant tout les développeurs indépendants et les petites équipes, alors que le Team System vise plutôt les gros projets orientés entreprises et services même si officiellement ce n'est pas le cas. SourceSafe 2005 doit être disponible en même temps que VS.NET 2005.

Est-ce une version pour dire que le produit continue à exister ou une simple mise à jour qui signifiera à terme la fin de l'outil ? On ne sait pas encore.

VisualStudio 200x alias Orcas (sortie : 2006 - 2007)

Au-delà de Whidbey, Microsoft pense déjà au futur de la programmation. Cette version serait pleinement compatible avec le futur Windows Longhorn, dont la sortie est prévue courant 2006 (aux dernières nouvelles). Orcas deviendra sa plate-forme de développement. Il implémentera toutes les nouvelles technologies Longhorn, dont les interfaces managées (Avalon). De nouveaux outils d'interface seront disponibles, une sécurité revue et corrigée, et bien entendu le support du nouveau modèle de données (issu directement du XML et de Yukon = WinFS). Longhorn introduira un nouveau modèle de développement : XAML.

SQL Server 2005 alias Yukon : l'autre gros morceau (sortie : 2005)

Yukon, alias SQL Server 2005 est l'un des produits Microsoft les plus attendus. Il mettra enfin SQL Server au cœur du XML et de .NET (avec intégration dans VS.NET). Microsoft annonce plusieurs axes : la disponibilité, la montée en charge, la sécurité, la gestion, l'intéropérabilité. Tout cela devrait apporter une plus grande souplesse de développement et mieux intégrer les données dans les applications .NET. Il sera possible de coder des applications orientées données avec C# ou Transact-SQL, ou tout autre langage de la CLR. L'outil inclura un module OLAP ou encore du datawarehouse. Yukon implémente le Code Access Security. Cette fonction contrôle le code pour qu'il ne puisse pas corrompre le système ni la base de données. L'intégration dans .NET permettra aussi d'avoir, en théorie, un meilleur code, la CLR vérifie le code managé. Il faudra aussi, dans certains cas, choisir entre T-SQL et le code managé. Au développeur de déterminer le bon usage. Un langage CLR sera plus pertinent dans les fonctions intensives CPU ou les procédures complexes. Par contre, côté performance, T-SQL pourra être plus approprié. L'interfaçage avec ADO.NET sera meilleur, car il supporte les types SQL. La grosse nouveauté dans le middleware de données et le mapping concerne la présence d'ObjectSpaces. Sur ObjectSpaces, notons que le projet fusionne dans WinFS (le futur File System), ObjectSpaces servant de persistance à WinFS et offrant ainsi, un mapping relationnel objet unique et unifié. La technologie devait faire partie, à l'origine, du package Whidbey. Cette annonce retardera donc la disponibilité d'ObjectSpaces et sera

disponible en téléchargement séparé. Yukon inclura aussi Xquery, un langage bien utile pour les données XML. On notera aussi la présence de deux nouveaux frameworks : SQL Server Service Broker et Reporting Services. Le Service Broker sert dans le cadre d'applications distribuées (avec du messaging asynchrone pour la base de données). Reporting Services permet de créer des rapports. Yukon permettra de créer, gérer et voir des rapports. Les services de notifications sont une partie importante du prochain SQL Server, pour mieux déployer des applications et prévenir les utilisateurs. La mobilité n'est pas oubliée avec le SQL Server Mobile Edition v3.

C ω (C Omega) : un projet prometteur (sortie : ?)

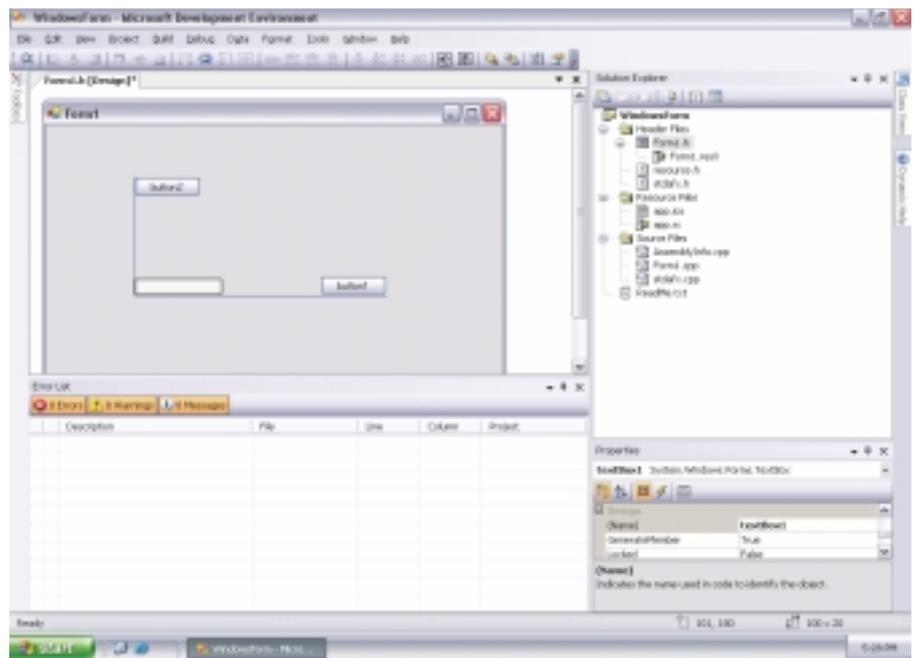
Microsoft travaille sur les langages actuels de .NET mais prépare l'avenir. Celui-ci passe notamment par le nouveau langage C ω. Connu aussi sous le nom Xen, C ω doit permettre de mêler la programmation XML à la programmation objet et à prendre en compte SQL. Il s'agit de développer rapidement avec ces trois éléments. Le prototype du langage est sorti en avril 2003. Depuis, le travail n'a pas manqué, notamment chez Microsoft Research en Angleterre. Cela a permis de pouvoir développer des applications multithread et des Web Services, à partir de ce nouveau langage. En interne, ce langage intéressera les équipes développant le futur Windows, SQL Server et Visual Studio. Plus concrètement, C ω est un projet de recherche qui regroupe deux autres projets le Polyphonic C# et Xen. Aucun plan d'intégration dans les produits n'a pour le moment été défini. Cela dit, la version actuel-

le de C# se présente comme une extension de C#, utilisable dans Visual Studio.

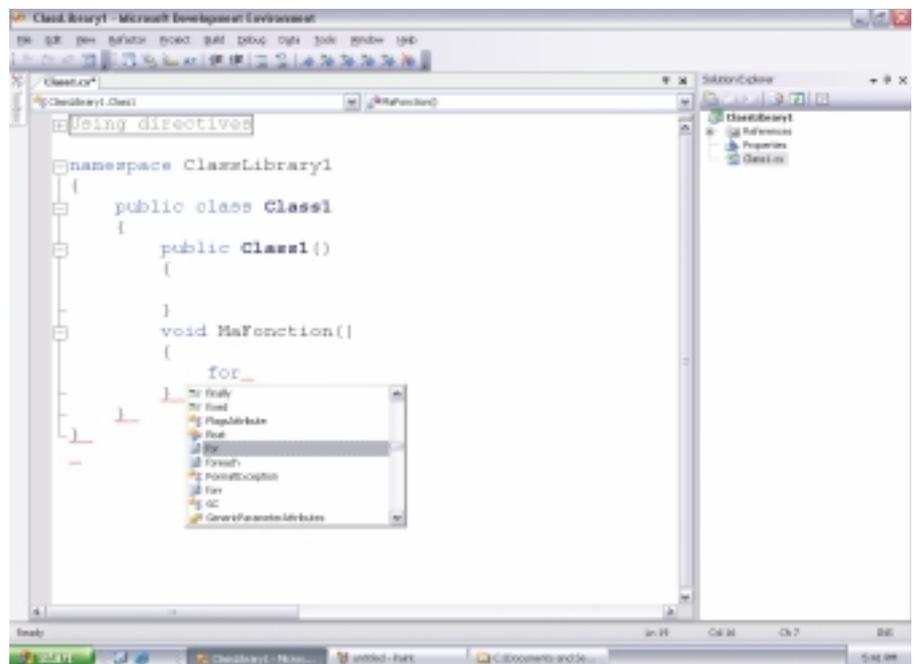
Autour de Windows Longhorn (sortie : 2006 – 2007)

Longhorn & WinFX : s'y préparer

Si Longhorn n'arrive qu'en 2006 (voire en 2007), ce n'est pas pour autant qu'il faut l'attendre sans réagir. On oublie de le dire, mais Longhorn annonce une refonte en profondeur du modèle des applications. Avec la présence au cœur du système de XML, SOAP, .NET et de l'architecture SOA, il faut repenser le concept et le fonctionnement de l'application. Plus que jamais, il faudra penser code managé et structurer l'application, selon les concepts SOA. Au-delà de cela, il y a toute la compatibilité entre les WinForms actuelles et les nouvelles technologies Longhorn. Prenez l'interface. Longhorn introduit la technologie Avalon, le nouveau modèle d'interface Windows, basé sur .NET. Faut-il ou non migrer une application WinForm en Avalon ? Microsoft propose 3 solutions : garder l'application actuelle en utilisant des composants Avalon, créer une application Avalon et réutiliser les contrôles forms de l'ancienne application, ou alors, remplacer entièrement l'interface par Avalon. Bien entendu, l'éditeur conseille d'utiliser les classes Avalon si l'application fonctionne uniquement sous Longhorn. Pour une utilisation dans plusieurs versions de l'OS, là, il faut avoir une approche mixte. Dans Longhorn et dans le travail préparatoire vers ce nouvel OS, la question de l'interopérabilité entre le code managé et non managé demeure au cœur du modèle de développement et des préoccupations. La CLR continuera à prendre en compte les composants non managés : COM, Com+, API Win32, etc. Il faudra suivre scrupuleusement les guidelines et les pratiques Microsoft. Sur Com+, on ne sait pas si dans Longhorn il montera dans la CLR ou non. Avec Longhorn apparaît un ensemble d'API entièrement managé, WinFX. WinFX est le domaine de développement du futur Windows, même si pour la compatibilité, Win32 demeure présent. Cela rappelle le modèle d'Apple autour de Carbon, pour assurer la compatibilité des applications entre MacOS 9 et X. Bien entendu, au cœur de WinFX, l'ensemble .NET. Contrairement à Win32 réalisé en C, WinFX est réalisé en langage .NET. Si aujourd'hui, le framework .NET ne permet pas d'accéder à l'ensemble des fonctions du système, demain, cela sera une réalité. Les



Le RAD de VS.Net 2005.



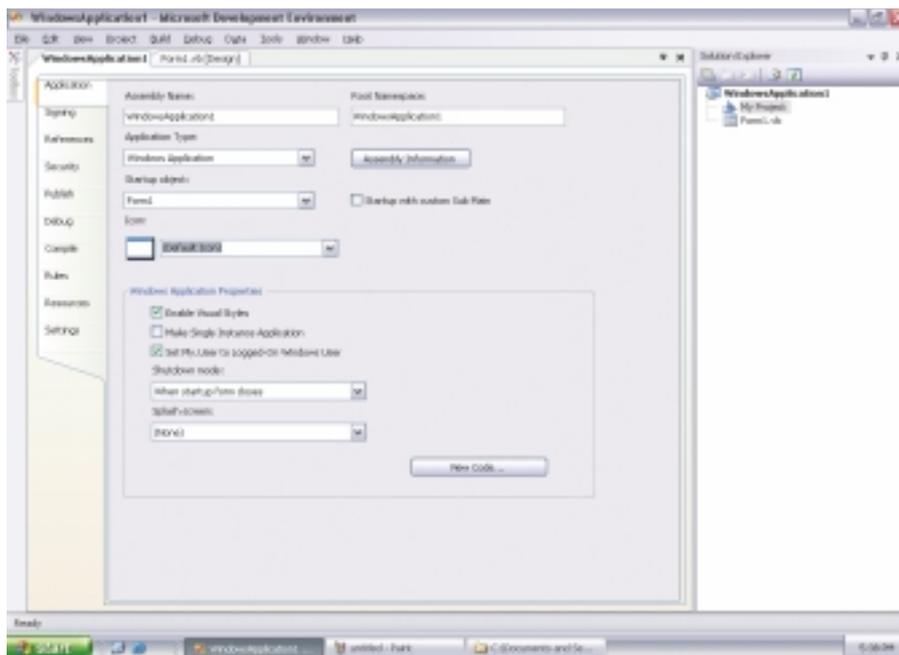
L'intellisense est généralisé partout dans VS.Net 2005.

fonctions de Longhorn seront exposées uniquement dans WinFX (donc pas de Win32).

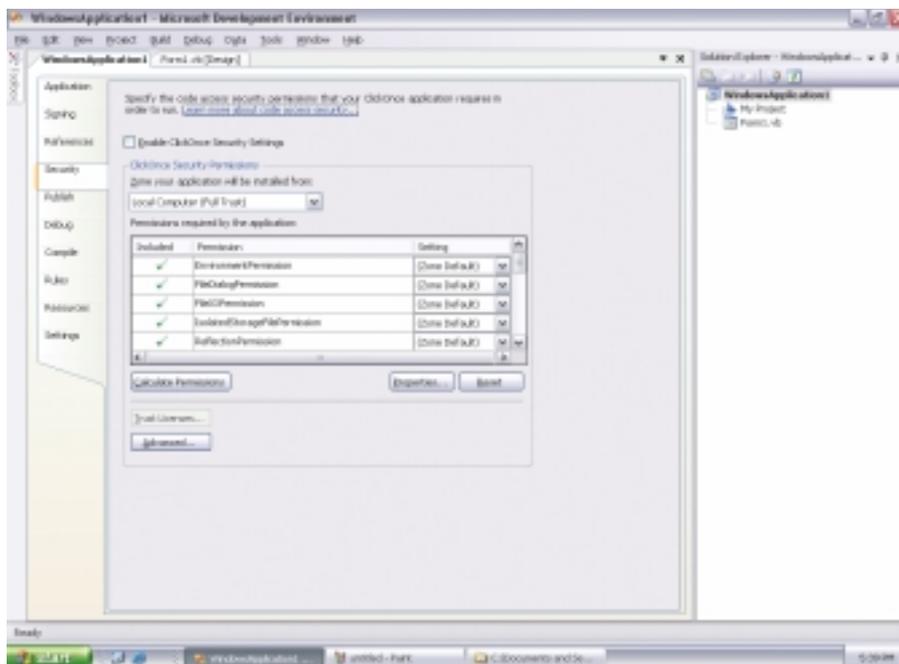
WinFS

Des rumeurs récentes faisaient état du retrait de WinFS de Longhorn... Il faut dire que WinFS est une révolution pour le file system de Windows. Il semble pourtant que cette technologie soit plus délicate à mettre au point que prévu. La version implémentée dans Longhorn pourrait perdre quelques fonctions officielle-

ment mineures, mais un certain flou demeure. WinFS s'appuie sur le moteur de Yukon. Le concept de WinFS est une " virtualisation " des fichiers. Au lieu de stocker son fichier dans un emplacement unique, il sera possible avec WinFS de classer et filtrer ses fichiers dans des dossiers dont on aura défini les critères. Cela signifie que l'on pourrait voir des fichiers non pas, par dossiers classiques, mais par contexte. Ce fonctionnement implique obligatoirement une machine puissante pour traiter



Les options assembly de VS.Net 2005.



Les nouvelles sécurités de Whidbey.

le flux de données et surtout, pour l'indexation des données qui demande des ressources systèmes énormes. Ce concept est révolutionnaire pour Windows mais n'oublions pas qu'en 1995, BeOS possédait déjà une telle structure et il est possible qu'Apple propose dans la prochaine version de MacOS X la même chose (courant 2005 ?). WinFS doit permettre de trouver plus facilement des données / fichiers (vive l'indexation). Les applications actuelles fonctionneront tout de même.

Avalon, XAML, Aero

Apple avait frappé un grand coup avec l'interface Aqua de MacOS X et le moteur utilisant le format PDF ainsi qu'une accélération native matérielle via Quartz Extreme. Finalement, Microsoft va aussi dans ce sens avec la technologie Avalon dans Longhorn. Le format graphique et le modèle n'avaient guère changé depuis Windows 3.x mais uniquement des mises à jour. Avalon est donc la couche de présentation s'appuyant sur un moteur graphique

vectériel. Avalon saura tirer parti des cartes graphiques modernes pour accélérer l'affichage et les traitements. Avalon poursuit 7 buts (que Microsoft appelle buts primaires) : utilisation du matériel, intégration interface – media – documents, intégration avec .NET, implémentation d'un langage déclaratif, focus sur la composition, assurer compatibilité et interopérabilité, et enfin, tirer parti des leçons des 10 dernières années. L'interface n'est pas Avalon en tant que tel. Elle se définit, via des fichiers XML écrits avec XAML. On aura droit aussi aux petits gadgets que l'on voit dans MacOS X : transparences, qualités des icônes, animations diverses...

Un des éléments de la nouvelle interface est le langage XAML. XAML est un langage déclaratif. Il se base sur XML. Il permet de spécifier une hiérarchie des objets en incluant des propriétés et la logique. XAML permet de séparer l'interface proprement dite de la logique. Ce mélange demeure un souci de programmation très grand dans les projets itératifs et la maintenance. Comme XML ne sait pas encore contrôler très bien les flux, on génère un code mixte : code et XAML, bien qu'il est possible de créer une application 100 % XAML. XAML est utilisable sur le framework .NET 1.x grâce à divers projets open source dont Xamlon (<http://www.xamlon.com/>).

Exemple Hello Word :

```
<Window xmlns="http://schemas.microsoft.com/2003/xaml" Visible="true">
  <SimpleText Foreground="DarkRed" FontSize="14">Hello World!</SimpleText>
</Window>
```

Aero est un élément encore à part. Il s'agit d'une interface graphique complémentaire utilisant massivement DirectX et la 3D. Il s'agit d'avoir sur son desktop une interface 3D (même partielle). Ce type d'interface n'a rien d'une nouveauté, les travaux ont été nombreux autour de cela. On pense à IBM et Apple (durant le projet Copland il y a plus de dix ans avec le Finder 3D). Actuellement, Sun travaille beaucoup autour de cela et prévoit de sortir quelque chose courant 2005. Aero est bien entendu un module additionnel. Il devrait être livré avec Longhorn.

Voilà pour le petit tour dans le pays des nouveautés Microsoft. Nous reviendrons très prochainement sur tout cela avec différents articles.

■ François Tonic

“Le plus important va au-delà de Whidbey”

Dans un dossier consacré aux futurs outils de Microsoft, il était normal de rencontrer des représentants de la société. Programmez ! a rencontré Olivier Ezratty (Directeur Division Développeurs et Plate-Forme d'Entreprise – Microsoft France) et Jean-Christophe Cimetière (Chef de Produit Plate-Forme d'Entreprise – Microsoft).



Olivier Ezratty

Programmez ! : Beaucoup de regards se tournent vers Whidbey. Avez-vous le même réflexe ?

Olivier Ezratty : Le plus important va au-delà de Whidbey. L'attente du marché est souvent double : étendre les produits, assumer une partie de la chaîne.

Mais l'intégration de nouveaux outils ne doit pas se faire au détriment d'interopérabilité. Il faut réaliser deux choses : intégration et interopérabilité. Nous proposons une approche intégrée.

Programmez ! : L'accord Sun – Microsoft va-t-il aboutir à une meilleure compatibilité entre .NET et Java ?

OE : La roadmap de l'accord est encore un peu floue. L'accord signé était un accord cadre. Il va y avoir des groupes de travail organisés avec de grands clients. Ce qui a été annoncé concerne le côté intégration d'infrastructure, telle que l'interopérabilité serveur – serveur, serveur – client, ainsi que les outils d'identification. Les détails sur les évolutions de l'intégration .NET avec J2EE ne sont pas encore figés.

Programmez ! : Depuis l'annonce de SQL Server 2005 (Yukon), il y a beaucoup d'attentes autour de ce produit. Est-ce pour combler un certain retard sur la concurrence ? L'enjeu de Yukon dépasse-t-il son propre environnement ?

OE : Se rapprocher de la concurrence ? On a plus un problème marketing qu'un problème technologique avec SQL Server dans la couverture des besoins des clients ! On répond aujourd'hui à 95 % des besoins. Yukon va innover dans des domaines où la concurrence n'est pas forcément présente : comme l'intégration de la CLR dans le moteur et celle de nombreux outils de décisionnel en standard. L'intégration de XML est aussi importante. Par contre, effectivement, il y a aussi beaucoup de progrès qui répondent aux attentes des clients en matière d'exploitation pour les applications les plus exigeantes : dans le transactionnel, l'administration, les services de transformations ou les backups. Et puis le cœur

de Yukon sera utilisé dans le système de gestion de fichiers WinFS de Longhorn !

Jean-Christophe Cimetière : Précisons tout de même que Yukon demeure fondamentalement un SGBD.

Programmez ! : Comment définir en quelques mots Visual Studio .NET 2005 ?

OE : Je dirais : productivité par l'évolution des langages (C#, VB.NET, C++ ; edit and continue), du framework .NET qui permet de réduire le nombre de lignes de code et les évolutions de l'IDE ; support d'Office et de Yukon, et enfin la gestion du cycle de vie des applications, car la vie du logiciel ne s'arrête pas au développement. Depuis 1 à 2 ans, nous encourageons les développeurs à passer à .NET. Avec Longhorn, cela prendra encore plus de sens, car on accèdera à 100 % des fonctions du système, via le framework.

JCC : J'y ajouterais l'approche intégrée, l'interopérabilité et l'industrialisation.

OE : La productivité peut s'améliorer. On fait en sorte que les outils tiers puissent s'intégrer dans notre outil.

Programmez ! : En dehors de Team System dédié aux équipes de développement, Microsoft a dévoilé ses plans autour de Visual Source Safe. Faut-il opposer cet outil à Team System ?

JCC : Visual Source Safe peut suffire à un développeur et à une petite équipe. Au-delà, le versioning ne suffit plus. Dans Team System, il y a le mot Team, équipe.

Programmez ! : Comment le développeur peut-il appréhender l'arrivée de VS.NET 2005 et de toutes les nouveautés ?

OE : Il est probable que 9/10e des développeurs se forment par eux-mêmes. Ils doivent tester, se former aux nouveaux outils. Nous investissons évidemment sur la communication online à cet effet, notamment avec des webcasts. Aujourd'hui, l'un des défis est de faire adopter

de nouvelles technologies, de nouvelles méthodes d'architecture, telles que SOA, les Services Web, XML... d'où l'importance de nos prochains outils. Notre souhait est de démocratiser l'accès à ces technologies et nouvelles méthodes de travail. À un certain stade, l'ancien projet, l'ancien code revient plus cher à maintenir que de tout redévelopper. Il y a un dilemme : il faut gérer l'existant et la nouveauté. Pour un développeur VB6, la productivité de VS.NET / VB.NET est un argument pour le faire migrer. Mais paradoxalement, on constate que .NET est plus facile à adopter par les développeurs Java que VB6. Whidbey va réduire cet écart et faciliter la migration des compétences pour les développeurs VB6.

JCC : Le prix est aussi un élément important. Il y aura des annonces tarifaires et de packaging au prochain TechEd Amsterdam, fin juin 2004. Il s'agit aussi de rendre accessible la version de base de VS.NET au plus grand nombre. La première chose à faire est de s'autoévaluer sur la programmation objet, le développement web... afin de savoir où l'on est exactement.

Programmez ! : La sécurité serveur et du code source est-elle devenue un des chevaux de bataille de Microsoft ?

OE : La sécurité est un enjeu qui concerne aussi les développeurs. Il faut une prise de conscience. De plus en plus d'applications sont exposées sur le Web. À la Conférence consacrée à la sécurité (en mai 2004), nous avons eu plus de 4 000 participants, dont environ 1 500 développeurs. On investit beaucoup dans ce domaine. Il faut produire du code sécurisé. Prenons le SP2 de Windows XP. Il est très important, notamment pour les éditeurs. Le système devient plus rigoureux. Certaines applications communicantes seront vues potentiellement comme douteuses par le système. Il faut plus de sécurité, plus de vigilance.

Programmez ! : Com / Com+ continueront-ils à exister dans Longhorn et .NET ?

JCC : Dans Longhorn, Com / Com+ demeure une brique du système pour des raisons de compatibilité ascendante. Il n'est pas prévu de la faire "sauter". En revanche, dans le Common Language Runtime de Longhorn, tous les mécanismes transactionnels seront disponibles nativement. Ce nouveau mode transactionnel sera utilisable aussi bien en environnement distribué via la couche Indigo, qu'au sein d'une application.

■ *Propos recueillis par François Tonic*



Jean-Christophe Cimetière

Hélice recherche un " savoir- être "

Au sein du groupe Hélice se retrouvent tous les métiers traditionnels d'une SSII. La société met cependant l'accent sur la qualité en étant certifiée Iso 9001 version 2000. Elle attache donc une attention toute particulière aux compétences techniques bien sûr, mais aussi au comportement de ses collaborateurs, avec un accent mis sur l'aspect formation.



Le groupe Hélice est constitué de trois sociétés aux noms poétiques : Hélice pour les prestations d'ingénierie de production et d'exploitation de systèmes informatiques, Hélicium pour la partie étude, conception et développement de systèmes d'information et enfin Hélios pour l'activité de conseil et d'expertise.

Présente surtout en région parisienne (60 à 70 % de son activité), la société compte trois implantations en régions, à Nantes, Lyon et Toulouse.

Diversité des technologies

Les développeurs et autres profils techniques peuvent travailler au sein d'Hélicium ou d'Hélice, l'activité de gestion des infrastructures comportant bien entendu des aspects de développement. Au niveau des technologies utilisées, le groupe a suivi l'évolution de ses clients : de COBOL/CICS/DB2, elle s'est ensuite orientée vers les technologies Objet, C et C++, puis plus récemment vers .Net et Java, avec C#, J2EE et autres servlets. Sans oublier SQL pour les accès aux bases Oracle, PHP, UML et XML et plusieurs AGL, dont Visual Age pour VB. Alain Ripeau, directeur d'agence du groupe Hélice, indique qu'il n'y a pas de limite au niveau des technologies utilisées et que les compétences requises dépendent avant tout des attentes du marché. Les " legacy " restent importants et assez centraux dans les systèmes d'information des grands comptes

clients d'Hélice, Cobol et CICS sont encore pratiqués, sans oublier bien sûr, Packbase.

Passion, compétences techniques et savoir - être

Alain Ripeau souligne que la qualité principale recherchée chez un futur collaborateur d'Hélice est le " savoir - être ". Avant même la technique, car souligne-t-il : " un langage se réapprend ". Il reconnaît que sur la population de programmeurs, il est parfois difficile de trouver ces qualités, ceux-ci ayant une certaine tendance à se réfugier dans la programmation. " Dans un projet, tout est imbriqué et il est absolument nécessaire de communiquer avec les autres ", rappelle-t-il. Hélice recherche donc des passionnés, mais pas des bidouilleurs " ce qu'il y a de plus dangereux ", selon Alain Ripeau, dans la mesure où " le côté procédure prend de plus en plus d'importance ". Il regrette que l'expression écrite, comme orale, de certains candidats soit quelque peu déficiente. Qui plus est, le collaborateur doit savoir s'intégrer. Parmi les diplômés que la société apprécie, celui de l'Epita, mais Alain Ripeau indique que la proportion de profils réunissant toutes les quali-

tés requises est à peu près similaire d'une école à l'autre...

Face à ces difficultés de savoir- être, le groupe s'est orienté " vers des notions de coaching ", liées à sa certification Iso 9001, qui prend en compte la qualité des collaborateurs. " Le manque de savoir - être ou la rétention d'information sont des fautes professionnelles " prévient Alain Ripeau.

Une université du soir

Le groupe Hélice dispose de son propre centre de formation et met un accent particulier sur la formation continue permanente. Une des initiatives les plus originales est l'université du soir : des sessions sont organisées de 19 à 21h afin de permettre aux ingénieurs en poste chez les clients d'y assister, s'ils le souhaitent, le principe étant celui du volontariat.

Par ailleurs, les ingénieurs en inter contrat sont dans les locaux de la société, et s'autoforment ou forment les autres. Objectif souligne Alain Ripeau : développer au maximum "l'employabilité" des ingénieurs, ce qui peut aller jusqu'à la certification.

Pour en savoir plus : www.groupehelice.fr.

■ Carole Pitras

150 collaborateurs nouveaux en 2004

Le groupe a réalisé l'an dernier un chiffre d'affaires de 41 M€, en légère progression, fait suffisamment rare pour être remarqué, pour un effectif de 500 collaborateurs. Pour l'année 2004 un chiffre d'affaires de 45 M€ est prévu, avec un effectif de 560 personnes. Pour y parvenir, le groupe a prévu de recruter, compte tenu des départs, 150 nouveaux collaborateurs, dont 30 concepteurs développeurs, 13 chefs de projet et 7 architectes.

Développer sécurisé

2^e partie

Y a-t-il des différences d'approche et de méthode pour sécuriser le code source ?

Un de mes anciens collègues, renommé pour ses compétences en calculs physiques et en algorithmique mathématique, redoutant une retraite subite, avait pris pour habitude de verrouiller tout accès à son code Fortran, tout en le laissant consultable par toute personne du service ! Attelé à la résolution d'un problème, il noircissait quelques feuilles de papier en posant le problème sous forme d'équations, puis codait son algorithme en utilisant pour noms de variables celles de ses lignes manuscrites. Impossible de comprendre les fonctions en lisant le code source ! Cette méthode consistait donc à rendre le code intentionnellement incompréhensible pour ses collègues, cependant, une fois une librairie compilée et livrée chez des clients, même si le Fortran avait été dûment commenté et les variables correctement nommées, il aurait été alors très difficile de reconstituer, à partir de la version binaire, un code lisible et compréhensible. Aujourd'hui, les langages tels que .Net et Java s'appuient sur les concepts d'autodescription par l'utilisation de métadonnées. Par essence, il est donc très simple, à partir des versions compilées des bibliothèques, de restituer une version lisible du code. Les développeurs et les éditeurs sont donc confrontés à un nouveau problème : la protection de leurs sources et de leur propriété intellectuelle, résolue par la maîtrise d'une technique, connue sous le nom d'obfuscation.

OBFUSCATION

L'obfuscation repose sur le même principe que la méthode utilisée par mon ancien collègue : rendre le code illisible sans altérer son comportement. Il s'agit d'un procédé qui est appliqué sur une assembly et qui ne modifie en rien le code source qui a servi à la générer. L'obfuscation va donc produire une nouvelle assembly que les désassembleurs MSIL pourront toujours ouvrir, mais qui sera si incom-

préhensible, que cela rendra le reverse engineering quasi impossible, protégeant ainsi le code source. La figure 1 montre le résultat dans Ildasm d'un désassemblage après une opération complète d'obfuscation. Afin de parvenir à un tel degré de confusion, tout en préservant un fonctionnement optimal, les outils d'obfuscation vont principalement s'appuyer sur 2 opérations : le renommage et la suppression des métadonnées inutiles. Ces techniques sont ensuite complétées selon les éditeurs par d'autres procédés.

Renommage :

Les noms symboliques utilisés lors de la programmation sont d'une grande aide pour la compréhension d'un algorithme. Modifier ces noms afin de les rendre incompréhensibles augmente donc énormément les efforts pour effectuer du reverse engineering sur le code.

Le premier rôle de l'obfuscateur sera donc de supprimer l'utilisation de tous les noms contenus dans l'assembly originale afin de les remplacer par un autre nom qui n'aura aucun sens particulier. Ce nom peut être généré automatiquement par un algorithme de transformation du nom ou choisi de façon aléatoire et être ainsi complètement découplé de l'original. Cette dernière technique assure une plus grande protection, se mettant à l'abri de la découverte de l'algorithme de transformation. Les noms générés par les obfuscateurs sont en général des caractères qui n'ont aucune signification et qui ne sont pas forcément alphabétiques; bien sûr, dans le contexte d'une assembly devant exposer ses types à d'autres outils, il est important de ne pas opérer la transformation de nom. Par exemple, si votre logiciel offre un ensemble d'API qui permettent de développer des add-ins, cet API doit toujours conserver le même nom et de préférence un nom compréhensible pour les développeurs des sociétés tierces. L'obfuscateur peut

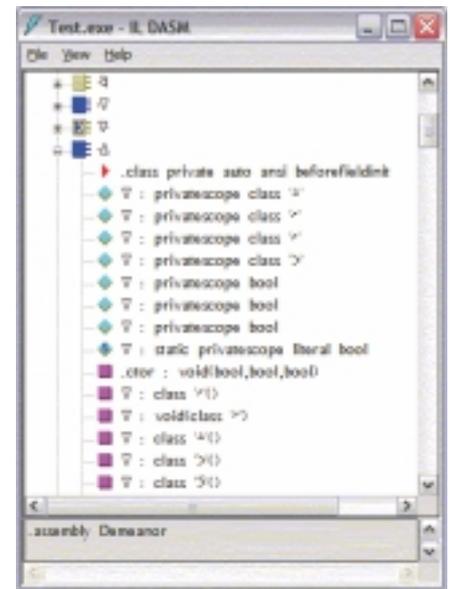


Figure 1

atteindre un degré de confusion supérieur, en affectant le même nom à plusieurs méthodes qui n'ont rien à voir entre elles. Ainsi, le remplacement ne produit pas systématiquement un nouveau nom pour chaque nom original. Par exemple, sur la figure 2, on constate que les méthodes `ChangeSize()` et `ChangeText()`, aux aspects fonctionnels différents ont été renommées avec le même nom : " f ". De plus, les obfuscateurs agissant sur le langage MSIL qui permet de réaliser des surcharges de méthodes, dont les types de retour sont différents (à l'instar de C#), la méthode `IncreaseSize()` peut elle aussi être renommée "f". Certains obfuscateurs offrent même la possibilité d'utiliser le même nom, non seulement pour des méthodes, mais aussi pour les types et leurs propriétés. Cette technique de réutilisation maximale du même nom s'avère très efficace pour réduire la lisibilité du code par un humain, néanmoins, elle ne modifiera guère le travail des désassembleurs qui commenceront alors par tout renommer avec des

noms différents ! Il s'agit donc de doser son utilisation en s'efforçant d'anéantir la lisibilité pour un développeur et pour un programme automatique.

Suppression des métadonnées inutiles :

Certaines métadonnées sont renseignées dans l'assembly, afin d'être lues par des outils de développement tels que Visual Studio.Net. Ces informations ne sont absolument pas utiles à l'exécution du code et peuvent donc être supprimées. Ainsi, le code MSIL n'a pas la notion des propriétés ou des événements qui sont transformés par les compilateurs en des notions de langage, moins évoluées, tout en préservant dans l'assembly la notion de haut niveau, afin qu'elle soit correctement présentée par des outils de développement. Par exemple, à partir du code original de la figure 3, on constate sur la figure 4a que la propriété `FontSize` a été transformée en MSIL en 2 méthodes : `get_FontSize()` et `set_FontSize()`. D'après l'appel effectué à la propriété dans le constructeur de la classe (figure 4b), on constate de même que le MSIL ne s'appuie pas sur la propriété `FontSize`, mais bien sur la méthode `set_FontSize`, générée par le compilateur. Pourtant, la propriété identifiée par un petit triangle rouge dans `Ildasm` n'a pas été supprimée. Elle n'est en fait là que pour faciliter le travail d'outils tiers et n'est pas utile au bon fonctionnement de l'application. L'obfuscateur va donc réaliser ici 2 opérations pour travestir le code, supprimer la propriété inutilisée, et modifier le nom automatique des accesseurs.

AUTRES TECHNIQUES

Modification du chemin d'exécution du code : Cette technique consiste à modifier la façon dont sont produites les structures de branchement ou de boucles par le compilateur, rendant difficilement compréhensible le chemin d'exécution du code. La figure 5 montre un exemple de modification d'une boucle imbriquée avec des branchements conditionnels.

Cryptage des chaînes de caractères :

Les chaînes de caractères constantes insérées dans le code peuvent être des portes d'entrée pour des personnes malveillantes, cherchant à repérer des mots tels que "password" ou "serial key" dans l'assembly. Les obfuscateurs proposent donc généralement de crypter auto-

```
public class SizedText
{
    private int fontSize;
    private string text;

    public SizedText(int FontSize, string Text){}

    public void ChangeSize(int FontSize){}
    public int IncreaseSize(int increment){}
    public void ChangeText(string Texte){}
}

public class a
{
    private int b;
    private string c;

    public a(int a, string b){}

    public void f(int a){}
    public int f(int increment){}
    public void f(string b){}
}
```

Figure 2

```
namespace MetaDonnee
{
    public class SizedText
    {
        public SizedText(int size)
        {
            this.FontSize = size;
        }

        public int FontSize
        {
            get{return _fontSize;}
            set{_fontSize=value;}
        }

        private int _fontSize;
    }
}
```

Figure 3

```
public void d()
{
    if (this.o > 0)
    {
        goto i0;
        do
        {
            while (true)
            {
                this.a(this.p[this.o - this.o / 20 * 20]);
                this.a(this.d);
                goto i1;
            }
            b: this.q = this.q.Substring(0, this.q.Length - 2);
        } while (this.q.Length < 2);
        i0: this.o = this.o - 1;
        goto b;
    }
}
```

Figure 5

```

SizedText::.ctor : void(int32)
.method public hidebysig specialname rtspecialname
instance void .ctor(int32 size) cil managed
{
    // Code size      14 (0xc)
    .maxstack 2
    IL_0000: ldarg.0
    IL_0001: call      instance void [mscorlib]System.Object::.ctor()
    IL_0006: ldarg.0
    IL_0007: ldarg.1
    IL_0008: call      instance void MetaDonnee.SizedText::set_FontSize(int32)
    IL_000d: ret
} // end of method SizedText::.ctor

```

Figure 4b

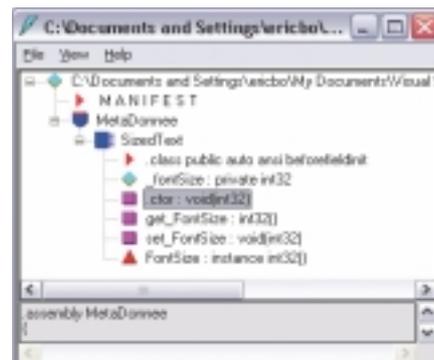


Figure 4a

matiquement les chaînes de caractères statiques, ces dernières étant décryptées lors de l'exécution. Il est important de noter que ce procédé n'offre pas une protection complète, en effet, afin de pouvoir décoder la chaîne de caractères, l'obfuscateur insère un appel à la méthode Decrypt() qu'il a ajouté dans l'assembly pour chaque chargement d'une chaîne de caractères constante. Il est donc possible de réussir à retrouver cette méthode et les endroits où elle est appliquée, afin de régénérer les constantes. De plus, l'exécution se voit un peu alourdie par les appels supplémentaires. Néanmoins, cette barrière supplémentaire réduit encore les intrusions possibles

dans le code source de l'assembly.

Cryptage de l'assembly

Un projet de nouvel obfuscateur (www.InvisiSource.com) propose la mise en place d'une technique novatrice : le cryptage de l'assembly. Après obfuscation, l'assembly serait totalement crypté puis livrée avec un chargeur spécifique qui la décrypterait au moment de son chargement en mémoire.

Les décompilateurs devraient alors apprendre à récupérer le code MSIL en mémoire, plutôt que sur disque, ce qui est une nouvelle barrière dans la protection contre le désassemblage.

COMPLEMENTS

Visual Studio .Net 2003 fournit une version "Community Edition" de Dotfuscator qui vous permettra de tester l'obfuscation sur vos assemblies, cette version est bridée et ne réalise que les deux opérations principales qui sont : le renommage et la suppression des méta données inutiles. Parmi les autres produits qui retiennent l'attention, le premier, largement salué par les spécialistes se nomme Déméanor for .Net et le second Salamander .Net proposant notamment un outil complémentaire appelé Protector.

■ **Éric Bouguen** *Consultant senior en développement de solutions Microsoft (Avanade)*

Prévenir les failles de sécurité



Trois failles communément exploitées au niveau du logiciel sont : les expressions SQL mal définies, le buffer overflow et les exceptions non catchées à l'exécution.

L'approche classique de l'industrie est d'éviter ces attaques en produisant du code, puis en exécutant un genre de "tests du singe" dans l'intention de simuler les actions d'un pirate. Les testeurs essaient de designer et exécuter un nombre et un spectre de tests, qui attaquent l'application par autant d'angles que possible, dans l'espoir qu'ils mettront une faille de sécurité à jour. Celle-ci peut ainsi être corrigée avant le déploiement de l'application.

Une autre façon, plus simple, de protéger votre code contre ces trois attaques courantes, est de compléter les tests de sécurité par un effort concerté de prévention des failles introduites lors de la programmation. La méthodologie AEP (Automated Error Prevention) de Parasoft offre une solution efficace et réalisable de prévention des trous de sécurité, par l'application automatique de bonnes pratiques telles que : les analyses statiques et dynamiques, le test unitaire, et la détection des erreurs à l'exécution. En appliquant ces mesures préventives, vous commencez à améliorer la sécurité de votre code, avant même d'écrire des cas de tests spécifiquement pour les tests de sécurité. Vous faites ainsi un bond en matière de mesures prises et profitez d'un moyen pratique pour améliorer la sécurité de votre application. Grâce aux pratiques AEP recommandées sur tout le cycle de développement logiciel, il est possible de corriger de nombreuses failles de sécurité et améliorer la qualité et fiabilité du code.

L'injection de code SQL

Lorsque des expressions SQL sont créées de façon dynamique à l'exécution du programme, il y a ouverture d'une brèche de sécurité : si le pirate a la possibilité de faire passer des données fixes dans une expression, ces entrées peuvent devenir une partie de l'expression.

Exemples d'injection de code SQL

L'exemple suivant représente un système qui a été conçu pour restreindre l'accès du site aux utilisateurs enregistrés. Malheureusement, un pirate peut utiliser cette technique d'injection de code SQL pour pouvoir faire beaucoup plus de choses.

```

Http login form:
<FORM name=login action=login.jsp METHOD
=post>
User name: <input name="USER">
<br>

```

```

Password: <input type="password" name=
"PASSWORD">
<br>
<input type="submit" value="Go">
</FORM>
login.jsp:
<%@ page import="java.sql.*" %>
<%@ page import="java.io.*" %>
<%
//MySQL
String DRIVER = "org.gjt.mm.mysql.Driver";
String DBURL = "jdbc:mysql://localhost:33
06/fruits";
String LOGIN = "fruits";
String PASSWORD = "fruits";
Class.forName(DRIVER);
Connection connection = DriverManager.get
Connection(DBURL, LOGIN, PASSWORD);
String sUsername = request.getParameter("USER");
String sPassword = request.getParameter("PASS
WORD");
int iUserID = -1;
String sLoggedInUser = "";
String s = "SELECT User_id, Username FROM
USERS WHERE Username = " +
sUsername + " AND Password = " + sPass
word + """;
Statement selectStatement = connection.create
Statement ();
ResultSet resultSet = selectStatement.execute
Query(s);
if (resultSet.next()) {
iUserID = resultSet.getInt(1);
sLoggedInUser = resultSet.getString(2);
}

PrintWriter writer = response.getWriter ();
if (iUserID >= 0) {
writer.println ("User logged in: " + sLoggedIn
User);
} else {
writer.println ("Access Denied!");
}
%>

```

Prévenir l'injection de code SQL

La solution habituelle pour éviter ce problème est la validation de toutes les données utilisateurs. En général, c'est un moyen efficace de blocage des entrées d'utilisateurs malveillants. Cependant, il est possible également de prévenir ces attaques en créant des expressions, de telle sorte qu'il soit impossible de les corrompre, même avec les données les plus

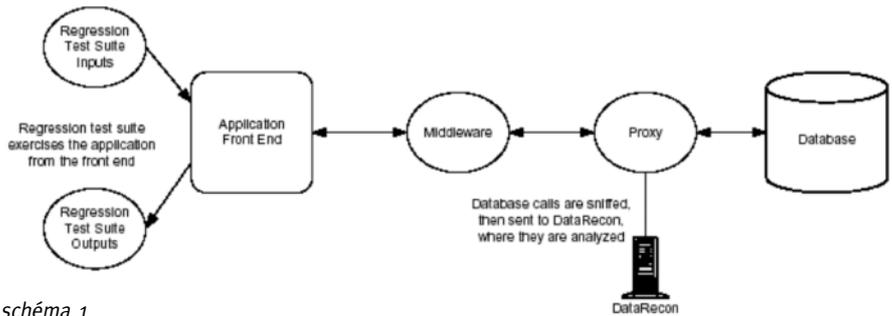


schéma 1

sophistiquées. Un moyen simple de faire barrière aux attaques par injection de code est de s'assurer que toutes les expressions SQL considèrent les données utilisateur comme des variables, et que ces expressions sont pré-compilées avant que les véritables données se substituent aux variables. En général, c'est un processus à deux étapes. La première étape consiste à créer et parser l'expression avec des variables à la place des données utilisateur prévues. La deuxième étape consiste à remplacer les variables par les données utilisateur, avant que l'expression ne soit transmise à la base de données. En appliquant cette stratégie, vous vous assurez qu'aucune donnée utilisateur ne sera jamais parsée à la place de la véritable expression SQL, et les données avec intention malveillante sont donc inefficaces.

En langage Java par exemple, un moyen de créer des expressions SQL sûres est de construire toutes les requêtes avec un PreparedStatement au lieu d'un Statement et / ou d'utiliser des procédures stockées paramétrées.

Les procédures stockées sont compilées avant l'intégration des données utilisateur et rendent impossible la corruption de l'expression SQL. Lorsque PreparedStatement est utilisé, la plupart des pilotes JDBC préparent une expression avec le serveur et donnent les paramètres séparément.

Dans un cas comme dans l'autre, il y a une distinction claire entre l'expression SQL et la variable après la compilation initiale.

Les variables sont encapsulées et des caractères spéciaux qui pourraient s'y trouver sont contournés de façon à convenir à la base de données cible. Par conséquent, il est impossible pour le pirate de faire passer des données malveillantes et faire en sorte qu'elles soient traitées comme si elles étaient l'expression elle-même. Une précaution nécessaire au

cas où le pirate réussirait à introduire du code pour la corrompre.

Même si vous utilisez un PreparedStatement, vous devez quand même être vigilant avec la construction de vos arguments. Tous les paramètres devraient être intégrés par le biais d'un appel de JDBC. Si vous opérez une concaténation de votre chaîne SQL et omettez les appels de JDBC, une tentative d'injection de code risque de fonctionner et le code parsé comme du SQL.

Comment vous assurez-vous que le code utilisé respecte ces pratiques de programmation ? La plupart des expressions SQL sont créées dynamiquement ; vous devez donc exécuter les chemins de l'application qui créent des expressions et vérifier si elles sont construites de façon sûre, c'est - à - dire si les expressions sont bien précompilées avec des variables, avant l'introduction de données utilisateur. D'autre part, il faut inspecter le code relatif à la base de données et vérifier que des pratiques de codage sont bien suivies ; en Java par exemple, les pratiques recommandant l'utilisation - correcte du PreparedStatement au lieu du Statement.

Les solutions AEP pour Java, C/C++ et .NET automatisent ces deux types de vérification qui font partie intégrante des bonnes pratiques de programmation. Les technologies AEP pour Java peuvent par exemple réaliser une analyse statique du code responsable de la création des expressions SQL. Si vous écrivez du SQL pour JDBC, cette analyse statique peut-être utilisée pour vérifier que vous utilisez toujours PreparedStatement. Elle peut aussi permettre de s'assurer que tous les PreparedStatement disponibles contenus dans le code sont construits correctement, avec tous les paramètres intégrés via des appels de JDBC, plutôt que par la concaténation d'une chaîne. De plus, pour déterminer si les expressions SQL sont construites en respectant le

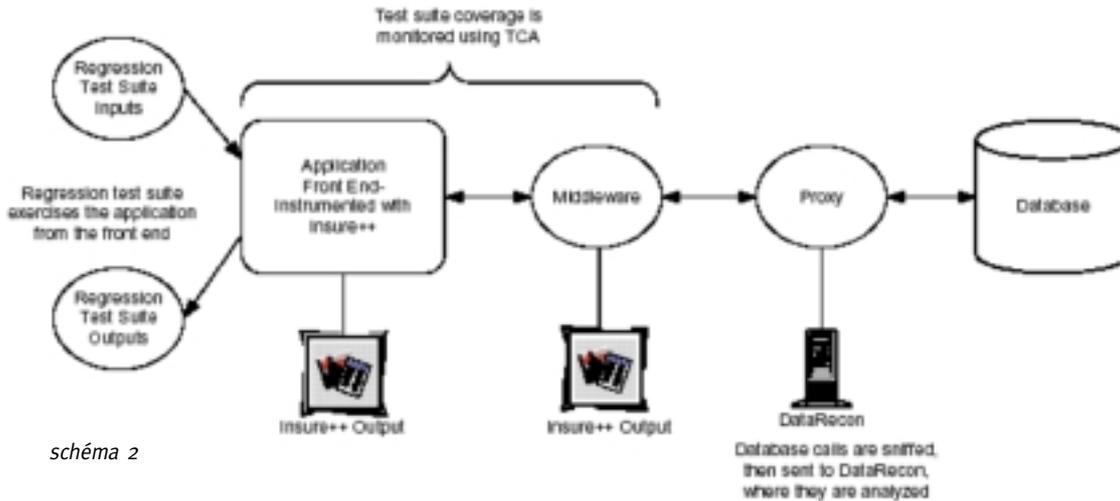


schéma 2

processus à deux étapes recommandé plus haut, les technologies de monitoring AEP utilisent l'analyse dynamique pour observer les appels à la base de données, lors des tests d'intégration réalisés sur l'application. Pendant que l'application est stimulée par une suite de tests fonctionnels exécutée via un client test, cette technologie de monitoring AEP surveille la couverture des tests pour établir quels chemins de l'application sont couverts, et piège toutes les expressions SQL passant par le proxy. Datarecon, un outil AEP pour bases de données de Parasoft, analyse ensuite comment les expressions SQL ont été construites dynamiquement et identifie toutes celles qui ne sont pas sûres. De cette façon, AEP vous permet d'identifier les failles de sécurité et de les corriger avant que les pirates n'aient l'opportunité d'exploiter votre application. Ce processus d'analyse est illustré dans le schéma 1.

Les buffer overflow

Le buffer overflow est un problème qui affecte bien trop souvent les applications en C/C++. Les attaques par buffer overflow ont lieu, lorsqu'un pirate réussit à faire passer des données à travers les défenses du programme et à écrire au buffer. Cela n'est possible que s'il peut trouver et exploiter une corruption de mémoire dans l'application.

Exemple de buffer overflow

Supposons, par exemple, que votre application en C++ a une zone mémoire sur la pile et qu'il soit possible d'écrire plus que ne peut contenir cette zone pour écraser l'adresse de retour de la fonction. Le pirate peut exploiter cette faiblesse de façon à ce que la fonction retour-

ne à une adresse, ou qu'elle exécute une procédure qu'il aura spécifiée. En fait, des buffer overflows découverts récemment dans des systèmes d'exploitation connus permettent ainsi aux pirates d'obtenir une licence pour réaliser différentes attaques, comme faire planter le système, installer des programmes, lire, modifier, et supprimer des données, modifier n'importe quelle partie du système et créer des comptes avec privilèges.

La prévention des buffer overflow

En général, les développeurs essaient d'éviter ce genre de problèmes, en limitant la taille de leurs données ou en vérifiant les données. Cependant, il est difficile de ne pas en oublier si vous n'avez pas de procédure pour identifier toutes les données qui doivent être vérifiées. Pour éliminer ces failles, vous devez prévenir les bogues de corruption mémoire qui sont à l'origine de leur apparition.

Les problèmes de corruption de mémoire peuvent être évités en appliquant les pratiques de test d'intégration AEP standard. Lors des tests d'intégration, l'application est compilée avec Insure++ (outil de détection d'erreurs à exécution) qui détecte les corruptions de mémoire et localise l'endroit où le programme déborde une zone mémoire. En stimulant l'application avec des suites de tests de non-régression, on identifie toute corruption de mémoire. Chaque bloc mémoire est observé et chaque accès à la mémoire est vérifié pour déterminer s'il est légal et s'il risque d'y avoir un débordement de code, autrement dit, un buffer overflow. Les problèmes ainsi identifiés sont rapportés et devraient être corrigés aussitôt que possible.

Si l'application subit la suite de tests en profondeur, la plupart des risques de corruption de mémoire devraient être mis en lumière pendant le processus. Pour vérifier si l'application est bien testée en profondeur, il est possible de monitorer la couverture des tests et identifier ainsi des parties de code non testées, puis ajouter des cas de tests selon les besoins pour une meilleure couverture. Lors de tests avec Insure++

par exemple, l'outil TCA de Parasoft permet d'identifier quels sont les blocs de code qui ont été couverts ou non.

Le processus est illustré par le schéma 2.

Les exceptions à l'exécution non catchées : la faille

Les gens pensent que les langages tels que Java et .NET (C/C++ managés) n'ont pas de problèmes de buffer overflow. C'est vrai, mais ils souffrent d'un autre problème qui n'est pas moins sérieux : les exceptions à l'exécution non catchées. Les exceptions typiques contrôlées donnent un moyen relativement facile de transférer les flux et permettre à un programme de continuer à tourner en cas de situation exceptionnelle. Cependant, les exceptions qui ne sont pas contrôlées (donc des exceptions envoyées automatiquement par le système lorsque le programme transgresse les règles de syntaxe et sémantiques du langage) sont généralement une indication de bogue. Ils proviennent souvent de problèmes relatifs à l'arithmétique, aux pointeurs, à l'indexage et peuvent apparaître dans le programme à n'importe quel moment. Dans certains cas, le transfert de flux inattendu qui en résulte et la fermeture de tâche qui peut avoir lieu, risquent d'amener instabilité, résultats imprévus ou même plantages et trous de sécurité.

La prévention des exceptions

La seule façon efficace de prévenir ces attaques est d'identifier toutes les exceptions non catchées potentielles tôt dans le cycle de vie du développement, de les analyser et de modifier le code pour éviter qu'elles ne fournissent une porte ouverte aux pirates. Les pratiques de test unitaire standard AEP

sont conçues pour identifier ces exceptions à l'exécution, non catchées. Chaque outil de test unitaire AEP (comme Jtest, C++Test ou .TEST) est prévu pour analyser le code automatiquement lors des tests et déterminer comment le tester. Son objectif est de tester chaque unité du code à l'aide d'un large spectre d'entrées acceptables et de vous prévenir des exceptions non catchées potentielles. S'il identifie des entrées qui peuvent causer une exception non catchée, il réalise un rapport avec les entrées et le type d'exception qu'elles ont provoqué. Pour être sûr que les pirates n'exploitent pas d'exceptions non catchées, il faut identifier les exceptions potentielles et modifier le code. On s'assure ainsi qu'aucune exception ne se produira lors de la release, ou du déploiement de l'application. Nous recommandons le traitement de toutes les exceptions détectées par l'outil AEP avant de travailler sur un autre code. La mesure appropriée pour chacune d'entre elles dépend de sa nature. Si, par exemple, une exception non catchée a été rapportée concernant une méthode Java qui se comporte mal (cette méthode n'envoie pas d'exception pour les arguments donnés), vous devez modifier le code de la méthode pour qu'elle se comporte normalement. L'avantage de cette manipulation est que vous prévenez ainsi une instabilité potentielle, des résultats incorrects, des problèmes de sécurité, et / ou des plantages. C'est le cas décrit dans l'exemple précédent. Si l'exception non catchée est rapportée au sujet d'une méthode Java qui ne recevra jamais d'arguments provoquant cette exception et / ou qui n'est pas prévue pour traiter ces arguments, il faudra ou bien modifier le code de façon à ce qu'il renvoie une `IllegalArgumentException`, ou introduire un commentaire Design by Contract qui spécifie les arguments valides. Dans les deux cas, vous prévenez l'exception et la faille de sécurité qui en résulte. Si une exception non catchée est rapportée pour une méthode Java prévue pour envoyer une exception non catchée, vous devrez vérifier que cette exception ne peut pas être exploitée par des personnes mal intentionnées, puis documenter la méthode pour clarifier le fait que l'exception est voulue et qu'elle a été vérifiée pour ne pas provoquer de trou de sécurité.

Enfin, une dernière mesure de prévention est recommandée pour les méthodes critiques, comme la méthode du dernier exemple `isUserNamePasswordCorrect()`; on devrait toujours ajouter un `try/catch` au code du client pour traiter les exceptions éventuelles.

Conclusion

Les pratiques de prévention sont souvent mésestimées et presque toutes sont remises en question ou ignorées au moins par certaines personnes. Demandez par exemple à un fumeur son opinion au sujet de la prévention du cancer des poumons ou des maladies du cœur... Pourtant, lorsqu'elles sont utilisées en complément des stratégies de détection, les pratiques de prévention permettent de réduire considérablement les risques de problèmes que vous souhaitez justement éviter. Elles vous font économiser du temps, du travail et des ressources en prévenant quelque chose qui est habituellement difficile à diagnostiquer et éliminer.

■ Adam Kolawa

Président et cofondateur de Parasoft. Docteur en Sciences Physiques, il est co-auteur de *Bullet proofing Web Applications (Hungry Minds 2001)*. ak@parasoft.com

La librairie des développeurs



O'REILLY®



Microsoft
Press

EYROLLES



CampusPress

a!
Apress®



ADDISON
WESLEY



L'offre la plus complète
d'ouvrages informatique
+ de **45 000 références**,
en français et anglais !

Comment commander

- Sur www.infotheque.com
- Par courrier : Infothèque
81, rue d'Amsterdam – 75008 Paris
- Par fax : 01 42 82 18 18

Contactez-nous

- 01 42 82 17 17
- infos@infotheque.com

Visitez-nous

- 81, rue d'Amsterdam – 75008 Paris
M^o Place de Clichy lignes 2 ou 13 – Bus 81 et 95

www.infotheque.com

La sécurité du code...



Toute erreur d'expression avec un langage est donc une erreur humaine qui se répercute sur la plate-forme cible. Les langages de programmation sont indispensables, car ils apportent des facilités de développement et des paradigmes propres à résoudre des problèmes spécifiques. La maîtrise de certaines de leurs subtilités est donc indispensable. Ici, nous séparerons volontairement les langages en deux grandes familles : ceux qui ciblent une plate-forme native (C/C++, VB6...) et ceux qui ciblent une plate-forme virtuelle (Java, C#, VB.NET, PHP...). On pourra classer dans les plates-formes natives Linux ou Windows et comme environnement virtuel, la JVM ou le CLR.

Plate-forme native

Les plates-formes natives (Linux, Windows,...) exécutent des applications sans aucune vérification, car le code compilé est directement exécuté par le processeur. Toutes les possibilités sont donc offertes, afin de profiter au mieux de la plateforme en termes de performance et d'accès mémoire. Mais la performance à un prix : celui de la stabilité.

C/C++

Les langages qui ciblent une plate-forme native telle que Linux ou Windows sont plus difficiles à maîtriser. Pourquoi ? Tout simplement parce que l'OS ne fait pratiquement aucune vérification sur la validité des instructions émises par le compilateur. Un compilateur peut ainsi allègrement donner le droit de générer du code permettant de lire un espace mémoire non autorisé. Les langages les plus utilisés autorisant ce type de manipulation mémoire sont le C et le C++.

Le langage C est particulièrement sujet aux problèmes de sécurité, car il n'a pas été conçu comme un langage pour le commun des mortels. En effet, la destination de ce langage est clairement orientée pour le développement de couche très bas niveau (OS, drivers) et certainement pas pour le développement des applications courantes.

Quant au C++, même si celui-ci apporte un

concept objet, il n'en est pas moins compatible sémantiquement avec son ancêtre...

Cast

Par " cast ", on entend le fait de pouvoir convertir une valeur en une autre (généralement des pointeurs).

Par exemple, il est possible en C/C++ d'écrire ceci pour lire le contenu de l'adresse mémoire 0x666 :

```
int i = *(int *)0x666;
```

Cela est non seulement parfaitement autorisé, mais également très dangereux. Le problème est qu'il est pratiquement impossible de s'en passer en C lorsque l'on alloue des données sur le tas : il faut " caster " les différentes structures vers le bon type de données. Il y a double risque : erreur de cast et erreur de taille d'allocation.

```
MyStruct* ms = (MyStruct*)malloc(sizeof(MyStruct));
```

Qui dit allocation mémoire dit libération de mémoire : en C cela se fait en utilisant la fonction free() de la CRT. Puisqu'il est possible avec ces langages de caster un pointeur, cela veut dire que la libération risque de mal se passer si l'adresse mémoire à libérer n'est pas celle retournée par malloc() ; ce qui arrive fréquemment lorsque l'arithmétique de pointeur est utilisée !

Le langage C++ corrige ces défauts en utilisant la syntaxe new <type> mais il reste toutefois possible de caster vers un autre type de pointeur. Afin de sécuriser au mieux la manipulation des données, il est fortement conseillé de s'interdire les casts de type C et d'utiliser les nouvelles primitives de cast du C++ :

- dynamic_cast : le compilateur générera des informations supplémentaires (RunTime Type Information : RTTI) afin de vérifier la validité du cast. Il faut noter que cela alourdit la consommation mémoire et nuit également aux performances des applications.
- static_cast : le compilateur n'autorisera le cast que si celui-ci est valide (structurellement parlant).
- reinterpret_cast et const_cast : à n'utiliser

qu'en cas d'absolue nécessité ! Ces opérateurs de conversions supposent en effet que le développeur connaît très exactement les données qu'il manipule.

Au bout du pointeur

Le C et le C++ ne distinguent pas un pointeur sur un tableau ou un pointeur sur une instance :

```
// allocation d'un tableau d'instances
Toto* arrayToto = new Toto[10];
// appel des destructeurs des instances
// et destruction du tableau
delete [] arrayToto;

// allocation d'une instance
Toto* toto = new Toto();
// destruction de cette instance uniquement
delete toto;
```

Il est donc particulièrement important de savoir ce qui doit être détruit, sous peine de dysfonctionnement grave des applications (allant de la corruption du tas mémoire à des fuites mémoire).

Destructeur virtuel

Les instances de classes C++ ont la particularité de subir une étape dite de " destruction " pendant laquelle l'instance est notifiée de sa disparition. Dans les cas d'héritage, il est souhaitable que les destructeurs soient virtuels, afin que ceux-ci soient appelés au moment de la destruction (scénarios mettant en jeu du polymorphisme) :

```
class Toto {
public:
    Toto() { ... }
    virtual ~Toto() { ... }
};

class Titi : public Toto {
public:
    Titi() { ... }
    virtual ~Titi() { ... }
};

Toto* t = new Titi();
delete t;
```

Le simple fait de ne pas appeler un destructeur peut provoquer des fuites mémoires et engendrer de gros problèmes de stabilité des applications.

Buffer overflow

Quand on parle de sécurité de code, il faut évoquer en plus des problèmes de fonctionnement, les problèmes de sécurité.

En effet, une attaque courante des applications repose sur le principe du débordement mémoire (buffer overflow). Le C et le C++ déclarent les variables sur la pile et donc, avec un peu d'astuce, il est possible de corrompre le flot d'exécution très facilement :

```
void helloWorld()
{
    printf("hello world !!\n");
    exit(0);
}

void bufferOverflow()
{
    int copyBuffer[8];
    copyBuffer[9] = (int)&helloWorld;
}
```

Le code malicieux de la fonction `bufferOverflow()` profite du fait qu'il connaît la structure de la pile pour détourner le pointeur d'instruction (l'adresse de retour est située à `[EBP+4]`).

Vous me direz qu'il n'est pas tellement possible d'écrire ce type de code, à moins de réellement le vouloir ; mais indirectement, des fonctions comme `strcpy()` le font bien plus souvent que l'on ne le croit : la copie se fait en fonction de la chaîne à dupliquer et non en fonction de la variable cible pouvant, par là même, faire ce que `bufferOverflow()` réalise explicitement. Dans ce cas, il faut utiliser `strncpy()` qui permet de copier de façon sécurisée (au prix d'une perte de performance il est vrai) afin de contrôler les copies mémoire. Lorsqu'une application C/C++ doit accepter des données en provenance d'un tiers, il est forte-

.NET & Win32

.NET est un environnement très sécurisé. Toutefois, pour des raisons de performances ou tout simplement de limitation de la plate-forme .NET, il peut être intéressant d'appeler nativement une fonction de la plate-forme Win32 (en utilisant `DllImport`). Mais attention ! Pour utiliser ce mécanisme, il faut déclarer sans erreur la signature de la fonction à appeler :

```
[DllImport("kernel32.dll")]
static extern
    bool WriteConsole(IntPtr hConsoleOutput,
        string lpBuffer,
        uint nNumberOfCharsToWrite,
        out uint lpNumberOfCharsWritten,
        IntPtr lpReserved);
```

Toute erreur de signature pourra provoquer des erreurs irréversibles dans un processus. Il faut toutefois noter qu'une application n'aura les droits de faire cela, que si elle en possède le droit !

ment suggéré de vérifier le contenu des données avant de les traiter !

Plate-forme virtualisée

Les langages ciblant des plates-formes virtualisées, génèrent du code vérifiable, c'est-à-dire qu'il est possible de s'assurer mathématiquement qu'une application ne pourra faire de mauvaise opération en mémoire, évitant ainsi de la mettre en danger.

Les techniques qu'utilisent ces plates-formes, vont de l'implémentation d'un ramasse-miettes (garbage collector), en passant par l'interdiction de l'arithmétique de pointeur. Par ailleurs, elles forcent un accès mémoire par indirection et contrôlent tous les casts effectués par une application.

En supposant que ce type de plateforme (JVM ou CLR) soit fiable, l'utilisation d'un langage de type Java, C# ou VB.NET ne comportera plus les mêmes risques qu'auparavant.

Les problèmes de sécurité sont maintenant d'ordre applicatif, architectural ou de déploiement. Parmi les problèmes applicatifs, on pourra citer par exemple, comme attaque, l'utilisation d'injection de code SQL si l'application construit dynamiquement des requêtes SQL. La solution est ici, bien entendu, d'utiliser des

procédures stockées. Concernant les problèmes d'architecture, la saturation d'un serveur est le type d'attaque le plus fréquemment rencontré. Lorsqu'une application est mise à disposition sur l'Internet, par exemple, il est nécessaire de vérifier son comportement sous très forte charge. Si une application crée un nouveau thread pour chaque nouvelle requête, les dénis de service (DoS) seront très certainement possibles.

Restent les problèmes de déploiement, inhérents aux droits affectés à la machine virtuelle : droit d'écriture, privilèges d'exécution etc... Généralement, ces problèmes sont couplés avec des bugs applicatifs.

Conclusion

Les anciennes plates-formes de développement apportent une facilité d'expression et une performance qu'il n'est plus possible d'avoir avec les langages ciblant des environnements virtuels. Mais soyons réalistes : l'augmentation dramatique de productivité et la robustesse apportée par ces derniers en font des outils incontournables de nos jours.

■ Pierre Chalamet

Expert .NET chez Neoxia
pierre.chalamet@neoxia.com



Etes-vous satisfait du magazine ?

www.programmez.com

MONO 1.0 : *compatible .Net en Open Source*

La stratégie de Novell

À l'heure, où ces lignes sont écrites, Mono est encore en bêta 2, la version finale étant prévue le 30 juin. Avoir un outil, c'est bien, mais savoir à quoi il sert et pourquoi l'utiliser, voilà deux réflexions qui méritaient bien quelques commentaires.

Aujourd'hui, Mono représente 20 développeurs Novell mobilisés en interne et une communauté de plus de 150 développeurs dont une cinquantaine actifs. C'est l'équivalent d'un gros projet Open Source. Depuis déjà plusieurs mois, Mono est même utilisé pour divers projets au sein de Novell, l'un d'entre eux est iFolder 3.0. Comment définir la cible de Mono ?

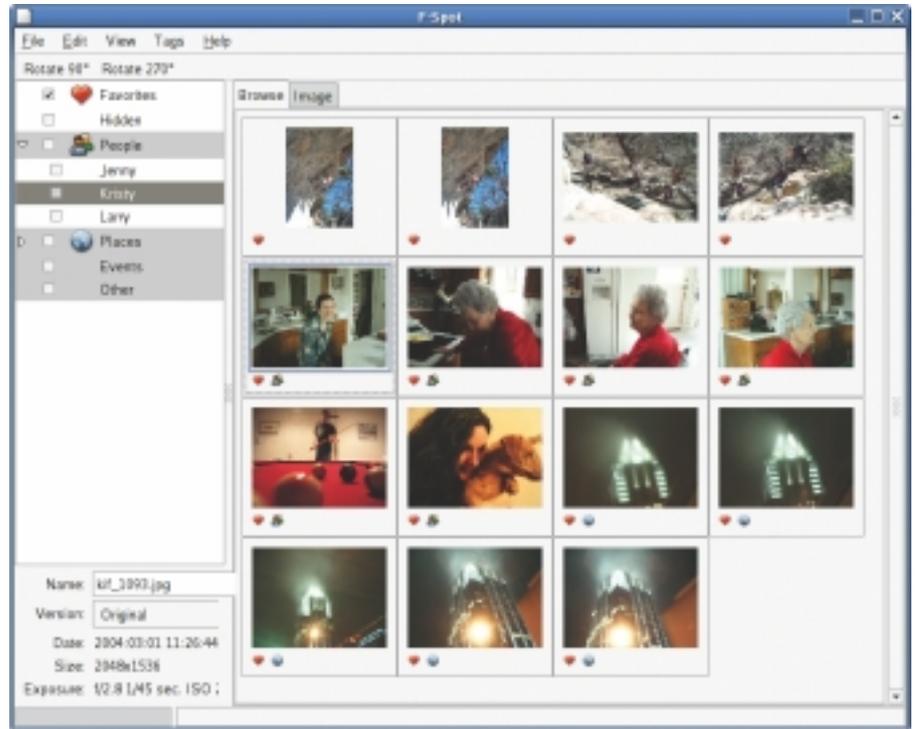


Éric Dasque

Éric Dasque (Mono Senior Project Manager - Novell) apporte un début de réponse : " il manquait à Linux une plate-forme de développement. On a regardé ce qu'il y avait sur le marché et on a trouvé que .NET était le plus approprié. Le fait que Mono soit compatible avec la plate-forme de Microsoft est une bonne chose, mais ce n'était pas notre but. C'est la conséquence de notre choix. Mono est avant tout une plate-forme de développement pour Linux et Unix. ". Deux avantages à Mono peuvent être donnés : multi-plate-forme, gratuité. Il y a encore quelques mois, on ne savait pas si Mono garderait son aspect gratuit ; avec l'arrivée de la version 1.0. On peut être rassuré, Mono restera librement téléchargeable. En complément, Novell souhaite apporter un " plus produit ". Cela passera par des packages de services et de supports, en plus de l'outil en lui-même. Il y aura aussi la possibilité de voir apparaître des licences commerciales pour éviter aux entreprises et développeurs d'être inquiets sur les licences libres.

Mono, pour quoi faire ?

Tout d'abord, il est possible de construire des applications écrites en C# fonctionnant sur MacOS X, Linux, Unix et Windows. On peut



bien entendu créer de nouvelles applications avec, mais clairement, pour Novell, ce n'est pas le premier objectif, ni le côté multi-plate-forme (Mono est une plate-forme pour Linux avant tout).

Éric Dasque fournit quelques pistes :

- Ma société migre de Windows à Linux, il faut porter les applications de gestion et autres, développées sous Windows,
- Je suis éditeur ou développeur vertical, je passe à un moment ou à un autre à Linux
- J'ai besoin d'une application Windows et Linux
- Je migre mon serveur de Windows à Linux, j'ai des applications IIS / ASP.NET, comment je migre ASP.NET sous Linux ?

Car si Mono permet de faire fonctionner les binaires .NET de Visual Studio .NET sous Linux, il est aussi compatible avec ASP.NET, ce qui est largement mis en avant. Car, jusqu'à présent, faire de l'ASP.NET sous Linux était impossible ! Sur les relations Microsoft - Mono, finalement, ce fut une bonne entente. Car, d'une certaine manière, Mono conforte l'ouverture de

Microsoft vers l'Open Source et renforce ainsi C# face à Java et même ASP.NET face à PHP. Mono en lui-même n'a pas de module RAD. Novell souhaite rester compatible avec les outils des développeurs. Si on développe avec VS.NET, il suffit (en théorie) de reprendre le binaire et de le copier sous Linux. Si on développe avec Eclipse, Novell travaille avec la société française Improve pour mettre au point un plug-in dans cet IDE. Cependant, comme Novell prend conscience de l'intérêt d'avoir une suite de développement complète, il fallait aussi un outil RAD. C'est pour cela que l'éditeur a pris l'outil Open Source SharpDevelop que Novell a largement repris et modifié (hormis le cœur de l'outil). Cet outil porte le nom de MonoDevelop. Il sera disponible sous Linux et MacOS X. Il est intégré depuis la bêta 2 à Mono, et sera disponible gratuitement. Novell intégrera rapidement Mono dans les distributions Linux SuSe. Pour les autres distributions, aucune information précise sur la question.

■ François Tonic

Mono 1.0 : une release majeure pour coder des applications réelles ?

A l'annonce du rachat de Ximian par Novell, les opinions les plus divergentes ont animé les forums de discussions consacrés à la programmation. Pour certains, le projet Mono était condamné. La sortie de la version 1.0 a étouffé les critiques dans l'œuf. Assurément Mono n'a pas cessé de nous étonner.

Utilisable sous GNU/Linux, BSD, Solaris, MacOS X et Windows, ce projet ne représente pas seulement une "alternative à java" : c'est d'abord et avant tout une implémentation complète des spécifications de l'ECMA. Ensuite, il s'agit d'un jeu d'API spécifique à Mono et destiné à la programmation sous Linux (comme GTK#), et enfin nous avons à notre disposition une solution visant la compatibilité avec le framework .NET.

Des applications réelles

Nous demandons aux sceptiques sous Linux d'installer sans tarder l'environnement de développement MonoDevelop. Nous avons été séduits par la possibilité fort utile d'auto complétion du code. Cet outil représente un excellent exemple de ce que l'on peut réellement coder avec Mono. Gnunit, une interface graphique pour les tests unitaires, F-spot, une visionneuse de photos, et Muine, un lecteur de morceaux musicaux, sont également trois applications réelles programmées sous mono.

Face aux nouveaux défis

Les défis pour Mono sont grands, comme l'a expliqué Miguel de Icaza, l'initiateur du projet, après avoir assisté fin 2003, à la dernière conférence développeur de Microsoft à laquelle il avait été convié. En effet Microsoft a annoncé une série de nouvelles technologies. Avalon, le toolkit GUI pour .Net ; Indigo, un framework RPC ; winFS, successeur de NTFS, un nouveau système de fichiers, dont la charpente est une base de données ; whidbey, une autre version de Visual Studio, comprenant une évolution du framework .Net 1.2 ; et enfin winfx, un jeu d'API pour Longhorn (le nouveau système d'exploitation Windows prévu pour 2006-2007).

Sans oublier, le prometteur langage de balise xaml. Celui-ci permet d'élaborer des applications GUI à partir d'un langage de style XML.



Par exemple le code xaml suivant...

```
<Window>
  <Button id="button" Background="Red">
    <SimpleText>Hello</SimpleText>
  </Button>
</Window>
```

... donnera après compilation quelque chose de ce style :

```
Window w1 = new Window ();
Button button = new Button ();
SimpleText s1 = new SimpleText ("Hello");
button.Items.Add (s1);
w1.Items.Add (button);
```

Le roadmap annoncé

En janvier 2004, après avoir digéré ces informations, Miguel a proposé un nouveau plan de route aux développeurs. Nous en sommes à la première phase, c'est-à-dire Mono 1.0. Mono 1.2 devrait suivre fin 2004 début 2005, et comprendra le support des types génériques pour le compilateur C#. Le compilateur pour VB.NET sera complet et le support Jscript terminé.

Mono 1.4 courant 2005, et enfin Mono 2.0 en 2006, année ou devrait sortir Longhorn (cette

version de Mono devra atteindre la compatibilité .Net 1.2 de manière stable). Mono 2.0 tentera de se rendre compatible avec le nouveau jeu d'API WinFX. C'est-à-dire se rendre compatible avec Winfs, Avalon et Indigo, ce qui ne sera pas une mince affaire...

Sous le capot de la version 1.0

Où en sommes-nous avec Mono 1.0 ? Pour répondre à cette question, il faut

distinguer trois axes : le premier est le cœur de Mono, avec en ligne de mire la compatibilité .Net 1.1 ; le deuxième est constitué d'exclusivités propres à Mono (ce que l'on appelle la "pile mono"), c'est-à-dire des extensions, ou outils qui n'existent pas dans le monde Microsoft ; le troisième comprend les futures améliorations, comme le support des types génériques (en "preview" via la commande gmcs). Reste le gros morceau des toolkits GUI dont nous discuterons plus loin.

Le cœur

Mono 1.0 propose un environnement d'exécution compatible avec la norme ECMA-CLI. Il faut aussi noter qu'il supporte maintenant les plates-formes StrongARM et HPPA. Son compilateur C# 1.0 (mcs) est fonctionnel. Le runtime a été amélioré : la gestion du Pool de Threads est plus proche de ce que l'on retrouve sous Windows (offrant une compatibilité supérieure). De nombreux bogues ont été colmatés.

Il est tout à fait envisageable d'utiliser Mono en entreprise. Sont opérationnels : ASP.NET (Web Forms et Web Services), ADO.NET (connectivité avec les bases de données), et

l'exécution distribuée (Java/RMI ou SOAP). Au sujet de la portabilité : si l'application .Net est pure (aucun appel à Win32) la portabilité est envisageable ; le gros hic se situera au niveau du GUI en cas d'application graphique.

L'offre "services Web" comprend un compilateur WSDL, des appels asynchrones, des extensions SOAP, et enfin le serveur XSP (basé sur le moteur ASP.NET de mono) qui peut s'utiliser, via Apache (en module).

XSLT précompile maintenant les pages, et certaines classes manipulant le XML ont été améliorées, comme la classe XMLTextReader. Le parser DTD est deux fois plus rapide (merci Atsushi Enomoto) tout en occupant deux fois moins de mémoire...

La pile Mono

1• Java avec IKVM.NET

Mono 1.0 comprend la machine virtuelle java IKVM.NET. L'implémentation de cette JVM poursuit un double but : d'une part de pouvoir exécuter dynamiquement n'importe quelle application java avec Mono, et d'autre part de pouvoir compiler statiquement une bibliothèque java au sein d'un assemblage .Net.

Concrètement, la commande "ikvm.exe" est l'exécutable comparable à "java.exe", qui se charge de l'exécution en mode "dynamique". Et la commande "ikvmc.exe" est le compilateur statique qui se charge de compiler une classe Java en un assemblage .Net. Ikvm.net se destine au développeur java qui désire migrer du code java vers .Net, sans passer par un outil propriétaire comme J#. En terme de compatibilité, le but est ici de coller le plus possible à l'implémentation du JDK 1.4 de Sun.

2• Compilation Just-in-Time et Ahead-of-Time

Deux développeurs de l'équipe internationale, Dietmar et Paolo, se sont concentrés sur le compilateur JIT et le nouveau précompilateur. Il a été possible de simplifier la portabilité du moteur JIT et aussi d'implanter de nombreuses techniques d'optimisation.

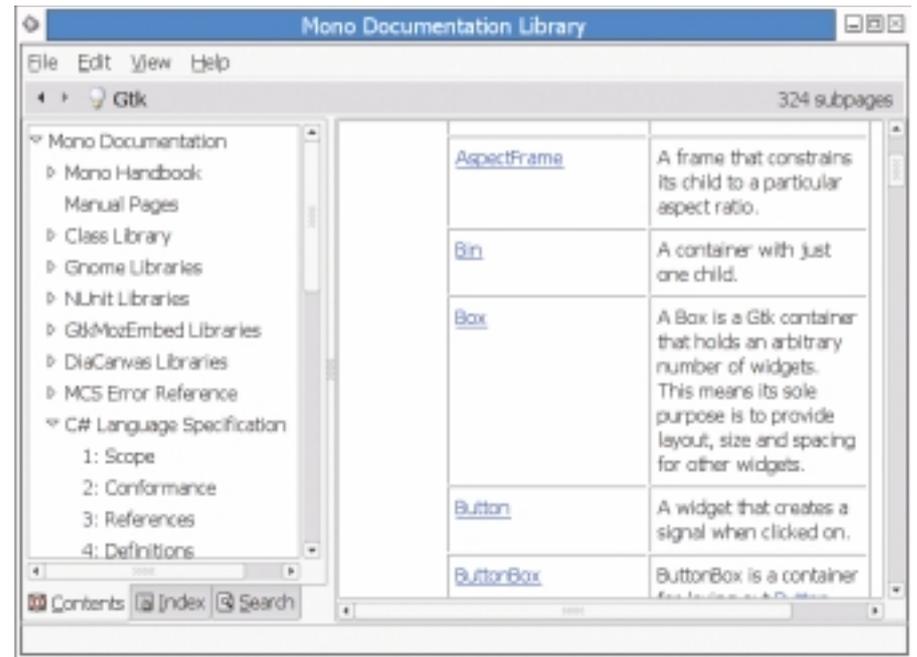
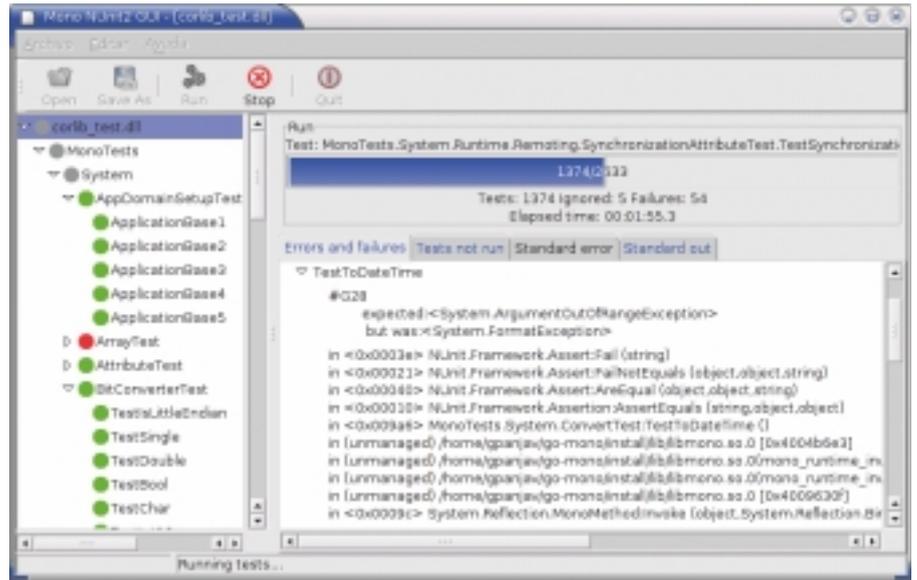
Concrètement, en mode JIT vous pouvez lancer la commande classique :

```
mono unTest.exe
```

En mode de précompilation, vous pouvez précompiler l'assemblage, en indiquant le drapeau aot (Ahead-of-Time) comme ceci :

```
mono -aot unTest.exe
```

Ce qui générera le fichier unTest.exe.so avec



du code précompilé. Pour exécuter celui-ci, il suffit alors de taper :

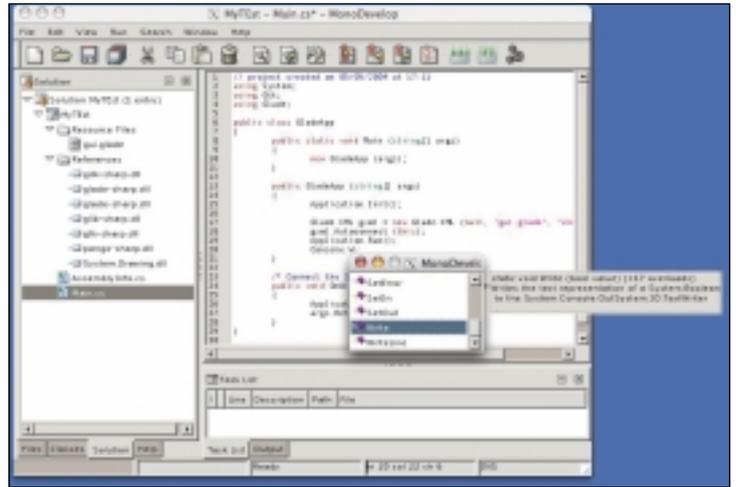
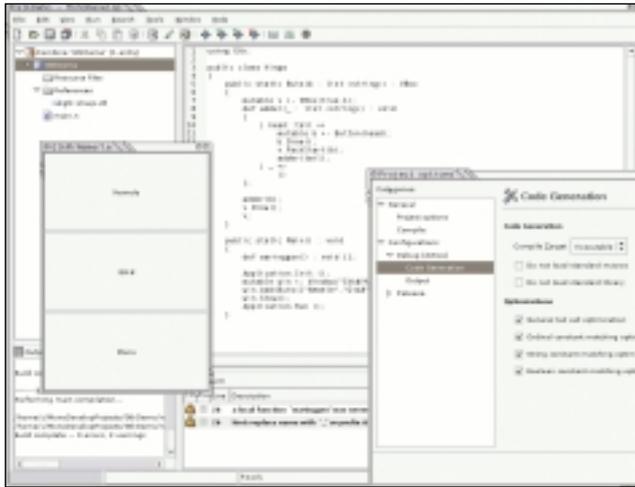
```
mono unTest.exe
```

Le runtime ira automatiquement prendre le binaire précompilé, à condition qu'il se situe dans le même répertoire.

De nombreuses optimisations ont été implémentées, dont voici la liste (tapez mono sans argument pour afficher cette liste).

```
-optimize=OPT Turns on a specific optimization:
peephole Peephole postpass
branch Branch optimizations
inline Inline method calls
cfold Constant folding
```

```
consprop Constant propagation
copyprop Copy propagation
deadcode Dead code elimination
linears Linear scan global reg allocation
cmov Conditional moves
shared Emit per-domain code
sched Instruction scheduling
intrinsic Intrinsic method implementations
tailc Tail recursion and tail calls
loop Loop related optimizations
fcmov Fast x86 FP compares
leaf Leaf procedures optimizations
aot Usage of Ahead Of Time compiled code
precomp Precompile all methods before
executing Main
```



En résumé, c'est à vous à trouver le juste compromis entre la qualité du code généré et la vitesse d'exécution. En mode JIT toutes les optimisations ne sont pas accessibles, mais un bon argument général est de toujours indiquer -O (pour toutes).

3• Monodoc

MonoDoc est une exclusivité Mono fort intéressante. Il s'agit d'un client browser GUI de la documentation Mono, dont l'interface a été conçue en GTK#. Monodoc propose également de mettre en place une interface de consultation basée web, via une implémentation asp.Net ! Bref, monodoc est aussi une belle illustration de ce qu'il est possible de coder réellement avec mono.

4• Sécurité Mono spécifique

Sur le plan de la sécurité, les classes de "Policies" sont apparues. Mais une autre exclusivité est la partie cryptographique propre à Mono. Cet espace de nom Mono.Security fournit les pièces manquantes au framework .Net de Microsoft. Sous Windows, c'est souvent CryptoAPI qui est employé pour signer du code, livrer un certificat X.509, etc. Mono implémente tout ceci en code 100% managé, pour une question d'indépendance vis-à-vis de la plate-forme, Chapeau !

Par exemple, l'espace de nom Mono.Security.Authenticode implémente les fichiers SPC (Software Publisher Certificate) et les fichiers de clé PVK (Private Key). Sous

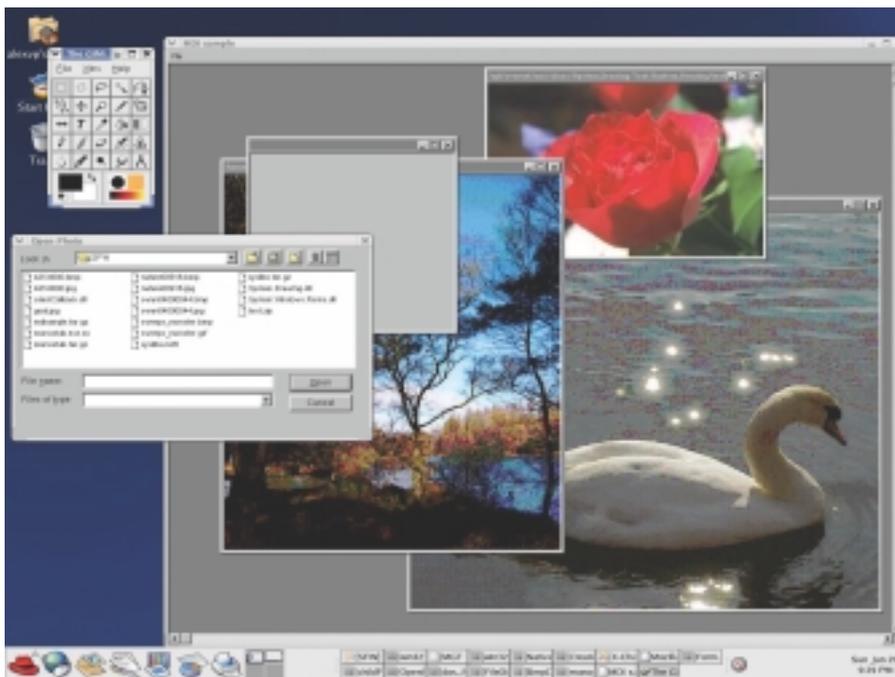
Mono.Security.Cryptography vous retrouvez les algorithmes MD2, MD4, et ARCFour nécessaires pour SSL (Mono.Security.Protocol.* dont une implémentation complète SSLv3). Idem concernant les certificats X.509 (Mono.Security.X509). Un module Mono.Security.Win32 vient compléter ces possibilités, dans le but d'offrir une compatibilité CryptoAPI pour une application tournant sous Windows et sous Linux.

5• Ponts vers des bases de données et autres fonctionnalités

Dans la catégorie propre à Mono, l'on retrouve aussi des fournisseurs de données pour les bases de données Postgress, MySQL, DB2, Sysbase, Sqlite et Oracle. MySQL peut très facilement remplacer SQL SERVER, sans pour autant nécessiter de réencoder quoi que ce soit. Enfin, citons brièvement l'intégration LDAP et également Cairo, deux fonctionnalités exclusives de la "pile mono".

Le monde des GUI

Il existe à l'heure actuelle plusieurs possibilités de développer une application graphique à l'aide de Mono (et pour chacune d'elles, il y a des avantages et des inconvénients). Où se situe le problème ? L'espace de nom "System.Windows.forms" fait partie de la distribution standard de Mono. Mais dès le départ les développeurs ont pris conscience qu'atteindre la compatibilité avec le système de Microsoft serait difficile. En effet, certaines fonctionnalités, comme les handles, les messages, wndproc, etc., ne sont atteintes que par appel d'API win32 (au travers du mécanisme P/Invoke). Néanmoins, trois solutions existent actuellement, si vous désirez utiliser Windows.forms dans votre application.



Windows.forms ?

Soit vous utilisez l'implémentation GTK#, c'est-à-dire que chaque méthode de Windows.forms est "redirigée" vers le sous-système GTK#, qui lui-même fait appel à GTK+. L'aspect visuel, "le rendu", est excellent, que ce soit sous Linux, Windows ou Mac OS X, mais par contre, la compatibilité est pauvre. Pas question ici de "migrer" une application complexe, ni d'utiliser un composant tierce.

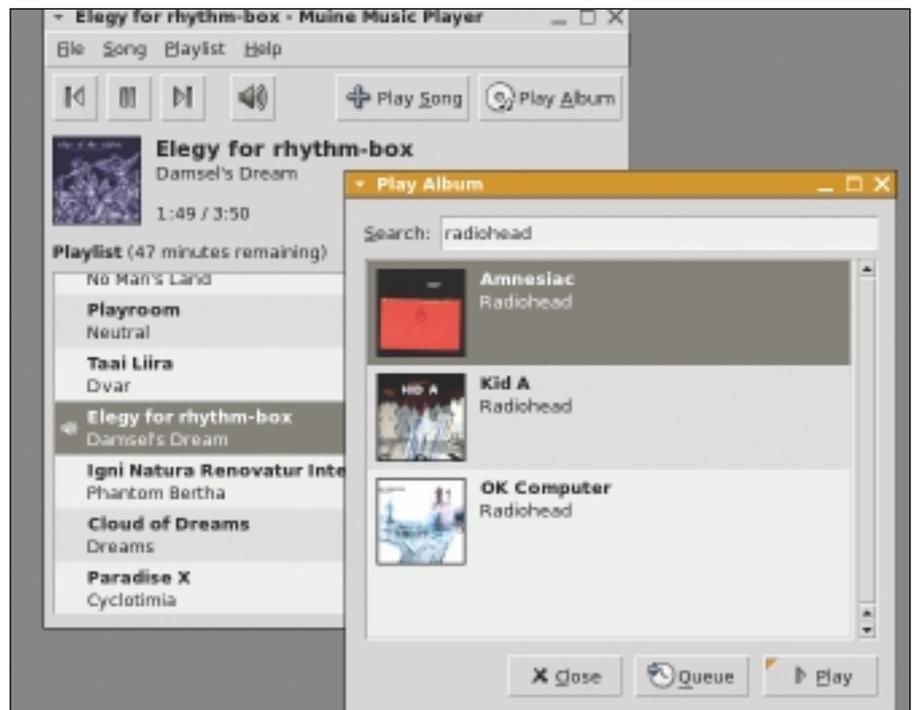
Soit vous adoptez l'implémentation Windows.forms du projet portable.net. Celui-ci fait appel à une bibliothèque depuis la version 0.5.8 pour implémenter Windows.forms en pur C#. Il s'agit d'une approche similaire à Java Swing et qui fonctionne avec des processeurs x86, PPC et ARM sous Linux, MacOS X ou Windows. L'avantage premier de cette bibliothèque est d'être utilisable sous Mono (licence GPL). Deuxièmement, il y a très peu de dépendance. Par contre, la compatibilité est pauvre, et le "look" obtenu au final n'est pas fabuleux. Soit vous approuvez la solution native de Mono, qui consiste à passer par l'émulation. De ce fait, lorsqu'un problème de compatibilité est rencontré, l'appel api win32 est invoqué, via Wine (en passant par des bibliothèques partagées). De ce fait, la compatibilité est bonne et un projet comme Webmatrix pourra à terme s'exécuter sous Linux. Le "look" obtenu n'est pas extraordinaire sous Linux mais les développeurs y travaillent en "patchant" Wine pour pouvoir mieux "coller" au thème sous-jacent.

Ou un Toolkit ?

Si vous ne partez pas d'un existant, c'est-à-dire, si vous développez une nouvelle application de A à Z, à notre avis, la meilleure solution actuellement, n'est pas de passer par Windows.forms, mais un toolkit graphique portable. Trois boîtes à outils existent pour remplir ce rôle : GTK#, Sharp WT et QT#.

QT# lie QT à Mono. Le rendu est évidemment excellent, la documentation ne manque pas et QT# est distribué selon les termes de la licence GPL. Sharp WT est un nouveau projet et de ce fait, la documentation est plutôt lacunaire. L'idée est de partir du toolkit SWT (Eclipse) pour le porter de java en C#. L'affichage sera plus ou moins réussi, selon le système d'exploitation

() Il y a bien Rotor, une implémentation "shared source" du CLI, signée Microsoft, mais évidemment les parties réellement "intéressantes" n'y sont pas (comme Windows.forms, ASP.NET ou ADO.NET). En outre, selon les propres aveux de Microsoft, Rotor n'est vraiment utilisable "qu'à des fins éducatives" (en réalité, surtout dans le but de passer la certification ECMA). Enfin, Rotor est implémenté sous BSD et MacOSX, mais non Linux.*



sous-jacent, et il est difficile de créer un nouveau contrôle, en raison de ce rendu multi plate-forme. GTK# est le projet le plus mature sous Mono. Son développement est fort actif, et la plupart des applications réelles actuelles se basent dessus (comme monodoc par exemple). Le look est excellent, GTK# s'intègre parfaitement à Linux (Gnome) ou à Windows (un port win32 est disponible). GTK# est stable, supporte l'internationalisation, et unicode. Il représente le toolkit idéal pour les (nombreux) développeurs Linux ayant déjà des notions de GTK+.

Conclusion

Si vous devez développer une application ASP.NET ayez le réflexe Mono. Quoiqu'il arrive vous ne le regretterez pas. Microsoft n'offre pas cette possibilité de développement .Net, libre de bout en bout (*).

Si maintenant vous réfléchissez au moyen de mettre en œuvre une application GUI portable entre Windows, Linux et Mac OS X, vous devez au préalable effectuer un choix : utiliser Windows.forms ou non. Tout dépend de votre machine de développement principale, des OS cibles, du code existant, de votre expérience précédemment acquise avec un toolkit. Pour

Première impression

Pour

- multi plate-forme, dont Mac OS X ;
- ASP.NET fonctionnel en entreprise avec un serveur embarqué XSP qui peut s'utiliser comme un module, via Apache ;
- Support d'une machine virtuelle Java ;
- API de sécurité portable ;
- Fournisseurs de données pour des bases de données libres comme MySQL ;
- Multiples optimisations implémentées ;
- Excellents toolkits graphiques portables stables : GTK# et QT#.

Contre

- Absence de compatibilité au niveau des Windows.forms ;
- Code existant, difficilement portable tel quel.

l'instant, notre préférence ira à GTK#. Evidemment, si vous considérez que le look final de votre application à moins d'importance que les fonctionnalités qu'elle contient, vous avez intérêt à vous tourner vers un portage natif des widgets en C#, ce qui entraînera un minimum de dépendances (ce qui importe, lors du déploiement).

Au risque de nous répéter, nous conseillons à tous les sceptiques de commencer par essayer MonoDevelop. Bonne programmation !

Mono : <http://www.go-mono.com/>

■ Xavier Leclercq

xavier.leclercq@programmez.com

Introduction au mapping OR avec ObjectSpaces

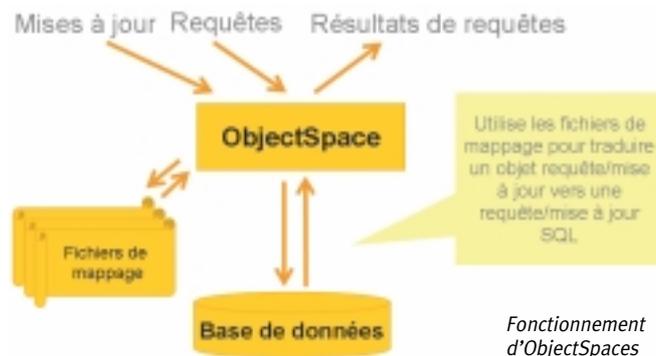
Il existe actuellement sur le marché plusieurs solutions de mapping Objet/Relationnel. Nous consacrons cet article à celle proposée par Microsoft sur la plate-forme .Net, portant le nom d'ObjectSpaces. ObjectSpaces est une nouvelle fonctionnalité qui sera diffusée après la sortie de Visual Studio 2005. ObjectSpaces est une couche abstraite qui se positionne entre la logique applicative et une source de données. Elle permet au développeur d'accéder et de manipuler des données en tant qu'objets, sans connaître la structure de la base de données correspondante. La récupération des objets d'une source de données et la persistance de ces objets est assurée sans avoir à écrire une seule ligne de code SQL.

Fonctionnement d'ObjectSpaces

Ce produit est constitué :

- d'un framework de persistance
- d'un outil de mapping objet relationnel intégré à l'environnement de développement, qui permet de définir des schémas XML assurant la correspondance entre les objets et les tables d'une base de données

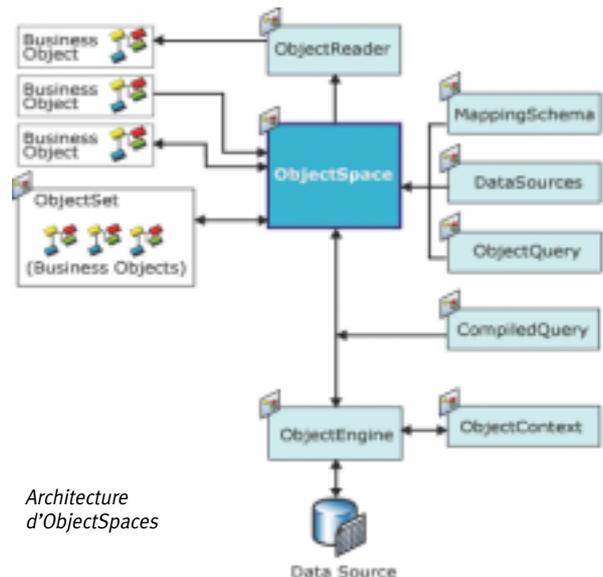
Le développeur va désormais instancier des classes du modèle ObjectSpaces et non plus ADO .Net et leur fournir des objets personnalisés qui contiendront les données à traiter. Le moteur ObjectSpaces se charge de traduire les requêtes sur ces objets en requêtes SQL et d'appliquer les modifications sur ces objets vers les tables correspondantes en base de données. De même, en cas de recherche de données, le moteur se charge de les récupérer et de les stocker dans des instances d'objets .Net. Il s'appuie pour réaliser ces opérations sur des méta-données XML stockées dans un fichier de mappage, qui est transmis au constructeur de la classe ObjectSpaces. Celui-ci mappe les objets relationnels avec les objets .Net et les types relationnels avec les types .Net.



Quand utiliser ObjectSpaces ?

Il est recommandé d'utiliser cet outil lorsque vous disposez d'une couche métier importante et que vous avez un modèle objet à persister sur un espace de stockage. ObjectSpaces exploite de nombreux concepts de mapping objet/relationnel que je souhaite vous exposer :

- Il supporte des relations 1-1, 1-n, n-n entre les classes
- Il gère les transactions (les méthodes BeginTransaction, Commit et Rollback de l'objet ObjectSpaces)



Architecture d'ObjectSpaces

- Il gère la concurrence d'accès de manière optimiste et lève automatiquement une exception typée (PersistenceException), permettant au développeur d'effectuer les traitements correspondants.
- Il propose un langage de requêtage OPath permettant d'effectuer des recherches dans des objets comme le ferait XPath pour des documents XML.
- Il propose un mécanisme de chargement de données à la demande, portant le nom de Delay Loading (lazy loading). Cela signifie que si un objet parent est accédé, mais pas un objet fils, les objets fils ne sont pas chargés. Ceci se traduit par un gain en performance et en consommation mémoire. Si un objet fils est accédé alors, il sera chargé à la demande.
- Il est non intrusif avec le modèle objet existant

L'architecture d'ObjectSpaces

Les classes présentées dans le schéma d'architecture ci-dessus sont incluses dans l'espace de nom System.Data.ObjectSpaces. La classe principale de cette architecture est la classe ObjectSpaces. Elle permet de récupérer les données d'une source, d'identifier les objets à persister, et de contrôler les transactions. Elle s'appuie sur d'autres classes :

Classe	Description
ObjectReader	Propose un flux d'objets en lecture seulement qui résulte de l'exécution d'un ObjectQuery sur une source de données
ObjectSet	Correspond à une collection d'objets
Mapping Schema	Identifie comment les propriétés des objets sont mappées aux champs d'une source de données
ObjectSources	Propose des informations de connexion vers une source de données
ObjectQuery	correspond à une requête d'un objet
ObjectEngine	
ObjectContext	Utilisé pour identifier et gérer les versions des objets afin d'assurer entre autres la gestion de la concurrence
ObjectExpression	Utilisé pour précompiler des requêtes à destination de l'ObjectEngine

Définition du mapping objet/relationnel

Dans notre exemple, nous allons définir le mapping de persistance entre nos objets métiers Client et Commandes et les tables Customers et Orders de la base de données NorthWind.

Afin d'assurer ce mapping, ObjectSpaces se base sur trois fichiers XML :

- **Object Schema Document (OSD)**
Ce fichier définit les objets métiers, leurs propriétés, ainsi que les relations entre les différentes classes
- **Relational Schema Document (RSD)**
Ce fichier contient des informations sur les tables du modèle relationnel, les champs de chaque table et les relations entre elles
- **Mapping Schema Document (MSD)**
Ce fichier fait référence aux deux autres fichiers et décrit les relations entre le schéma RSD et OSD. Ainsi, il va décrire dans notre exemple que la classe Client est mappée à la classe Customers et que le champ Nom correspond au champ CompanyName.

Nous utilisons pour constituer ces fichiers l'éditeur de schéma graphique proposé dans Visual Studio .Net 2005.

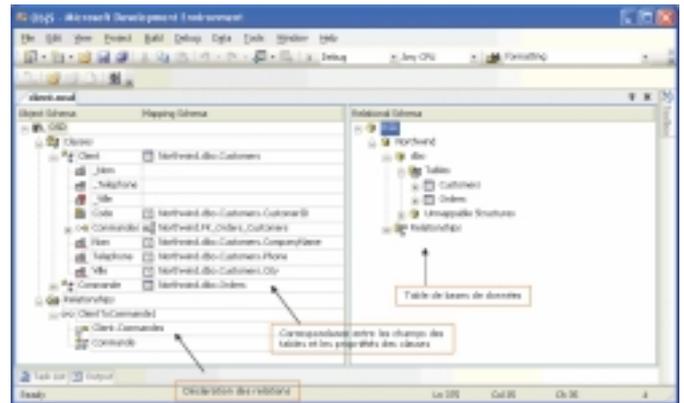
Voici un extrait du fichier généré :

```
...
<m:Mappings>
  <m:Map Source="Northwind.dbo.Customers" Target="Client">
    <m:FieldMap SourceField="CustomerID" TargetField="Code" />
    <m:FieldMap SourceField="CompanyName" TargetField="Nom" />
    <m:FieldMap SourceField="Phone" TargetField="Telephone" />
    <m:FieldMap SourceField="City" TargetField="Ville" />
    <m:RelationshipMap Source="Northwind.FK_Orders_Customers"
Target="Commandes" />
  </m:Map>
  <m:Map Source="Northwind.dbo.Orders" Target="Commande">
    <m:FieldMap SourceField="OrderID" TargetField="Identifiant" />
    <m:FieldMap SourceField="OrderDate" TargetField="Date
Commande" />
  </m:Map>
</m:Mappings>
...
```

Le langage de requêtage OPath

A l'instar de SQL, pour interroger une base de données, Microsoft a créé un langage de requêtage portant le nom de OPath. Comme pour du SQL ou des requêtes XPath, les requêtes OPath sont des chaînes de caractères utilisées pour récupérer un sous-ensemble d'objets d'une source de données.

L'utilisation du prédicat [] permet de filtrer les données. L'utilisation du



L'éditeur de mapping O/R

point (.) permet de naviguer dans les relations d'un objet. Exemples de requête OPath

```
Orders.Details.Quantity > 50
Filtre sur les commandes dont la quantité d'un des articles est > 50

Orders[ShippedDate > RequiredDate]
Liste des objets commandes pour lesquels le champ ShippedDate est
supérieur au champ RequiredDate

Orders[Freight > 1000].Details.Quantity > 30
Filtre sur les commandes dont la quantité d'un des articles est > 30
mais seulement pour les objets commandes dont le champ Freight est supérieur
à 1000
```

La récupération d'objets d'une source de données

Les objets ObjectSpaces, ObjectQuery et ObjectReader

La première étape pour récupérer des données consiste à instancier un objet de type ObjectSpaces. Cet objet prend comme paramètres un schéma XML (client.msd) et une connexion vers une source de données. Ensuite, il faut préparer la requête souhaitée en utilisant un objet Query en lui indiquant le type d'objet à récupérer et la requête OPath à exécuter. Enfin, il ne reste plus qu'à exécuter la requête, via un objet de type ObjectReader et la méthode GetObjectReader. L'objet ObjectReader permet de récupérer le résultat d'un ObjectQuery sous la forme d'un flux d'objets forward-only. Nous parcourons ce flux selon deux méthodes équivalentes (foreach loop ou while loop).

```
Imports System.Data.ObjectSpaces
...
Dim ospace As ObjectSpace
Dim oquery As ObjectQuery(Of Client)
Dim oreader As ObjectReader(Of Client)

ospace = New ObjectSpace("../client.msd", New SqlConnection("Data source=(local);User ID=sa;Initial Catalog=Northwind"))

oquery = New ObjectQuery(Of Client)("Client.Nom LIKE 'A%'")

oreader = ospace.GetObjectReader(oquery)
```

```
Dim c As Client
For Each c In oreader
    lb.Items.Add("Nom : " + c._Nom + " Ville : " + c._Ville)
Next

oreader = ospace.GetObjectReader(oquery)

Do While oreader.Read()
    c = CType(oreader.Current, Client)
    lb.Items.Add("Nom : " + c._Nom + " Ville : " + c._Ville)
Loop
oreader.Close()
```

Voici à l'exécution le résultat de notre requête.

[Fichier "Application de test.bmp, avec comme titre " Formulaire de test d'ObjectSpaces"]



Formulaire de test d'ObjectSpaces

L'objet ObjectSet

ObjectSpaces propose également un autre objet permettant de récupérer le résultat de l'exécution d'un ObjectQuery, il s'agit de l'objet ObjectSet. Il stocke en mémoire les objets issus d'une source de données. Il conserve en mémoire les valeurs d'origines de champs, permettant ainsi la gestion de la concurrence optimiste. Il peut également être associé à des DataControl afin d'assurer du DataBinding. Pour ceux qui sont familiers d'ADO .Net, il s'apparente à un objet DataSet. Voici la requête précédente modifiée afin d'utiliser un objet ObjectSet.

```
...
oquery = New ObjectQuery(Of Client)("Client.Nom LIKE 'A%'")
oset = ospace.GetObjectSet(oquery)

Dim c As Client

For Each c In oset
    lb.Items.Add("Nom : " + c._Nom + " Ville : " + c._Ville)
Next
```

Imaginons que nous souhaitons récupérer des informations sur un client, dont le numéro de téléphone est le '0621-08460' et obtenir la liste de ses commandes.

Pour ce faire, il faut réaliser des modifications dans notre ObjectQuery. Tout d'abord, modifions notre requête OPath en lui indiquant le numéro de téléphone, puis en complétant le paramètre Span qui indique à quelle profondeur de la hiérarchie d'objets naviguer. Dans notre cas, nous voulons que les commandes soient également retournées à l'exécution de la requête. Il ne reste plus qu'à parcourir la liste des commandes de notre client en utilisant un for each.

```
...
oquery = New ObjectQuery(Of Client)("Client.Telephone='0621-08460'", "Commande")

oreader = ospace.GetObjectReader(oquery)
```

```
Dim client As Client
Dim commande As Commande

For Each client In oreader
    lb.Items.Add("Nom : " + client._Nom + " Ville : " + client._Ville)
    For Each commande In client.Commandes
        lb.Items.Add("N°Commande : " + commande._Identifiant + " Date : " +
            Convert.ToString(commande.DateCommande))
    Next
Next

oreader.Close()
```

La gestion des transactions

ObjectSpaces utilise un modèle de gestion de la concurrence " optimiste " pour gérer les mises à jour vers une source de données. Il conserve une copie des valeurs d'origine des propriétés des objets à persister et peut, en cas de concurrence d'accès, lever une exception de Type " PersistenceException ". Il est ainsi possible de récupérer une référence vers l'objet à l'origine de cette exception et assurer une gestion d'erreur personnalisée.

L'objet ObjectSpaces propose également les fonctionnalités classiques assurant la gestion des transactions. L'écriture des données d'un objet vers une source de données est permise par la méthode PersistChanges qui prend en paramètre un objet ou une collection d'objets à persister. Pour vérifier le fonctionnement de la méthode PersistChanges, nous avons modifié le nom de l'objet Client " Around the Horns " et " Around the Horn " et exécuté le code suivant.

```
Try
    ospace.BeginTransaction()
    ospace.PersistChanges(oset)
    ospace.Commit()
Catch err As PersistenceException
    ospace.Rollback()

    Dim pe As PersistenceError
    For Each pe In err.Errors
        ' Gestion d'erreur personnalisée
    Next

End Try
```

Voici la requête SQL générée par ObjectSpaces pour persister notre objet que nous avons capturé à l'aide du général de profils SQL2000. Nous constatons bien que l'ordre SQL va effectuer une mise à jour du nom de la compagnie AROUT (CustomerId) à " Around the Horn ".

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;BEGIN
TRANSACTION
```

```
exec sp_executesql N'
Update TO Set TO.[CompanyName]=@C4
from [N orthwind].[dbo].[Customers] TO
Where (((TO.[CustomerId] Is Null) and (@O0 Is Null)) or TO.[CustomerId]
=@O0) and (((TO.[PostalCode] Is Null) and (@O21 Is Null)) or TO.[Postal
Code]=@O21) and (((TO.[CompanyName] Is Null) and (@O3 Is Null)) or
TO.[CompanyName]=@O3) and (((TO.[City] Is Null) and (@O15 Is Null))
or TO.[City]=@O15) and (((TO.[Phone] Is Null) and (@O27 Is Null)) or
TO.[Phone]=@O27);
```

```
Declare @r int, @e int;Set @r=@@ROWCOUNT; Set @e=@@ERROR;If @e
= 0 Begin if @r = 0 RAISERROR("No rows affected.",16,1);If @r > 1 RAI
SERROR("Multiple rows affected.",16,1);End', N'@C4 nvarchar(15),@
O0 nchar(5),@O21 nvarchar(7),@O3 nvarchar(16),@O15 nvarchar(6),
@O27 nvarchar(14)', @C4 = N'Around the Horn', @O0 = N'AROUT',
@O21 = N'WA1 1DP', @O3 = N'Around the Horns', @O15 = N'
London', @O27 = N'(171) 555-7788'
```

```
COMMIT TRANSACTION
```

La suppression des objets

Pour supprimer un objet d'une source de données, il faut d'abord sélectionner cet objet, le marquer comme élément à supprimer (via la méthode MarkForDeletion) et enfin assurer sa persistance.

```
Try
Dim c As Client
c = oset.Item(0)

ospace.BeginTransaction()
ospace.MarkForDeletion(c)
ospace.PersistChanges(oset)
ospace.Commit()
Catch err As PersistenceException
ospace.Rollback()
Dim pe As PersistenceError
For Each pe In err.Errors
' Gestion d'erreur personnalisée
Next
End Try
```

Cet outil peut répondre aux besoins d'applications qui ont implémenté une logique métier importante et souhaitent bénéficier d'un framework de persistance .Net. Certes, la version utilisée est en bêta test, il faudra attendre la version finale pour pouvoir l'utiliser en production. Il présente bien la plupart des concepts de base du mapping objet/relationnel. Un regret tout de même : à ce jour, ObjectSpaces ne fonctionne qu'avec SQL Server 2000 et Yukon. ■ **Christian Peyrusse**

ESIG

www.esig-est.com

Ouverture
le 18 octobre
à Nice de la première
session consacrée
à l'industrie du jeu vidéo.

Game designer

Sound designer

Développeur
(consoles, mobiles et micros)

Animateur 2D/3D

Chef de projet

Niveau d'admission : BAC+2

ESIG Nice
04.92.07.92.07
nice@esig-est.com



Migration de Delphi 7 à Delphi 8

L'arrivée de Delphi, en 1995, fut une véritable révolution dans le domaine des L4G, où l'on pouvait pour la première fois glisser et déplacer ses composants pour construire une application Client /Serveur. En 2004, Borland réitère l'exploit en lançant une version de Delphi pour le framework .NET. Il s'agit de la version de Delphi 8 dont nous allons étudier en détail les aspects de migration à partir d'une application Delphi 7 Win 32. Nous traiterons également des nouveautés apportées par cette version.

Rappel sur la VCL

La version 8 de Delphi contient des composants visuels, mais aussi toute une série de composants préexistants en Delphi 7 (issus de la VCL) et des composants ajoutés à partir du framework .NET. Cette nouvelle version 8 permet aux développeurs d'effectuer la migration vers la nouvelle plate-forme .NET, tout en supportant les besoins des applications déjà existantes.

Les versions antérieures à Delphi 8 permettaient de produire des applications natives pour Win32. De plus, Borland Kylix permet de construire des applications sous Linux en Delphi. Les développeurs vont dorénavant pouvoir choisir entre les applications natives en Windows Forms ou des Web Forms .NET en ASP.NET, ou utiliser la VCL traditionnelle de Delphi portée en .NET. Dans certains cas il n'est même pas nécessaire de choisir, une même application pouvant mixer à la fois les deux frameworks, VCL.Net et FCL (.Net Framework Class Library).

Vers une migration en .NET

Dans la plupart des cas, la migration entre la VCL Delphi Win32 et Delphi pour .NET est possible. Delphi 8 permet de compiler les applications qui ont été réalisées en Delphi 7 ou dans des versions précédentes, via certains changements.

Nous remarquerons de plus qu'il reste toujours possible de créer à partir de Delphi 8 des applications Win32, car Delphi 8 contient le compilateur déjà fourni pour Delphi 7. Or Delphi 7 permettait déjà de faire du code en .NET ou du Win32. La compilation nécessite le choix d'une définition IFDEFs à l'intérieur du code source.

Delphi 7 contient par défaut les lignes de définitions suivantes :

```
MSWINDOWS
WIN32
```

Pour Delphi 8 le compilateur contient les lignes de définitions suivantes :

```
CLR
FCL
MANAGEDCODE
```

Cela signifie que l'on a la possibilité de faire des choix de compilation suivant le type d'application souhaité.

```
project HelloWorld;
{$APPTYPE CONSOLE}
begin
  {$IFDEF CLR} // Delphi pour .NET
    writeln('Bonjour, Framework.NET!');
  {$ENDIF}
  {$IFDEF WIN32} // Delphi 7
    writeln('Bonjour, application Win32!');
  {$ENDIF}
  {$IFDEF LINUX} // Kylix
    writeln('Bonjour, monde Linux!');
  {$ENDIF}
end.
```

Noter que l'on a le choix entre trois plates-formes. Dans ce cas, il n'est pas souhaitable d'utiliser la close {\$ELSE}. De plus, on remarquera que même si le code est acceptable aujourd'hui, Borland pourra être amené à l'avenir à supporter d'autres plates-formes. Il est donc conseillé d'utiliser exclusivement la clause {\$IFDEF} pour écrire du code sur une plate-forme spécifique.

VCL, VCL pour .NET, et Forms Windows®

La VCL est la librairie de composants graphiques historique de Delphi. Cette librairie contient plus que de simples composants visuels. Or la VCL existait uniquement pour les plates-formes Win32.

Maintenant que Delphi a été porté sur le Framework .NET, la VCL a de même migré en .NET. Ceci signifie que nous pourrons utiliser non seulement les winforms .NET mais aussi la VCL .NET pour la portabilité de nos anciennes applications. Cette portabilité est due au fait que les événements et les propriétés de la VCL Win32 ont migré en Delphi .NET. Nous remarquons que l'utilisation de la VCL, la CLX et la VCL.NET restent similaires. Pour les winforms il n'y a pas de fichier séparé (de type DFM) pour la principale raison que tous les gestionnaires d'événements sont gérés dans le code source.

La VCL .NET est plus spécifiquement une encapsulation des classes winforms du Framework .NET. Le passage de la VCL win32 à la VCL .Net est plus aisé que le passage direct de l'API win32 au Framework .NET. Nous constatons donc que le passage de la VCL à la VCL.NET permet l'extension de la durée de vie des applications écrites en Delphi.

Les incompatibilités Delphi 7 vers Delphi 8 pour .NET

Bien que la migration de la VCL win32 vers la VCL.NET se fasse relativement simplement, des améliorations ont du être apportées à Delphi 7 concernant les types et fonctionnalités du langage. Effectivement, les applications .NET tournent sur la machine virtuelle de Microsoft : le CLR. Nous nous trouvons alors dans un cadre d'exécution sécurisé incompatible avec certaines fonctionnalités de Delphi 7.

De nombreuses caractéristiques du langage Delphi 7 ont donc du être corrigées par rapport à Delphi 8 qui devient encore plus objet et "propre". Voici dans le tableau suivant les types, pointeurs, méthodes et procédures qui ont été modifiés.

Caractéristique du langage Delphi™ 7	Recommandation pour Delphi 8
Real8	Double
absolut, Addr, @	Non supporté
GetMem, FreeMem, ReAllocMem	New et Dispose, ou structure tableau(amas)
La classe Borland Pascal "object"	Class type .net
Fichiers de tout type incluant les structures record	Streams, Serialization, databases
Code inline et mot def assembleur asm	Non supporté
ExitProc	Non supporté
RRChar, Move	Réécrit au travers de boucles for
interface Delphi 7 DOOM	Véritable interface .Net
PChar	String ou StringBuilder 1)

Code non sécurisé

Delphi 7 peut nous préparer à faire des applications en Win32. Delphi 8 va nous permettre de les passer en .NET mais avec l'installation de warnings. Ces warnings sont retirés par défaut. Nous pouvons activer ces warnings en utilisant Project | Options - Compiler Warnings. Les nouveaux warnings tournent autour de types unsafe (insécurisé), de code unsafe, des cast de type unsafe. Vous pouvez activer, par du code, les warnings suivants :

```
{#WARN UNSAFE_TYPE ON}
{#WARN UNSAFE_CODE ON}
{#WARN UNSAFE_CAST ON}
```

Vous devez insérer ces warnings en début de chaque unité de code. Pour les types non sécurisés, le compilateur notifie, quand vous déclarez des PChar, des pointers non typés, des fichiers de type ?, des Real48, des records variants, ou quand vous utilisez des passages par variable non typés. Confronté à du code " insécurisé " (" non safe "), vous recevez des warnings, quand vous utilisez les fonctions Addr, Ptr, Hi, Lo, Swap, BlockRead, BlockWrite, GetMem, FreeMem et ReallocMem. Finalement, chaque cast de type avec des pointeurs ou des objets qui sont considérés comme non fonctionnels, provoquera des warnings. Si vous ne pouvez pas remplacer du code non sécurisé (unsafe) pour votre source, vous devrez marquer votre code comme non sécurisé pour assurer la portabilité avec les instructions {\$UNSAFECODE ON} ... {\$UNSAFECODE OFF}.

Nouvelles caractéristiques de langage

Delphi 8 .NET a aussi introduit de nombreuses nouvelles extensions pour implémenter la plate-forme .NET. Par exemple, des classes scellées (sealed) ont été introduites ainsi que des méthodes finales (final). Pour se conformer au code .NET standard, qui indique qu'un attribut private ne se voit que de l'intérieur de la classe (tout comme le mot-clé protected qui permet de voir un attribut pour les sous-classes), le nouveau mot-clé " strict " doit être placé avant private et protected. Ce mot-clé n'est pas respecté par Delphi 7 mais seulement dans Delphi 8 .NET. Une grande nouveauté de Delphi 8 est la surcharge d'opérateurs. Les opérateurs surchargés sont les suivants :

Opérateurs	Surchargeabilité
$+$, $-$, $*$, div , mod , and , or , xor , shr , shl , not	Ces opérateurs unaires peuvent être surchargés.
$+$, $-$, $*$, div , mod , and , or , xor , shr , shl	Ces opérateurs binaires peuvent être surchargés.
$+$, $-$, $*$, div , mod , and , or , xor , shr , shl	Les opérateurs de comparaison peuvent être surchargés (voir la remarque ci-dessous).
$+$, $-$, $*$, div , mod , and , or , xor , shr , shl	Les opérateurs logiques et bits à bits.
$+$, $-$, $*$, div , mod , and , or , xor , shr , shl	L'opérateur de cast explicite et explicite sont surchargeables.

```
type
  TMyClass = class
    class operator Add(a, b: TMyClass): TMyClass; // Ajout de deux opérandes de
  type TMyClass
    class operator Subtract(a, b: TMyClass): TMyClass; // Soustraction de type TMyClass
    class operator Implicit(a: Integer): TMyClass; // Conversion implicite d'un
entier en type TMyClass
    class operator Implicit(a: TMyClass): Integer; // Conversion implicite de
TMyClass en Integer
    class operator Explicit(a: Double): TMyClass; // Conversion explicite d'un
double en type TMyClass
  end;

// Exemple d'implémentation de Add
class operator TMyClass.Add(a, b: TMyClass): TMyClass;
begin
  // ...
end;

var
  x, y: TMyClass;
begin
  x := 12; // Conversion implicite à partir d'un Integer
  y := x + x; // Appelle TMyClass.Add(a, b: TMyClass): TMyClass
  b := b + 100; // Appelle TMyClass.Add(b, TMyClass.Explicit(100))
end;
```

Les composants VCL Delphi 7 qui ne sont pas dans la VCL .NET de Delphi 8

Un certain nombre de composants de la VCL de Delphi 7 n'ont pas été portés dans la version de Delphi 8 .NET. Voici la liste des catégories de composants non portés pour Delphi 8 pour .NET : dbGO pour ADO, WebBroker, InternetExpress, WebSnap, et le support des fichiers XML au travers du composant TXMLTransform.

De la VCL win32 à la VCL pour .NET

Les composants principaux de la VCL Win32 apparaissent dans la VCL .NET. Pour des raisons de sécurité, certains composants de la VCL en Win32 n'ont pas pu être portés par la palette outil (Tool Palette). Néanmoins, des catégories similaires à Delphi 7 sont présentes : la catégorie Standard, Additionnel, Win32, System, Win 3.1, Dialogs, Data Access, Data Control, dbExpress, DataSnap, BDE, Interbase, Interbase Admin, Indy Clients, Indy I/O Handlers, Indy Intercepts, et Indy Misc.

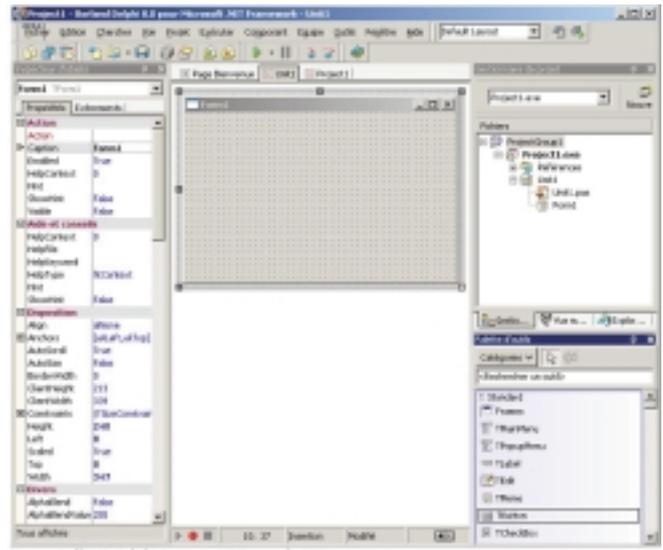


Figure 1. IDE Delphi 8 pour .NET avec la VCL

Pour les applications VCL

Sur près de 300 composants de la VCL.Net, il n'en manque qu'une dizaine. Les composants manquants sont TChart, TActionManager, TActionMainMenuBar, TactionToolBar, TXPColorMap, TStandardColorMap, TWilighColorMap et TCustomizeDlg. Une recherche plus approfondie témoigne de l'absence du composant TActionManager, listé dans l'aide et dans l'espace de nom namespace Borland.Vcl.ActnMan, mais qui n'est pas présent dans la palette des outils. Pour ce qui concerne l'onglet Win32, pour VCL .NET, les composants TDBLookupList et le TDBLookupCombo n'ont pas été portés.

Clauses de portabilité de Delphi 7 à Delphi 8

Avant de migrer une application Delphi 7 vers Delphi 8 pour .NET, il est souhaitable de transformer le nouvel exécutable en un code sécurisé au travers de l'outil PVerify. Si lors de la recompilation d'un exe Delphi 7 en Delphi 8, nous détectons des warnings d'erreur .NET, c'est souvent parce que des composants manquent.

Dans ce cas, la clause `{$IFDEF CLR}` peut être utilisée, notamment pour introduire des composants .NET du type `System.ComponentModel`, qui fournissent les interfaces nécessaires aux composants de la VCL.NET.

```
uses
  Windows, Classes, Graphics, Forms, Controls,
  {$IFDEF CLR} System.ComponentModel, {$ENDIF}
  StdCtrls;
```

Ceci implique que pour chaque passage de l'unité Delphi 7 à Delphi 8, cette clause de compilation devra systématiquement être rajoutée pour assurer une compatibilité avec du code écrit en Delphi 7. Ainsi, on garde la possibilité de créer un exe Win32 ou .NET (au choix) pour le même code source. Nous constatons ici que le transfert total et effectif en .NET n'est donc pas encore au rendez-vous, puisqu'il nous reste la possibilité de passer en Win32, solution qui a été volontairement abandonnée par Microsoft pour passer en full .NET.

Déploiement d'une application en VCL pour .NET

Avant chaque migration d'une application écrite en Delphi 7 pour Delphi 8 - supportant une base de données - il convient de répertorier les composants de la VCL impactés. Si l'on prend par exemple une application écrite pour Win32 pour la faire migrer en .NET, on s'aperçoit que seule une référence est faite à l'assembly .NET `System.Drawing.dll`. Les assemblies .NET en Delphi 8 peuvent être considérées comme des packages chargés lors de l'exécution au runtime. Dans certains cas, il est alors intéressant de faire une assembly .NET de l'application, plutôt que de mettre en place un très gros projet purement Delphi.

En rajout, le développeur devra choisir les assemblies .NET nécessaires au fonctionnement de la VCL en tant que paramétrage par défaut. Ceci implique que ces petits exécutables " assemblies " devront être aussi déployés. Par ailleurs, quand on migre un projet VCL de Delphi 7 en Delphi 8 .NET, nous remarquons que certaines assemblies de fonctionnement de base de la VCL.NET ne sont pas présentes. Ceci implique qu'il faut ajouter manuellement les assemblies aux projets, afin que les unités Delphi compilent le tout en un exécutable monolithique complet. En priorité, il faut référencer les assemblies `Borland.Delphi.dll` ainsi que l'assembly `Borland.Vcl.dll`.

Application avec base de données.

De nombreuses technologies sont désormais utilisables pour le Framework .NET. Par exemple, nous avons accès à l'ADO.NET, ce qui rend ainsi de nombreux projets Win32 obsolètes. Cette section de l'article va nous indiquer ce qui est utilisable en Delphi .NET.

Le tableau suivant nous montre ce qui a été porté de Delphi 7 à Delphi 8.

Delphi 7	Delphi 8
Borland® Database Engine (BDE) (dBASE, Paradox®)	BDE (dBASE, Paradox)
SQL Links	Déprécié
Borland® dbExpress (Interbase®, Microsoft® SQL Server®, Oracle®, IBM® DB2®, Informis®)	dbExpress (Interbase, SQL Server, Oracle, IBM DB2, Informis, SQL Anywhere®)
Borland® Interbase™ (IBX)	Interbase (IBX)
Borland® dbGo™ for ADO	Pas disponible pour l'instant

Table 2: Voici les technologie d'accès aux données de Delphi 7 à Delphi 8.

A part `dgGo` pour ADO, qui n'a pas été porté en .NET, nous avons la possibilité de choisir le BDE (pour dBASE et Paradox). Nous pouvons aussi utiliser `dbExpress` et `interbase Express (IBX)`. Il est ici important de signa-

ler qu'il faut mettre l'accent sur la migration complète de `dbExpress`, le BDE étant déprécié depuis mi-2002. Nous remarquerons que le Décision cube pour .NET n'a pas été porté.

Les composants d'accès aux données

Dans la catégorie des composants d'accès aux bases de données, le `TXMLTransform`, `TXMLTransformProvider` et le `TXMLTransformClient` ne sont pas inclus dans Delphi 8 .NET. Les composants de contrôles de données sont complets à l'exception du `TDBChart`. En ce qui concerne le BDE, les composants `TTable`, `TQuery`, `TDatabase`, `TSession` et `TBatchMove` sont conservés. Nous remarquons que les composants SQL Links suivants ne sont pas portés : `TStoredProc`, `TUpdateSQL`, `TNestedTable` (qui ne sont donc pas présents dans la palette des composants, mais qui peuvent être trouvés dans la VCL pour .NET dans l'unité `Borland.Vcl.DBTables.pas`).

Dans la catégorie `dbExpress`, tous les composants ont été portés, à l'exception du `TSimpleDataSet`.

Pour la catégorie `Interbase` et `Interbase Admin`, les composants sont complets, à l'exception du composant `TIBEvents` de l'onglet `Interbase`, ainsi que le Composant `TIBInstall` et `TIBUninstall` de l'onglet `Interbase Admin`. Le `DataSnap` ne contient plus que les composants `TDCOMConnection`. Les composants `TSocketConnection`, `TSimpleObjectBroker`, `TWebConnection`, `TConnectionBroker`, `TSharedConnection` et `TLocalConnection` n'ont pas été portés. On remarquera que le composant `TConnectionBroker` peut être trouvé dans l'unité `Borland.Vcl.DBClient.pas`. Le `TSharedConnection` se trouve dans l'unité `Borland.Vcl.MConnect.pas` et le `TLocalConnection` dans l'unité `Borland.Vcl.TConnect.pas`.

Application Web

La technologie ASP.NET pour le framework .NET permet au développeur .NET de construire une application déployable autour d'un serveur IIS, qui peut être désignée en Drag and Drop directement sur une page HTML. L'application web se déploie en des scripts CGI et des DLL ISAPI. Cela signifie que les anciens composants autour de `WebBroker`, `InternetExpress` et `WebSnap` n'ont pas été portés, puisque ASP.NET invalide ces composants. Il est donc préférable de continuer à maintenir les applications tournant autour de ces composants en Delphi 7 et de passer, pour de nouvelles applications web, en Delphi 8 avec ASP.NET.

Les Web Services

Delphi 6 avait introduit des composants tournant autour de web services et supportant le protocole SOAP. Le Framework .NET supporte de même le protocole SOAP et construit ses web services autour de la technologie.NET. Ceci implique que les web services en Delphi .NET sont implémentés autour du framework .NET. Les anciens composants Delphi 6 et 7 ne sont donc plus supportés. Ici, de la même manière, nous voyons que l'objectif de Borland est d'utiliser le plus possible les technologies fournies par le framework .NET de Microsoft pour les web services.

Divers problèmes

Trois catégories de la VCL de Delphi 7 n'ont pas été directement portées sur Delphi .NET. Il s'agit des `ActiveX`, `COM+` et de l'onglet `Servers`. Néanmoins, il est possible d'importer les composants COM et ActiveX en les ajoutant en référence du projet (voir copie d'écran).

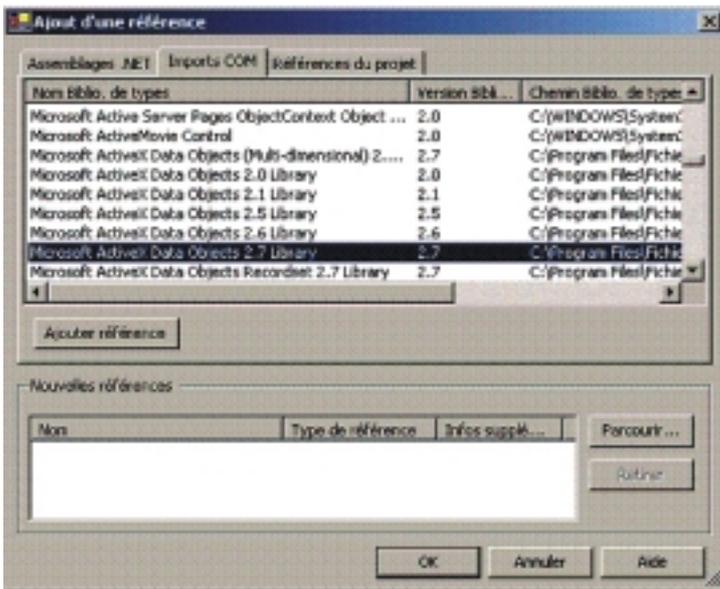


Figure 3: imports d'objet ActiveX

Il est alors possible de conserver l'interopérabilité de l'ancien code écrit pour du Win32, mais sans nous garantir que le code .NET soit sécurisé comme espéré. Néanmoins, bien que la catégorie des objets COM et ActiveX ne soit pas présente, il est possible de trouver des exemples d'inclusion d'objets COM au niveau du répertoire Demos\Vcl\Samples qui inclut le package Borland.Vcl.Samples.

Si cela ne s'avère pas suffisant, vous pouvez utiliser les composants de contrôle standard de la VCL pour .NET.

Le passage de la VCL Delphi 7 vers Delphi 8 est relativement simple. De plus, il est possible de choisir entre les composants WinForms et la VCL. Au travers de la VCL, vous pouvez utiliser pratiquement tous les accès aux données standard du type BDE, dbExpress, IBX, ou le DataSnap. Pour les WinForms, il est possible d'utiliser les composants ADO.NET ou le provider de données Borland Data Providers pour .NET.

En première remarque, nous constatons que certains composants de Delphi 7 n'ont pas été migrés en Delphi 8. Il ne faut donc pas s'attendre à ce que la migration d'une application Win32 se fasse automatiquement sous Delphi 8. ASTON souhaite donc accompagner les utilisateurs vers le portage ou une partie de la réécriture de leur application en Delphi sous le Framework .NET de Microsoft.

En seconde remarque, ce que nous espérons dans le futur, c'est que sur cette nouvelle base de travail autour de .NET, Borland nous fournisse de nouveaux composants intelligents encapsulant des fonctionnalités .NET tels que ceux encapsulant les fonctionnalités de l'API Win32. Les développeurs Delphi pourraient alors continuer à programmer en Pascal Objet, tout en bénéficiant du savoir-faire de Borland en terme de création d'applications en " glisser-déplacer ", ce qui s'avérerait particulièrement productif. Ainsi, la philosophie de programmation Delphi, pour les versions 8 et suivantes, serait conservée et continuerait à séduire les nombreux passionnés de cet outil fantastique.

■ **Nicolas Sciara**, Consultant ASTON Education



Aston est une SSII de 270 personnes, implantée à Paris, Rhones-Alpes et Sud Ouest. Pour en savoir plus sur ASTON : www.aston.fr

PHP 5 avancé

Cyril Pierre de Geyer
Eric Daspet



730 pages,
457 versions,
30 cas pratiques,
26 chapitres,
36 mois de travail,
10 ans d'expérience,
5ème version,
2 auteurs,
un livre



recommandé par :





Des vitamines C++ pour Java

Marier Java et C++ présente de nombreux avantages. La JNI ou Java Native Interface de la machine virtuelle Java permet aisément de faire collaborer les deux langages.

Java et C++ sont deux langages majeurs du monde de l'informatique, avec chacun leurs points forts et leurs inconvénients. Il y a de nombreuses raisons justifiant le choix de Java pour le développement d'une application. Java est relativement simple et la portabilité du code compilé peut séduire. Java est très sécurisé et il offre des facilités sympathiques pour la programmation réseau. En revanche, Java souffre de sa lenteur à l'exécution, même si la rapidité des machines et les performances des compilateurs "Just-In-Time" vont croissant. C++ lui, n'a rien de simple, mais est facilement portable au niveau du code source. Il présente sur Java l'avantage de la performance brute. En outre, des millions de lignes de codes ont été écrites avant la naissance de Java. Si le code est éprouvé et débogué, pourquoi le réécrire en Java, avec les problèmes potentiels que pose toujours une réécriture dans un autre langage, sans parler du coût en terme de temps de développement. Les concepteurs de Java ont perçu cela et ont conçu une interface, la JNI, pour que la JVM puisse invoquer du code natif et réciproquement, pour que du code Java puisse être invoqué par du code natif. Lorsque les performances sont critiques, lorsqu'il s'agit de réutiliser du code et aussi, n'oublions pas ce cas de figure, quand la JVM n'offre pas une fonctionnalité et qu'un appel aux APIs de l'OS sous-jacent est incontournable, la JNI vous offre le moyen de vous tirer d'affaire, tout en étant bien conscient que cela peut avoir des répercussions sur la sécurité, car il est bien évident qu'une méthode native d'une classe Java n'offre pas plus de sécurité que tout autre code C++. Il convient donc de faire attention à ce que l'on écrit. Moyennant quoi le mariage de C++ et de Java peut apporter de beaux enfants.

L'interface JNI

La JNI est bien conçue. Elle est facile et souple à utiliser et ses possibilités sont complètes. Pour qu'il soit possible d'accéder aux méthodes de la JNI, la JVM fournit un pointeur sur l'interface. Par ce pointeur, qui

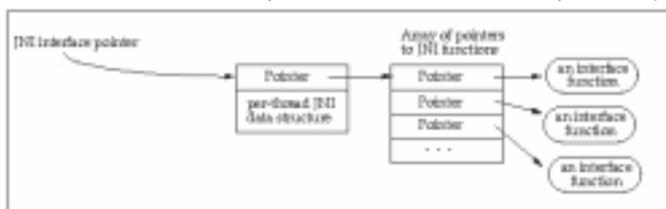


Figure 1 : Organisation de la JNI

est en quelque sorte un pointeur sur des pointeurs de fonctions, il est possible d'invoquer les méthodes de la JNI. D'après la documentation Java, (Figure 1) les pointeurs de fonctions sont rangés dans un tableau qui ressemble à une table de fonctions virtuelles d'une classe C++.

Déclarer une méthode native

Rien n'est plus simple que de déclarer une méthode native en Java.

Supposons que nous voulions écrire une méthode qui renvoie le double de la valeur qu'elle reçoit en paramètre. En Java pur tout d'abord, cela s'écrit comme ceci :

```
class maclasse
{
    public double doubleJava(double valeur)
    {
        return 2*valeur;
    }
}
```

La méthode doubleJava est écrite en Java, donc le corps de la méthode est présent. Si nous voulons transformer cette méthode en une méthode native que l'on baptisera doubleNative, il suffit de supprimer le corps de la fonction et d'ajouter le mot clé 'native'. Enfin, la déclaration de la méthode sera suivie de ; comme ceci :

```
class maclasse
{
    public native doubleNative(double valeur);
}
```

Le corps de la méthode ne se situe donc plus dans la classe. Il doit faire partie d'une librairie partagée qui doit être chargée avant l'utilisation de la classe. Une édition de liens dynamique est effectuée par la JVM, lors du chargement de cette librairie. Le moyen le plus simple pour charger une librairie native, ou plus exactement pour demander à la JVM de le faire, consiste à placer le code de chargement dans un bloc static dans la classe. Un bloc static étant exécuté au premier chargement de la classe, nous sommes ainsi assurés que l'édition de liens a eu lieu avant tout appel de méthode. Exemple :

```
public class maclasse
{
    public native double doubleNative(double valeur);

    static
    {
        System.loadLibrary("maclassejni");
    }

    // code java 'normal' ici ...
}
```

Le choix du nom de la librairie est du ressort du programmeur. Le nom de la librairie ne correspond pas au nom de fichier de la librairie du point de vue du système d'exploitation. Le nom du fichier doit être tel que le système d'exploitation (via la machine virtuelle) soit capable de charger la librairie. Dans notre exemple, nous nommons notre librairie 'maclassejni', cela signifie que :

- Sous Windows le fichier s'appellera 'maclassejni.dll'
- Sous Linux/UNIX le fichier s'appellera 'libmaclassejni.so'

Les problèmes

Travailler avec la JNI n'est pas difficile, c'est même relativement agréable, une fois que l'on sait résoudre tous les problèmes qui sont de deux types. Ceux qui concernent la JNI proprement dite. Ceux-là sont les plus faciles. Dans la suite de cet article, nous construirons un exemple qui devrait aplanir toutes les difficultés. Mais il y a aussi les problèmes liés au système d'exploitation sous-jacent et aux compilateurs C/C++ utilisés. Ces difficultés sont parfois fort agaçantes. Résolvons d'abord celles-là, sous Windows et sous Unix/Linux. Nous commençons donc en quelque sorte par la fin, en supposant que notre code est écrit et compilé sous la forme d'une librairie partagée.

Sous Windows

Lorsque la machine virtuelle charge une librairie, elle passe bien évidemment par le système d'exploitation. Si la recherche échoue, la JVM vous envoie une volée de bois vert sous la forme de la redoutable exception `java.lang.UnsatisfiedLinkError`. Par défaut, la JVM se repose sur l'OS sous-jacent et ne cherche rien toute seule. Autrement dit, si la librairie est visible par l'OS, cela fonctionnera. Sous Windows une dll est trouvée, si elle figure dans le répertoire de lancement de l'application, dans les répertoires Windows, à la racine du disque, si un chemin a été défini dans la base de registre, et enfin si elle est dans un répertoire pointé par la variable d'environnement PATH. Compter sur le répertoire courant, ou la base de registre est une très mauvaise idée en Java. Le mieux est de définir un répertoire destiné à contenir la ou les dlls et à faire pointer ce répertoire par le PATH. Une autre bonne possibilité (également sous Linux) est d'étendre la recherche de librairie de la JVM, en lançant celle-ci depuis un script qui contiendra une ligne dans cet esprit :

```
java -Djava.library.path=repertoire MaClasseJni
```

Enfin si vous venez de modifier votre PATH depuis le panneau de configuration, n'oubliez pas que vous devez fermer et rouvrir votre console...

Toutes ces opérations effectuées, cela ne marche toujours pas ? Si vous regardez bien, vous voyez que la même exception est levée, mais que le message d'erreur est légèrement différent, car il ne concerne plus la librairie dans son entier, mais seulement une fonction (la première invoquée). Voici ce qui se passe. Automatiquement, les fonctions de votre dll obéissent à une convention d'appel `__stdcall`. Vous ne pouvez pas agir sur ceci, car la JVM compte sur cette convention d'appel. Cela ne se voit pas directement, mais cette convention d'appel est imposée dans le fichier généré par l'outil `javah`, dont nous parlerons plus loin. Si vous avez l'intention de mettre un peu de code assembleur dans vos routines, sachez que la convention d'appel `__stdcall` consiste à pousser les paramètres de droite à gauche sur la pile (comme en C avec `cdecl`) et que c'est l'appelé, votre fonction donc, qui doit restaurer la pile, comme en convention Pascal. En C/C++ le compilateur se charge bien évidemment de ça et vous n'avez pas à vous en occuper. En revanche, ce que vous devez savoir, est que la convention d'appel `__stdcall` va en outre induire une décoration des noms de vos fonctions. Cette décoration consiste en l'ajout du caractère underscore (`_`) comme préfixe, et en l'ajout du signe `@` suivi d'un nombre, comme suffixe. Le nombre correspond au nombre d'octets passés sur la pile (Figure 2). Prenons un exemple. Soit une fonction C :

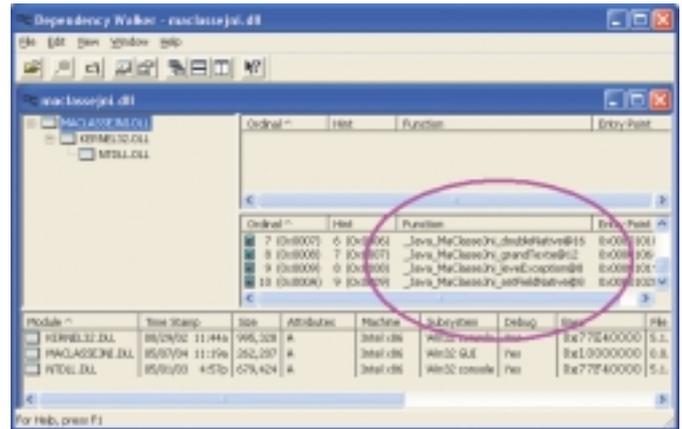


Figure 2 : Attention la convention d'appel `__stdcall` implique une décoration des noms de fonction.

```
int mafonction(int a, double b);
```

le nom `mafonction` sera décoré comme suit :

```
_mafonction@12
```

Mais vous l'avez deviné, la JVM, pas au courant de tout ça, va chercher à charger la fonction 'mafonction' dans la librairie et bien sûr, ne va pas la trouver.

Vous devez donc intervenir sur votre compilateur/éditeur de liens, pour remédier à la situation. En principe, les outils Borland ne décorent pas, ce qui est aimable à eux. En revanche, avec Visual C++ ou Mingw (gcc version Windows) c'est différent. Avec Visual, vous devez créer un fichier `.def` dans votre projet. Il s'agit d'un simple fichier texte, qui contiendra des directives pour poser des alias dans la librairie. Pour l'exemple ci-dessus, le fichier `.def` serait constitué de deux lignes :

```
EXPORTS
    mafonction=_mafonction@12
```

Bien sûr Visual C++ doit tenir compte de ce fichier. C'est a priori automatique, si le fichier a été édité sous Visual. Sinon donnez une commande supplémentaire à l'éditeur de liens :

```
/def:".\fichier.def"
```

Si vous travaillez avec Mingw, donnez une commande supplémentaire à l'éditeur de liens :

```
--add-stdcall-alias
```

et tout devrait fonctionner normalement.

Sous Linux

De même que sous Windows, la librairie doit être vue par l'OS. Ainsi, dans sa documentation de la JDK, Sun déclare qu'un nom de fichier, en accord avec les règles du système d'exploitation cible, suffit pour assurer le chargement de la librairie. En fait, l'expérience montre que ceci

n'est pas exact. Par exemple, sous Linux, les répertoires contenant des bibliothèques partagées sont énumérés dans le fichier de configuration `ld.so.conf`. On pourrait penser que placer la bibliothèque dans un de ces répertoires suffit pour en assurer le chargement. Ou encore, en ajoutant une entrée dans `ld.so.conf` et en faisant relire le fichier au moyen de `ldconfig`. En fait, les choses vont différemment. Il faut que le répertoire contenant la bibliothèque soit identifié dans une variable d'environnement exportée et nommée `LD_LIBRARY_PATH`. Si vous voulez être sûr, lors du premier essai saisissez dans la console :

```
export LD_LIBRARY_PATH=/repertoire && java MaClasseJni
```

Enfin, n'oubliez pas que Linux est sensible à la casse, alors que ce n'est pas le cas de Windows. `Javah` génère les fichiers en-têtes à partir des noms de fichiers Java. Ainsi, un `MaClasseJni.java` donnera un `MaClasseJni.h`. Dans votre code C++ veillez à bien donner `#include "MaClasseJni.h"` et non `#include "maclassejni.h"`.

Le lecteur trouvera un `makefile` d'exemple pour Linux sur le Cd-Rom accompagnant le magazine.

Savoir s'entêter

Enfin pour en terminer avec ces lourdes formalités, votre compilateur doit avoir accès aux fichiers en-têtes de la JDK, relatifs à la JNI. Ces fichiers en-têtes sont dans le répertoire `include` de votre JDK. Attention, des en-têtes spécifiques à l'OS sont dans un sous-répertoire `include`. Informez également votre compilateur de l'existence de ce sous-répertoire.

Passons maintenant au code. Voici notre classe d'exemple :

```
public class MaClasseJni
{
    private native void setFieldNative();
    private native String grandTexte(String texte);
    private native double doubleNative(double valeur);
    public native void lance_exception() throws ClassNotFoundException;
    public native double division(double dividende, double diviseur);

    // etc...

    private int field;
}
```

- `setFieldNative` montre comment modifier une variable de classe depuis une méthode native.
- `grandTexte` illustre le travail 'à la chaîne'. La méthode reçoit un objet `String` et en retourne un autre.
- `doubleNative`. Nous en avons déjà parlé. Mais afin de varier les plaisirs, nous verrons comment appeler une méthode de classe Java depuis une méthode native.
- `lance_exception` illustre la manière de lancer une exception Java depuis du code natif.
- `division` montre comment lancer une exception personnalisée depuis du code natif. Lorsqu'il travaille avec des doubles, Java accepte de diviser par 0.0 et retourne la valeur 'Infinity'. Jugeant cette valeur peu pratique à manipuler, nous préférons lever une exception :

La JDK vient avec un outil qui génère, à partir du byte-code Java, des en-têtes C/C++ contenant des prototypes de fonction. Cet outil se nomme 'javah'. La première opération consiste donc à traiter le byte-code Java (et non le code source) avec lui :

```
javah MaClasseJni (et non javah MaClasseJni.class)
```

Ce compilateur génère un fichier en-tête `.h` directement utilisable. Jetez un coup d'oeil dans ce fichier est instructif. D'abord, on y voit que tous les noms de méthodes natives ont été modifiés, selon le schéma immuable :

```
Java_classe_methode
```

Si la classe fait partie d'un package tel que `org.monpackage`, le nom de la fonction deviendra :

```
Java_org_monpackage_classe_methode
```

On voit également que les méthodes ont des signatures. Une signature de méthode est une petite chaîne de caractères décrivant les types des paramètres reçus et le type de la valeur renvoyée par la méthode (Figure 3). Comprendre les signatures devient utile lorsqu'il s'agit d'accéder à une variable ou une méthode de classe Java depuis C++.

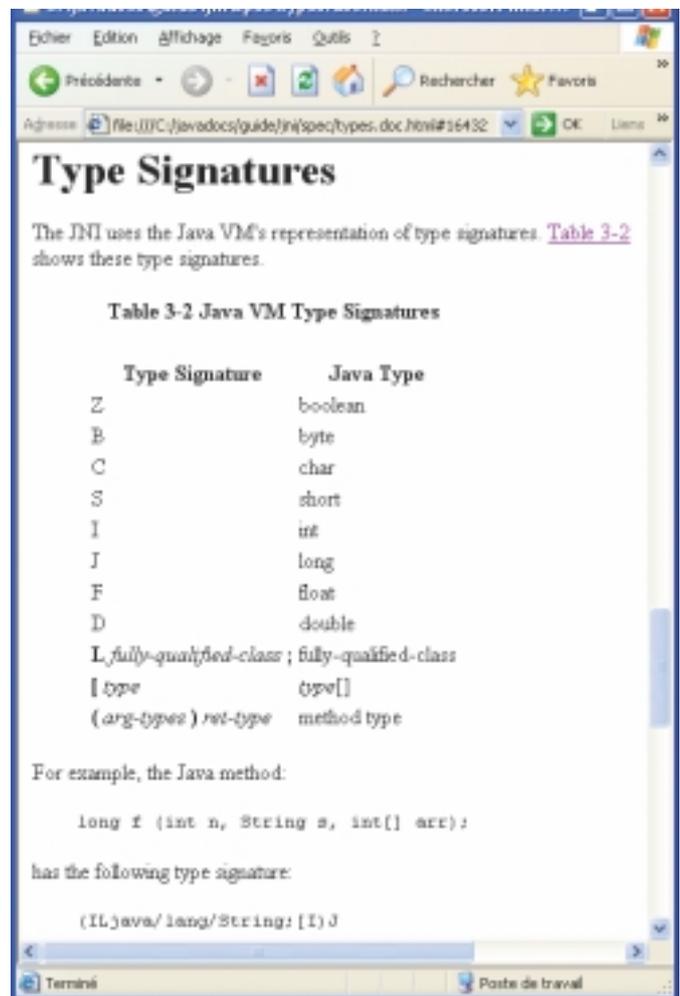


Figure 3 : Les signatures de méthodes et de membres permettent d'accéder aux entités Java depuis le code C++ et sont documentées dans le Javadoc.

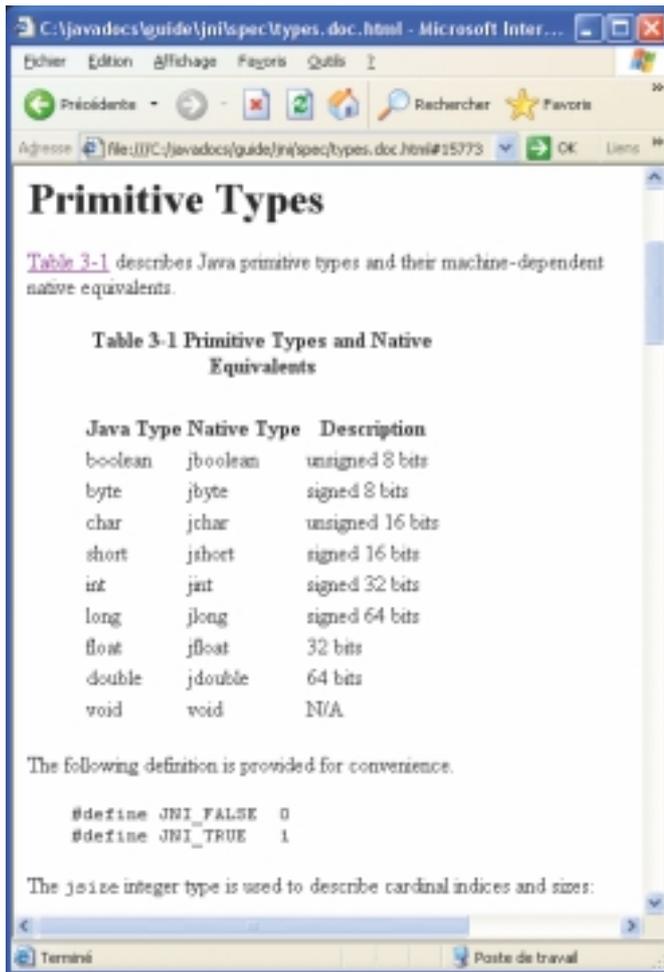


Figure 4 : Les types natifs vus pas la JVM

Sont enfin dignes d'intérêt les paramètres que la JVM envoie aux méthodes natives. Les types Java sont automatiquement convertis en types natifs, dits primitifs. Aucune difficulté dans cette partie. (Figure4)

Deux autres paramètres apparaissent dans les prototypes et méritent toute notre attention :

- jobject obj est un pointeur sur la classe Java propriétaire de la méthode native appelée. Disons que c'est la même chose qu'un pointeur this.
- JNIEnv *env est le pointeur sur la fameuse interface JNI. Ce pointeur sert à appeler les très nombreuses méthodes de l'interface. Rien de plus simple que d'appeler une méthode de la JNI depuis C++:

```
env->MethodeJni();
```

Depuis C la syntaxe est un peu plus lourde :

```
(*env)->MethodeJni();
```

Il ne nous reste plus qu'à implémenter nos fonctions prototypées. Voyons quelques éléments du programme d'exemple.

Modifier une variable de classe

Notre classe d'exemple comporte un champ privé nommé field. Pour pouvoir accéder à un champ il faut savoir à quelle classe nous avons affaire. Ceci se fait avec la méthode `GetObjectClass` qui retrouve la définition de la classe à partir du pointeur 'this'. Ensuite il faut récupérer le champ, grâce à la méthode `GetFieldID`. On remarquera, qu'entre autres paramètres, cette méthode attend une signature. En effet, tout comme les méthodes de classes les membres de classes ont une signature, constituée d'ailleurs de la même façon. A ce stade, nous avons accès au champ seulement. Il importe de bien comprendre que nous n'avons pas encore obtenu la valeur du champ proprement dite. Pour obtenir ou affecter une valeur de champ, il faut employer les méthodes :

```
Get<type>Field()
Set<type>Field()
```

Où <type> correspond au type du champ en question. Pour notre exemple, le champ contient un entier, donc les méthodes à employer sont :

```
GetIntField()
SetIntField()
```

Le programme d'exemple utilise la deuxième méthode pour affecter la valeur 1 au champ field.

Travail à la chaîne

La méthode `grandTexte` du programme d'exemple illustre le travail avec les chaînes Java. Depuis la classe Java on passe une String à la fonction native. Celle-ci reçoit alors une jstring qui reste une chaîne Java. Cela veut dire que nous avons cette fois affaire à un objet et non à une valeur primitive. Donc, la chaîne encapsulée dans cet objet est en Unicode, comme toute chaîne Java.

Pour les besoins de la démonstration nous supposons que notre compilateur C++ ne connaît pas les `wstring` de la bibliothèque standard. Nous devons donc récupérer la chaîne de caractères sous la forme ASCII traditionnelle, modifier la chaîne, puis le transformer à nouveau en une chaîne Unicode, avant de la retourner à la fonction appelante dans le programme Java.

Si vous jetez un oeil sur le programme d'exemple, vous verrez que ces opérations sont bien effectuées. Vous y verrez aussi une chose supplémentaire très importante. Globalement, le programmeur qui travaille avec la JNI n'a pas plus à se préoccuper de la gestion de la mémoire qu'il ne doit le faire avec du Java pur. Le ramasse-miettes s'occupe normalement de tout. Il y a quand même quelques exceptions auxquelles il faut être attentif. Lorsque nous appelons la méthode `GetStringUTFChars`, cela a pour effet d'interdire au Garbage Collector de Java de disposer de la mémoire occupée par l'objet chaîne. Il faut qu'il en soit ainsi, sinon nous ne pourrions pas nous fier au pointeur de caractères retourné par la méthode. En contre-partie, pour éviter une fuite mémoire, nous devons dans ce cas nous préoccuper de la libération de l'objet. ceci se fait, au moyen de la méthode `ReleaseStringUTFChars`. Le reste de la démonstration, à savoir appel de code Java depuis le code C++ et levée d'exception, obéit aux principes déjà vus. Nous approfondirons le travail avec la JNI ultérieurement.

■ Frédéric Mazué - fmazue@programmez.com



C# 2.0 : Les évolutions du langage

Avec le lancement de Mono 1.0, les développeurs C# peuvent déjà tester gratuitement certaines nouvelles possibilités du langage, qui seront également incluses dans la prochaine mouture de Visual Studio 2005, baptisée "Whidbey". Examinons ensemble de quoi il s'agit.

La première spécification du langage C# date de décembre 2001 (document ECMA-334). En avril 2003, une version révisée de cette spécification voit le jour et est ratifiée par l'ISO (ISO/IEC 23270). En juillet 2003 Microsoft publie un "draft" présentant les nouvelles fonctionnalités de C# ("C# Version 2.0 Specification"). A court terme, Novell (Mono), Microsoft, Borland et d'autres éditeurs suivront tous le même mouvement. Ce C# deuxième mouture comprendra les itérateurs, les méthodes anonymes, la généricité, etc. Microsoft promet que ces extensions au langage seront implémentées de manière à conserver une compatibilité maximale avec le code existant. Par exemple, les nouveaux mots-clés "yield" et "partial" pourront (selon le contexte) être conservés comme identifiants.

Les itérateurs

Souvent, les développeurs doivent créer des classes supportant l'énumération. En C# vous disposez du mot clé foreach pour parcourir les éléments d'une collection. Pour pouvoir être parcourue, une collection doit comporter une méthode "GetEnumerator" qui retourne un énumérateur (en utilisant les interfaces System.Collections.IEnumerable et System.Collections.IEnumerator). L'interface IEnumerable expose l'énumérateur, qui prend en charge une itération simple sur une collection. Voici un exemple de la manière dont un programmeur C# doit opérer actuellement :

```
class Customers : System.Collections.IEnumerable, System.Collections.IEnumerator
{
    System.Collections.ArrayList customerIDs;
    System.Collections.IEnumerator ienum;

    public void SortByName()
    {
        this.Reset();
    }
    public void SortByID()
    {
        this.Reset();
    }
    public Customers()
    {
        ienum = customerIDs.GetEnumerator();
    }
}
```

```
public System.Collections.IEnumerator GetEnumerator()
{
    return (System.Collections.IEnumerator)this;
}
public void Reset()
{
    ienum.Reset();
}
public bool MoveNext()
{
    return ienum.MoveNext();
}
public object Current
{
    get
    {
        return new Customer(Convert.ToInt32(ienum.Current));
    }
}
}
```

Comme on le voit, un tel système est parfois assez complexe à implémenter en C# 1.0, tandis que les itérateurs de C# 2 faciliteront cette tâche. Bonne nouvelle : vous pouvez déjà tester ces itérateurs avec "gmcs" la preview du compilateur C# 2 de Mono.

C'est le mot clé "yield return" qui spécifie à l'itérateur de retourner la valeur suivante de l'itération. L'interface utilisée sera "System.Collections.IEnumerator". On parle bien "d'interface", car il s'agit d'indiquer les méthodes minimales implémentées par une classe (une classe peut implémenter plusieurs interfaces, mais le comportement du code n'est pas défini dans l'interface, mais bien dans la classe qui l'implémente).

Commençons par présenter un exemple d'itérateur facilement compréhensible :

```
using System;
using System.Collections;

class X {
    static IEnumerator GetIt ()
    {
        yield return 1;
        yield return 2;
    }
}
```

```

        yield return 3;
    }

    static int Main ()
    {
        IEnumerator e = GetIt ();
        int total = 0;

        while (e.MoveNext()){
            Console.WriteLine ("Value=" + e.Current);
            total += (int) e.Current;
        }

        return 0;
    }
}
// gmcs exemple1.cs
// mono exemple1.exe

```

Ce qui produit l'affichage :

```

Value=1
Value=2
Value=3

```

Comme vous pouvez le constater, c'est le mot clé "yield return" qui pilote le retour d'une valeur (du côté "fournisseur"). L'avancement du pointeur est commandé par e.MoveNext(), tandis que la valeur courante est retournée par e.Current (du côté consommateur). Toute la puissance de ce code repose sur le fait que l'objet IEnumerator "fournisseur" se concentre sur la manière de fournir l'information, tandis que le code "consommateur" s'articule sur la manière de "digérer", de "consommer" ces informations.

Les possibilités sont multiples : vous pouvez par exemple utiliser une itérative du côté du "fournisseur", pour piloter le retour d'une série de valeurs, comme le réalise le code suivant :

```

using System;
using System.Collections;

class X {
    static int start, end;
    static int i;

    static IEnumerator GetRange ()
    {
        yield return 1;
        for (i = start; i < end; i++)
            yield return i;
        yield return 100;
    }

    static int Main ()
    {
        start = 10;
        end = 30;
    }
}

```

```

int total = 0;

IEnumerator e = GetRange ();
while (e.MoveNext()){
    Console.WriteLine ("Value=" + e.Current);
    total += (int) e.Current;
}

return 0;
}
}

Ce qui produit l'affichage :
Value=1
Value=10
...
Value=de 11 à 28
..
Value=29
Value=100

```

Notez aussi que le mot clé "yield break" peut-être utilisé pour indiquer qu'une itération est complète.

Les méthodes anonymes

Une méthode anonyme autorise un programmeur à créer un bloc de code qui peut être encapsulé dans un délégué, pour permettre une exécution ultérieure. Entre parenthèses, le concept de méthode anonyme remonte aux fonctions dites "lambda" de Lisp.

Un délégué représente un objet qui référence à son tour une méthode (une sorte de "pointeur" vers la méthode, quoiqu'en C#, et en plus dans un contexte d'objets, parler de pointeur est une hérésie). Lorsque le délégué est invoqué, la méthode qu'il référence est appelée.

Prenons un exemple très simple d'utilisation d'un délégué : au sein d'un formulaire, vous disposez d'un bouton de commande ; lorsque nous initialisons ce bouton, nous donnons comme instruction que l'événement Click pointe vers la méthode Addclick.

```

class InputForm: Form
{
    ListBox listBox;
    TextBox textBox;
    Button addButton;
    public MyForm() {
        listBox = new ListBox(...);
        textBox = new TextBox(...);
        addButton = new Button(...);
        addButton.Click += new EventHandler(AddClick);
    }
    void AddClick(object sender, EventArgs e) {
        listBox.Items.Add(textBox.Text);
    }
}

```

Avec une méthode anonyme, vous créez un délégué dont le code suit

directement (mais il n'y a pas de nom qui désigne la nouvelle méthode qui est bien "anonyme").

```
class InputForm: Form
{
    ListBox listBox;
    TextBox textBox;
    Button addButton;
    public MyForm() {
        listBox = new ListBox(...);
        textBox = new TextBox(...);
        addButton = new Button(...);
        addButton.Click += delegate {
            listBox.Items.Add(textBox.Text);
        };
    }
}
```

C'est la structure "delegate {};" qui indique que nous avons affaire à une méthode anonyme. Notez l'accolade finale, qui est suivie d'un point-virgule terminant le code "inline". La liste des paramètres est optionnelle. Dans la même veine, nous pourrions indiquer :

```
addButton.Click += delegate(object sender, EventArgs e) {
    MessageBox.Show(((Button)sender).Text);
};
```

La définition de ce délégué comporte deux paramètres "sender" et "e", ce qui correspond à la définition (object sender, EventArgs e) d'AddClick. Lorsque l'événement se produit, la méthode anonyme est appelée, et les deux paramètres sont effectivement passés comme arguments.

C'est le compilateur C# qui convertit automatiquement le code entre accolades en une référence unique. Malheureusement, à l'heure où nous rédigeons ces lignes, le compilateur gmcs de Mono (preview du futur compilateur C# 2) n'implémente pas encore les méthodes anonymes (qui sont testées pour l'instant par une poignée de développeurs dans une branche à part).

Les classes de type Partial

Cette nouvelle possibilité permettra de "splitter" une même classe en plusieurs codes sources ".cs" ce qui, dans certains cas, en facilitera la maintenance. Ainsi MaClasse1.cs et MaClasse2.cs...

```
Maclasse1.cs
public partial class Customer
{
    private int id;
    private string name;
    private string address;
    private List<Order> orders;
    public Customer() {
        ...
    }
}
Maclasse2.cs
public partial class Customer
{
```

```
public void SubmitOrder(Order order) {
    orders.Add(order);
}
public bool HasOutstandingOrders() {
    return orders.Count > 0;
}
}
...fusionneront pour donner ceci :
```

```
public class Customer
{
    private int id;
    private string name;
    private string address;
    private List<Order> orders;
    public Customer() {
        ...
    }
    public void SubmitOrder(Order order) {
        orders.Add(order);
    }
    public bool HasOutstandingOrders() {
        return orders.Count > 0;
    }
}
```

C'est à nouveau le compilateur qui se chargera de rapprocher l'ensemble au sein d'une même classe.

La généricité

C'est évidemment le changement majeur attendu. Nous n'allons qu'effleurer le sujet, qui est vaste et complexe, avec des notions, comme les délégués génériques, les contraintes de type, la surcharge générique, etc. Grossièrement exprimé, par "généricité", on entend le pouvoir pour un objet d'être utilisé tel quel dans différents contextes. Ce n'est qu'au dernier moment que le programmeur définit le contexte. Prenons un premier exemple simple, mais parlant :

```
using System;

public class Stack<S>
{
    public Stack (S s)
    {Console.WriteLine(s);}

    public void Push (S s)
    {Console.WriteLine(s);}
}

public class X
{
    static void Main ()
    {
        Stack<int> s1 = new Stack<int> (3);
        s1.Push (4);
    }
}
```

```
Stack<string> s2 = new Stack<string> ("Hello");
s2.Push ("Test");
}
}
```

La pile "Stack" sera, soit de type entier (Stack<int>), soit de type chaîne (Stack<string>). Les méthodes Stack (constructeur) et Push seront "redéfinies dans ce contexte" pour pouvoir traiter, soit des entiers, soit des chaînes de caractères. La généricité permet ici d'utiliser une même méthode avec des arguments de type différent.

Parfois, il est possible de ne pas déclarer explicitement le type, c'est ce que l'on appelle en anglais le "type inference". Le compilateur redéfinira de sa propre initiative la méthode générique Choose, selon le type des arguments passés lors de l'appel de la méthode :

```
using System;

class Util
{
    static Random rand = new Random();

    static public T Choose<T>(T first, T second) {
        return (rand.Next(2) == 0)? first: second;
    }
}

class Application
{
    static void Main() {
        int i = Util.Choose(5, 213);
        Console.WriteLine(i);
        string s = Util.Choose("foo", "bar");
        Console.WriteLine(s);
    }
}
```

L'affichage produit sera, soit "5,foo" ou "5,bar" ou "213,foo" ou "213,bar".

Enfin, voici un dernier exemple, qui comprend une classe générique utilisant une variable de type statique (static int count), ce qui produit l'affichage "1", "1" "2" :

```
using System;

class C<V>
{
    static int count = 0;
```

```
public C() {
    count++;
}

public static int Count {
    get { return count; }
}
}

class Application
{
    static void Main() {
        C<int> x1 = new C<int>();
        Console.WriteLine(C<int>.Count);

        C<double> x2 = new C<double>();
        Console.WriteLine(C<int>.Count);

        C<int> x3 = new C<int>();
        Console.WriteLine(C<int>.Count);
    }
}
```

Ce mécanisme est le bienvenu, car il fera gagner énormément de temps au développeur C# et en outre, produira du code plus solide, en incluant éventuellement des contraintes supplémentaires de vérifications de types.

En conclusion

Nous avons envie de dire que, si au départ le C# était fort simple, ce qui se prépare actuellement en coulisse augure un C# dont la complexité n'aura rien à envier au C++. Pourtant, au départ, un des arguments en faveur du C# était justement qu'il simplifiait la programmation par rapport au C++ pour rivaliser avec la facilité de prise en main du bon vieux Visual Basic...

Évidemment, ce n'est pas le programmeur C# chevronné qui s'en plaindra, bien au contraire. Mais les limites entre C# et C++ vont devenir plus floues...

Finalement, le C# est peu créatif, et on peut se demander pourquoi il n'emprunte pas plus à la programmation fonctionnelle pour se démarquer du lot. Pour laisser une place au soleil au futur F# ?

Spécifications officielles ECMA du C# :

<http://www.ecma-international.org/publications/files/ecma-st/ECMA-334.pdf>

■ Xavier.Leclercq@programmez.com

✓ *L'actualité en ligne*

www.programmez.com

✓ *Abonnez-vous en ligne*

www.programmez.com/abonnement



A la découverte de Jython

Jython est un interpréteur Python, écrit en Java. Une association de deux langages qui permet d'exploiter le meilleur de deux mondes pleins de richesses.

En 1990 est né Python. Un langage de script Open Source vraiment remarquable qui allie avec un rare bonheur la clarté et l'élégance de sa syntaxe avec une puissance peu commune : types de haut niveau (listes, tuples, dictionnaires), typage dynamique, orientation objet, et surcharge des opérateurs, etc... A cela, s'ajoute une bibliothèque de modules d'extension extrêmement riche. Python n'est soutenu par aucune campagne de marketing et pourtant il séduit toujours plus de programmeurs par son efficacité.

Guido van Rossum, le créateur de Python, a choisi le langage C pour écrire l'interpréteur de Python, afin d'assurer la meilleure performance à l'exécution et aussi la portabilité la plus large. Ainsi, toute plate-forme disposant d'un compilateur C dispose de Python ipso facto.

L'avènement de Java

Entre le moment où Guido a créé Python et celui où vous lisez ces lignes, un autre événement majeur s'est produit dans le monde informatique : l'apparition de Java. Java est un langage orienté objet, facile à manier et qui se veut largement portable, même s'il l'est moins que Python pour l'instant. Les points forts de Java, à savoir et pour ne citer que ceux-là, la sécurité, l'orientation objet, le ramasse-miettes et la richesse de Swing, ont séduit nombre de 'pythoneurs'. Ceux-ci ont donc eu l'idée de marier les deux langages, avec l'intention de tirer le meilleur de chacun. Deux tentatives ont d'abord été faites. La première fut d'interfacer Python avec Java via la Jni (Java Native Interface) de celui-ci. La seconde tentative consista en une traduction du code-byte Python en code-byte Java. Ces deux tentatives n'ont rien donné. La première était trop complexe, la seconde trop peu performante.

Qu'est-ce que Jython ?

Finalement, Jim Hugunin a employé les grands moyens et écrit un interpréteur Python complet en pur Java. Initialement baptisé JPython, le langage évolue à la suite de Python, quoique moins rapidement. Au cours de son évolution, le langage a été rebaptisé Jython. Jython est Open Source et vous pouvez le télécharger à <http://www.jython.org>. La popularité de Jython va croissant, et des livres commencent à lui être consacrés. En effet, Jython présente de nombreux atouts séduisants de par son caractère hybride. Tout d'abord, la sécurité. Celle-ci étant apportée naturellement par la JVM. Une gestion de la mémoire éprouvée, car c'est évidemment le ramasse-miettes de la JVM qui s'acquitte de cette tâche. A l'époque, c'était un progrès intéressant sur le mécanisme de comptage de références de Python. Désormais, depuis la version 2.0, le Python écrit en C dispose aussi d'un ramasse-miettes. En revanche, la présence du ramasse-miettes impose aux pythoneurs purs et durs quelques contraintes mineures, dont nous parlerons plus loin. Un atout majeur de Jython est son éventail de bibliothèques potentielles, car tout package Java peut être importé dans un script Python, et les classes Java peuvent même être dérivées en classe Jython. Python est connu par le nombre impressionnant de ses modules d'extensions, mais Jython n'est pas en reste, tant l'univers Java est maintenant immense. Il découle de ceci,

que Swing est le GUI natif de Jython. Autre avantage, Jython peut être embarqué dans une application Java. Autrement dit, un programme Java peut exposer un interpréteur à l'utilisateur final. Imaginons, par exemple, un programme de représentation de fonctions mathématiques. Grâce à la présence d'un interpréteur, il y serait facile de traiter une définition de fonction, saisie par l'utilisateur lors de l'exécution.

Bien évidemment, Java n'est pas le seul à amener des points forts. Python apporte sa syntaxe claire et l'extrême rapidité de développement qui découle, entre autres, de ses types de haut niveau. Tout ce qui est fastidieux à faire depuis une application Java, par exemple une analyse de fichiers texte, devient trivial avec un script Jython. Enfin il y a une cerise sur le gâteau : un script Jython peut, grâce au compilateur jythonc, être compilé en une classe Java, ou même en une archive jar. Après quoi, du code Java peut employer la classe, exactement comme elle le ferait d'une classe java normale et de façon totalement transparente. Nous allons maintenant aborder tous ces aspects d'un point de vue pratique. Le lecteur qui n'aurait aucune notion de Python ne devrait pas quitter cet article maintenant. Python (et donc Jython) est le langage le plus rapide à apprendre qui soit. Une ou deux après-midi suffisent.

Installation et prise en mains

Jython vient sous la forme d'une classe-archive auto extractible. Tout ce que vous avez à faire est de lancer la classe, comme un programme Java ordinaire, en vous assurant que le répertoire courant est dans le CLASSPATH. par exemple :

```
java -cp . jython-2.1
```

Après quoi, l'installation est automatique, une fois que vous avez spécifié un emplacement (Figure : 1). L'installation effectuée, faites pointer le PATH de votre système sur le répertoire d'installation. Pour vérifier que tout est en ordre, ouvrez une console et lancez Jython. Vous retrouverez alors dans l'interpréteur de commandes interactif (Figure : 2) que vous pouvez quitter en saisissant Ctrl-D (Unix/Linux) ou Ctrl-Z (Windows).



Figure 1 : Jython en cours d'installation.

Un coucou hybride

Sacrifions à la tradition et écrivons un programme de type 'Hello world'. Puisque nous savons que nous pouvons utiliser Swing, nous allons le

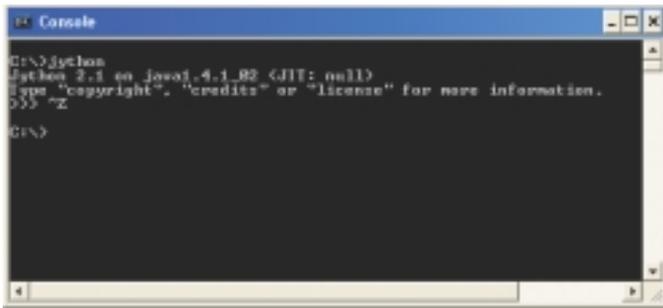


Figure 2 : L'interpréteur Jython interactif.

faire. Nous voulons donc une application fenêtrée, dotée d'un bouton qui terminera le programme au premier clic. Commençons par écrire le code comme le ferait un codeur Java. Cela donne le résultat (voir



Figure 3 : Hello world à la mode Jython.

Encadré 1). Le code de tous les exemples, et même plus, se trouve sur le Cd-Rom accompagnant le magazine. Le code est tout simple à comprendre. Nous importons tout d'abord les packages nécessaires. Ensuite nous écrivons une classe Jython qui dérive d'une classe awt, afin d'avoir un écouteur d'événement. Nous pouvons commencer à apprécier la coopération des deux langages. Le reste de l'application est trivial. Le programme (Figure : 3) se lance tout simplement par :

```
jython coucouhybride.py
```

Remarquons que si vous utilisez une JVM antédiluviennne, Swing y est peut-être localisé à un endroit nébuleux (cas des JVM 1.1 par exemple). Dans ce cas, utiliser le module spécial Jython pawt, comme montré en commentaire dans le code d'exemple.

Tout ceci est fort bien. Un programmeur Java y trouvera déjà largement son compte, mais un pythoneur s'attendra à juste titre à une syntaxe plus concise et il ne sera pas déçu. Quand Jython importe une classe, il use et abuse du mécanisme de réflexion et constitue un dictionnaire de propriétés. Ces propriétés peuvent être initialisées, via des arguments mots-clés transmis au constructeur, ou encore directement, car des accesseurs sont également générés automatiquement. En outre, une fonction Jython est une fonction objet. Du coup, plus besoin de s'em-bêter avec les classes écouteurs d'événements. Nous réécrivons le code (Encadré 2) Nous avons gagné en concision et en simplicité. Remarquez que les packages awt relatifs aux événements ne sont même plus importés...

Comme mentionné plus haut, des accesseurs sont créés et opèrent en sous-main. Nous aurions pu encore écrire :

```
fenetre = JFrame('Exemple Jython')
# etc, etc
fenetre.visible = 1
```

Nous avons déjà apprécié le typage dynamique de Jython avec notre fonction objet. Ce typage dynamique est présent partout. Supposons qu'en pur Java nous voulions définir une zone de texte avec un contenu et une dimension préférée. (Une dimension préférée est requise si

l'on veut que le composant soit visible dans un JScrollPane). Ceci représente beaucoup de lignes de code à écrire. Par exemple, nous devons instancier un objet 'Dimension'.

En Jython, une seule ligne de code suffit :

```
zone = JTextArea("du texte ici", visible = 1, preferredSize = (200, 100))
```

Remarquez le tuple (200, 100). Le mécanisme de typage dynamique transforme le tuple en un objet 'Dimension'. Sympathique non ?

Attention toutefois

La présence du ramasse-miettes de Java impose de coder avec précaution. Examinons ce code Python :

```
def fonction():
    fichier = open('nom_de_fichier')
    # faire quelque chose
    # puis quitter la portée
```

Ce code Python est correct. Grâce au mécanisme de comptage des références, la fermeture du fichier est assurée dès que l'on sort de la portée de la fonction, si le compteur de références est à zéro. Dans un contexte Java, les choses sont différentes, car on ne sait jamais quand le ramasse-miettes va détruire effectivement les objets qui doivent l'être. Ce qui est certain, si l'on reprend l'exemple précédent, c'est que le fichier ne sera pas fermé dès la sortie de la portée de la fonction. Le programmeur doit donc veiller à ces particularités et éventuellement écrire un code comme :

```
def fonction():
    fichier = open('nom_de_fichier')
    # faire quelque chose
    fichier.close()      # fermer explicitement le fichier
                        # puis quitter la portée
```

Des composants Jython pour votre JVM

Jython est un outil précieux pour développer très rapidement des composants complexes destinés à être intégrés dans une application Java. Le code ci-contre (Encadré 3) est largement inspiré d'un code écrit par



Figure 4 : Un tableau de bord de calculatrice implémenté en Jython.

David Ascher, un des membres de l'équipe de développement de Jython, code lui-même calqué sur un code écrit par Guido van Rossum pour Python et Tkinter ;) Il s'agit d'un composant étendant JPanel. Nous constituons le tableau de bord d'une calculatrice simple (Figure : 4). Nous voyons que Jython permet l'utilisation des layouts et que nous remplissons le nôtre avec une simple itération dans une liste. La mise en place des gestionnaires d'événements est tout aussi simple que précédemment. Enfin, lorsque l'utilisateur demande le résultat de l'opération, nous n'analysons surtout pas le contenu de la zone de texte, mais nous demandons plutôt à l'interpréteur d'évaluer la chaîne. Si nous devions faire la même chose en pur

Java, cela nous entraînerait beaucoup plus loin.

A ce point, nous souhaitons incorporer notre panel comme un vulgaire composant Swing (Figure : 5). Le code (voir l'Encadré 4) fait ce travail. Nous importons notre composant comme tout composant Swing. Le code considère que le composant fait partie d'un package.



Figure 5 : Notre panneau est maintenant incorporé dans une application Java.

Compilation d'un script

Jython vient avec un utilitaire génial : jythonc. Jython compile les scripts en classes Java. Il travaille en deux temps. D'abord, il génère du code Java équivalent au script, puis il invoque javac. Si vous ne craignez pas les maux de tête jetez un coup d'oeil au code Java généré. C'est vraiment intéressant. Jythonc peut en outre construire une archive jar contenant éventuellement jython lui-même (se reporter à la documentation), et sait placer les classes dans un package. Pour notre exemple nous donnons :

```
jythonc -jar Calc.jar -package org.fred.jython Calc.py
```

A la suite de quoi vous trouverez tous les fichiers intermédiaires (le code Java notamment) dans le sous répertoire jpywork et l'archive jar dans le répertoire courant. Pour que tout fonctionne, vous n'avez plus qu'à faire pointer votre classpath sur l'archive et aussi sur jython lui-même, qui figure lui aussi dans une archive jar que vous trouverez à la racine de

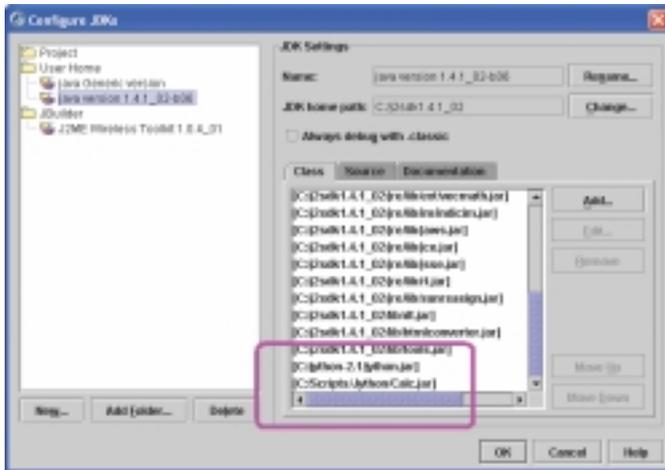


Figure 6 : Configuration de JBuilder pour qu'il voie Jython.

votre installation Jython. Le cas échéant, vous devrez peut-être aussi configurer votre environnement de développement Java, afin qu'il voit toutes ces classes, comme j'ai dû le faire pour mon JBuilder (Figure 6)

Un coucou embarqué

Enfin le dernier exemple (Encadré 5) démontre comment embarquer l'interpréteur Jython dans une application Java, comment déclarer une variable dans l'interpréteur ou en récupérer la valeur, ou encore comment charger un module Jython et invoquer une de ses méthodes. Pour ce dernier point, le fichier contenant le script doit être pointé par la

registry (se reporter à la documentation) de Jython qui est l'équivalent du PYTHONPATH de son grand frère.

Encadré 1

Hello World Jython écrit dans un style Java (fichier coucouhybride.py)

```
#!/usr/bin/env jython

from java.lang import System
from java import awt
from javax.swing import JFrame, JButton
#from pawt.swing import JFrame, JButton

class clic(awt.event.ActionListener):
    def actionPerformed(self, event):
        print "Bye à la mode Python"
        System.out.println("Bye à la mode Java")
        System.exit(0)

fenetre = JFrame("Exemple Jython")
bouton = JButton("Coucou hybride")
bouton.addActionListener(clic())
fenetre.contentPane.add(bouton)
fenetre.pack()
fenetre.setVisible(1)
```

Encadré 2

Hello World Jython écrit dans un style Python (fichier coucouhybride1.py)

```
#!/usr/bin/env jython

from java.lang import System
from javax.swing import JFrame, JButton

def quitter(e):
    print "Bye à la mode Python"
    System.out.println("Bye à la mode Java")
    System.exit(0)

fenetre = JFrame("Exemple Jython", visible = 1)
bouton = JButton("Coucou hybride", actionPerformed = quitter)
fenetre.contentPane.add(bouton)
fenetre.pack()
```

Encadré 3

Un panneau complexe écrit simplement :-)

```
#!/usr/bin/env jython

#from pawt.awt import *
#from pawt.swing import *
from pawt.swing import test
from java.awt import *
from javax.swing import JPanel, JTextField, JButton

labels = ['7', '8', '9', '+',
```

```
'4', '5', '6', '=',
'1', '2', '3', '*',
'0', '.', '=', '/', ]
```

```
class Calc(JPanel):
    def __init__(self):
        self.setLayout(BorderLayout())
        self.keys = JPanel(GridLayout(4, 4))
        self.display = JTextField()

        for label in labels:
            key = JButton(label)
            if label == '=':
                key.actionPerformed = self.enter
            else:
                key.actionPerformed = self.push
            self.keys.add(key)

        self.add(self.keys, 'Center')
        self.add(self.display, 'North')

    def push(self, event):
        self.display.replaceSelection(event.actionCommand)

    def enter(self, event):
        self.display.text = str(eval(self.display.text))
        self.display.selectAll()

if __name__ == '__main__':
    print 'coucou'
    calc = Calc()
    calc.setVisible(1)
    test(calc)
```

Encadré 4

Utilisation depuis Java d'un script Jython compilé
package org.programmez.fred;

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import org.fred.jython.*;

public class UseCalc extends JFrame
{
    public UseCalc()
    {
        super();
        JPanel inner = new JPanel();
        getContentPane().add(inner);
        inner.setLayout(new BorderLayout());
        Calc calc = new Calc();
```

```
calc.setVisible(true);
inner.add(calc, BorderLayout.CENTER);
JButton bouton = new JButton("Quitter");
bouton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        bouton_actionPerformed(e);
    }
});

inner.add(bouton, BorderLayout.SOUTH);
}

void bouton_actionPerformed(ActionEvent e)
{
    System.exit(0);
}

public static void main(String[] args)
{
    UseCalc use = new UseCalc();
    use.pack();
    use.setVisible(true);
}
}
```

Encadré 5

L'interpréteur Jython embarqué dans une application Java
package org.programmez.fred;

```
import org.python.util.PythonInterpreter;
import org.python.core.*;

public class JythonEmbedded {
    public static void main(String []args)
        throws PyException
    {
        PythonInterpreter jython =
            new PythonInterpreter();

        System.out.println("Coucou embarqué");
        jython.exec("import sys");
        jython.exec("print sys");

        jython.set("a", new PyInteger(42));
        jython.exec("print a");
        jython.exec("b = 2+2");
        PyObject b = jython.get("b");

        System.out.println("b: " + b);
        jython.exec("import Calc1");
        jython.exec("Calc1.run()");
    }
}
```

■ Frédéric Mazué - fmazue@programmez.com

Clubic.com : la référence pour l'informatique de loisir



Jerry Nieuviarts

Développé par une équipe de 8 passionnés, Clubic.com accueille aujourd'hui plus de 180.000 visiteurs chaque jour et est considéré

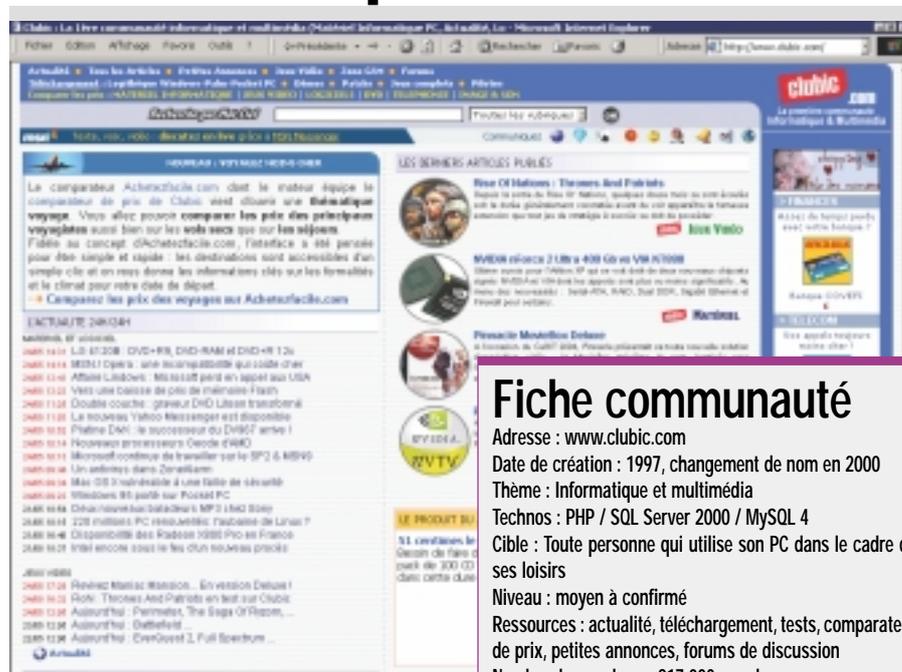
comme une référence de l'informatique de loisir. Rencontre avec Jerry Nieuviarts, le directeur.

Programmez : Peux-tu nous présenter brièvement Clubic.com ?

Clubic.com a été créé en 1997 comme un site perso sous le nom de Puissance PC. Il était alors hébergé par l'hébergeur gratuit associatif Mygale.org (devenu Multimania Lycos). Le site s'est depuis professionnalisé, grâce à une équipe permanente. Avec Clubic.com, nous souhaitons réunir sur un même support tous les contenus et services qui sont utiles, lorsque l'on utilise son ordinateur dans le cadre de ses loisirs.

Quel est le profil du membre type de Clubic ?

Au sein des visiteurs de Clubic.com, on trouve aussi bien des "power users" qui décortiquent les tests de produits dans les moindres détails, que des utilisateurs occasionnels qui se contenteront de lire la conclusion. Nous essayons de conserver cette pluralité de "lectures," aussi bien au niveau du contenu, que des services proposés sur le site. Dans le comparateur de prix, par exemple, l'utilisateur averti pourra se servir des nombreuses possibilités de filtrage sur les caractéristiques des produits, alors qu'un internaute moins expérimenté se contentera de la popularité des produits et des avis des autres visiteurs pour faire son choix.



Que trouve-t-on sur Clubic que l'on ne retrouve nulle part ailleurs ?

Des logiciels, des démos, des patches et des drivers en téléchargement direct depuis nos serveurs, avec un débit très important. Nos serveurs sont en effet connectés directement en très haut débit, aux réseaux des principaux FAI. Un comparateur de prix qui propose des filtres très évolués pour des produits pointus.

Quelle est l'architecture technique du site ?

Pour servir les pages web du site, nous utilisons 5 serveurs frontaux qui tournent sous

Fiche communauté

Adresse : www.clubic.com
 Date de création : 1997, changement de nom en 2000
 Thème : Informatique et multimédia
 Technos : PHP / SQL Server 2000 / MySQL 4
 Cible : Toute personne qui utilise son PC dans le cadre de ses loisirs
 Niveau : moyen à confirmé
 Ressources : actualité, téléchargement, tests, comparateur de prix, petites annonces, forums de discussion
 Nombre de membres : 217.000 membres
 Nombre de forums : 21
 Visiteurs : 2,2 millions de visiteurs uniques par mois
 Nombre de contributeurs actifs : 8 salariés à plein temps
 Nombre d'inscrits à la Newsletter : 80.000

Apache et qui interrogent des bases de données Microsoft SQL Server 2000 et MySQL 4.

Qui s'occupe du développement, avec quelles technos / langages ?

Tous les développements se font en interne. Nous avons débuté avec le couple Microsoft IIS / ASP sous Windows 2000 qui a très bien fonctionné au départ puis nous avons migré vers le couple Apache / PHP sous Linux, pour plus de souplesse au niveau de la répartition de la charge. Pour les bases de données, nous utilisons majoritairement SQL Server que nous complétons par MySQL pour des applications plus spécifiques. Pour agréger les pages ainsi générées à la volée, nous utilisons du Javascript et du DHTML.

Des projets pour la rentrée 2004 ?

Oui, bien sûr. Je ne vais pas en revanche être très original en vous disant qu'ils sont pour le moment confidentiels...

■ *Propos recueillis par David Thévenon*

Une communauté complète

Cadre supérieur passionné d'informatique, j'ai découvert Clubic il y a 4 ans, au hasard d'un moteur de recherche. J'ai trouvé en Clubic un site généraliste qui traite de l'actualité du monde du PC, par les news ou les articles, tout en donnant tous les outils pratiques pour faire fonctionner son matériel, à savoir le téléchargement d'utilitaires et des mises à jour des pilotes et logiciels. Je peux facilement choisir où acheter mon matériel grâce au comparateur de prix, et j'utilise régulièrement le service de petites annonces gratuites pour le matériel d'occasion. De plus, le forum permet de dialoguer de façon conviviale avec d'autres passionnés, de s'entraider pour résoudre des problèmes touchant au PC ou encore d'optimiser le fonctionnement de son ordinateur. Clubic offre une excellente ouverture sur le monde de l'informatique. Par contre, si je cherche des articles plus techniques je lui préfère généralement les sites spécialisés (en majeure partie anglo-saxons). ■ **Mathieu BERNARD**, modérateur du forum Clubic.com

Jeux vidéo : quels outils pour le développeur indépendant ?

Programmeur de jeux vidéo est devenu un métier prenant, passionnant et attractif. Et pourtant, la consécration historique de cette profession est récente.

Les premiers jeux commerciaux sont nés pendant " les années psychédélics ", à savoir les années 70, sous l'impulsion du visionnaire américain Nolan Bushnell, qui créa la société Atari. Les années 80 qui suivirent, consacrèrent l'avènement du jeu vidéo, avec l'hégémonie des sociétés nippones, comme Nintendo ou Sega. Mais les années 90 ont vu disparaître un grand nombre de studios à taille humaine. En parallèle, les éditeurs se sont centralisés, développés et ont parfois même fusionné, pour produire une série de blockbusters, impressionnants techniquement, mais disposant d'un budget pouvant faire pâlir certaines productions Hollywoodiennes.

La profession évolua, comme auparavant l'automobile et l'aviation. Il n'y a plus de place pour l'artisanat dans la conception d'un jeu vidéo. Plus question de développer un jeu vidéo sans un budget conséquent et une équipe complète et polyvalente, c'est-à-dire, composée de programmeurs spécialisés, mais aussi d'infographistes, de sound designers, de producteurs, de chefs de projets, etc. Irrémédiablement, ce métier changea. Pour certains, ce bouleversement eut pour conséquence de tuer la créativité et l'originalité, en industrialisant à l'extrême une profession. Cependant, avec la fin des années 90 et l'émergence d'Internet, une nouvelle forme d'artisanat du secteur est réapparue. Un artisanat nécessitant cependant une expérience et des compétences qui ne peuvent être improvisées, ni surfaites. Le concept " Internet " constitue aujourd'hui une alternative intéressante pour le programmeur de jeux vidéo indé-

pendant, en permettant la diffusion d'un logiciel à plus grande échelle. Faut-il pour autant considérer qu'il existe par ce biais une nouvelle alternative pour le développeur de jeux indépendant, ou pour le petit studio de développement ?

Les forces en présence

On ne présente plus DirectX et OpenGL

Pendant longtemps, l'architecture ouverte d'OpenGL fut la référence absolue des concepteurs de jeux. Sa caractéristique multi-plateforme, aussi bien portable sous Windows, Linux, Macintosh, voire même certaines consoles de jeux, ont fait d'OpenGL le numéro un, historiquement et techniquement, des programmeurs de jeux. Mais les choses sont rarement figées... (Figure 1)

Figure 1 : DirectX et OpenGL, les références pour la programmation de jeux vidéo

DirectX est la réponse de Microsoft, non pas à OpenGL comme certains le pensent, mais plutôt à Windows 95. En effet, l'agonie du système d'exploitation MS-DOS et la transition vers Windows 95 fut douloureuse pour les concep-



teurs de jeux vidéo sur PC. Linux n'existait pas véritablement encore en 1995, et Windows ne représentait pas un système d'exploitation dominant pour l'exploitation de jeux. Conscient de cette faiblesse, Microsoft mit en chantier un environnement de développement spécialisé, dont la première mouture se nommait originalement " Game SDK " pouvant être traduit littéralement par " Kit de développement logiciel de jeux ". Bien entendu, cette première version de ce qui devint plus tard DirectX, fut loin d'être aboutie et Microsoft dut revoir par deux fois sa copie, avant de présenter un environnement exploitable, avec la version 3.0 de DirectX.

DirectX 9.0 : Un seuil de maturité technologique

Neuf années se sont écoulées depuis sa création. Totalement intégré dans l'univers des jeux vidéo, du multimédia et également de l'Internet pour Windows (via le composant DirectPlay), DirectX reste plus que jamais la référence absolue pour le développement d'applications performantes, éblouissantes, fascinantes... mais également complexes. Car c'est là le point faible de DirectX : il effraie les non-initiés. Ses détracteurs premiers dénonçaient, à juste titre, sa complexité d'utilisation et d'intégration. Ils considéraient DirectX comme une pâle copie d'OpenGL, tant l'architecture et le design de ce dernier semblaient à la fois plus souple, voire plus agréable, que " l'immonde usine à gaz " de son principal concurrent. Mais il faut reconnaître que Microsoft a fait de gros efforts à partir de 1997. Si la version 1.0 de GameSDK fut une vraie catastrophe, la version 3.0, qui apparut quelques années plus tard, commençait à présenter quelques caractéristiques intéressantes. Aujourd'hui, avec la version 9.0, DirectX est l'outil le plus puissant dédié aux jeux vidéo. Mais il n'en demeure pas moins que la maîtrise de DirectX, ou même d'OpenGL réputé plus accessible, est souvent longue et fastidieuse et par conséquent onéreuse pour un studio de développement ne possédant pas ces compétences en interne. C'est pourquoi, un certain nombre de sociétés propose des environnements optimisant l'exploitation et réduisant les temps et délais de développement de jeux impliquant l'utilisation de DirectX ou d'OpenGL. Parmi les outils existants, la société Criterion tire son épingle du jeu avec son moteur RenderWare qui représente depuis quelques années, une référence incontournable pour tout type de développement, indépendamment de la plate-forme cible.

Microsoft-XNA : la révolution multi-plate-forme

Lors du dernier GDC (Games Developer Conference) Microsoft a dévoilé sa dernière

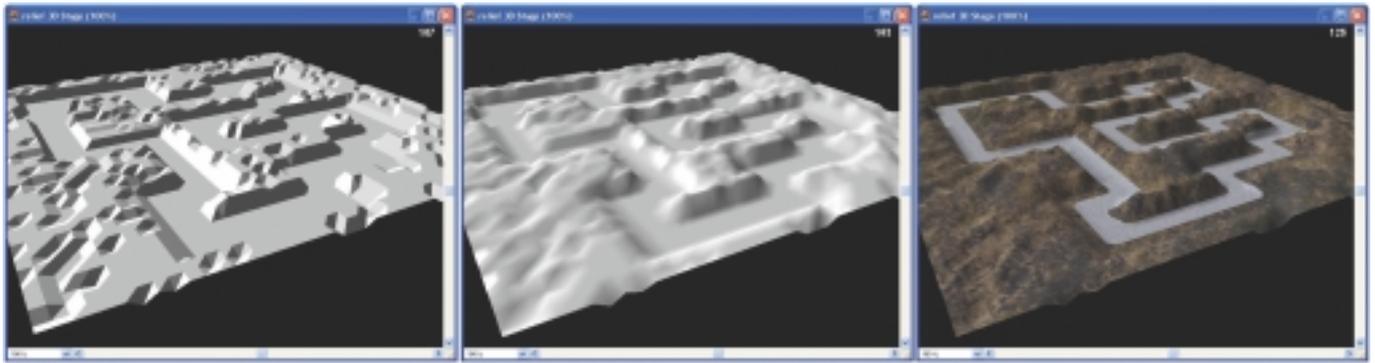


Figure 6 : Depuis la version 8.5, Director intègre un moteur 3D aux capacités étonnantes

technologie, qui porte le nom de XNA (Xbox Next-Generation Architecture). XNA est une nouvelle technologie permettant aux développeurs de créer des jeux s'exécutant sur Windows, Xbox et Windows Mobile, en utilisant un seul et même environnement de développement. Autrement dit, il serait possible de concevoir un jeu de nouvelle génération pour 3 plates-formes différentes avec un seul et même développement, ce qui réduirait fortement les coûts de production. Les développeurs seront ainsi libérés des préoccupations impliquant la portabilité du code et pourront se concentrer sur la créativité uniquement.



Pour plus d'information sur XNA, voir le site <http://www.microsoft.com/xna/>

Renderware studio, la technologie collaboratrice

RenderWare Studio est un environnement de développement de jeux, de type collaboratif. Cette technologie de Criterion est mondialement appréciée par les studios de développement. RenderWare, qui est même devenu la référence absolue pour le développement de jeux sur PlayStation II, GameCube, Xbox ou PC.



(Figure 4)

La particularité de RenderWare est de proposer un espace de travail commun, intégrant directement le travail des designers et des autres concepteurs de jeux non programmeurs (voir la figure 4). Renderware est un superbe outil, performant et fiable, qui permet des développe-



Figure 5 : Flash et Director, des amis pour la vie ?
Site Internet : www.macromedia.com

ments efficaces, en équipe. Cependant, Renderware peut apparaître dans certains cas surdimensionné, ou carrément pas adapté, au dossier de spécification de certains projets d'envergure moyenne. Une autre alternative s'impose en conséquence, et cette alternative pourrait bien être celle proposée par Macromedia, avec son duo fétiche Flash - Director.

Director - Flash, pour le meilleur ou pour le pire...

Director est un environnement de développement multimédia complet et polyvalent. De nombreux CD-Roms interactifs et jeux vidéo ludo-éducatifs utilisent Director. A la fin de la décennie 90, 70% des CD-Roms et jeux ludo-éducatifs utilisaient cet environnement. Et pourtant, Director est paradoxalement un logiciel peu connu parmi la communauté des développeurs francophones, ce qui constitue un paradoxe assez incroyable, étant donné les capacités de cet environnement et surtout le nombre de logiciels multimédia conçus avec Director. (Figure 5)

Au fil des années, Macromedia fit donc évoluer son logiciel fétiche. Director fut porté sur PC au début des années 90, et vit l'arrivée d'un premier langage de script qui porte le nom de langage Lingo ("jargon" en anglais). Ce langage était, dans ses premières versions, plutôt basique, mais il évolua de manière importante, jusqu'à devenir un impressionnant langage de programmation complet, intuitif, orienté objet, et même performant. Aujourd'hui,

Director est un environnement extraordinaire techniquement et polyvalent, bénéficiant depuis la version 10 d'un nouveau langage de programmation au standard ECMA, ce qui permet de programmer actuellement un logiciel avec la syntaxe JavaScript, par exemple, directement dans Director. D'autre part, Macromedia a bénéficié du support de la société Intel pour concevoir un moteur 3D étonnant, complètement orienté online, connu sous le nom de Shockwave 3D. Et les fonctionnalités de Director ne s'arrêtent pas là, le moteur de simulation physique Havok est gratuitement disponible avec Director, ce qui permet de concevoir des jeux 3D surprenants. Cerise sur le gâteau, Director est à présent complètement cross-platform, avec la possibilité de compiler un exécutable Windows ET Macintosh... bref, Director est probablement le logiciel le plus complet et le plus impressionnant d'un point de vue technique et ergonomique, car il bénéficie d'un développement de plus de vingt ans... qui dit mieux ? (Figure 6) Director souffre depuis quelques années de la concurrence de son petit frère Flash, du même éditeur Macromedia, et qui s'est spécialisé dans la production de contenu animé et interactif pour Internet. Malheureusement, ce succès à un peu cannibalisé commercialement Director qui possède pourtant sa propre technologie online et qui porte le nom de Shockwave. Director est bien plus complet et génère du contenu beaucoup plus performant que celui généré par son petit frère Flash, mais il ne bénéficie pas de la même renommée, preuve en est la diffusion du plug-in Flash, qui est installé sur 98% des ordinateurs, alors que Shockwave n'est diffusé qu'à hauteur de 70%. De plus, la politique marketing de Macromedia est parfois confuse et brouille les pistes... il est possible de créer un jeu ou un CD-Rom basique avec Flash, mais ceci cause du tort à

Director, dont c'est pourtant la spécialité. Heureusement pour ce dernier, il n'existe pas de moteur 3D temps réel dans Flash, et les performances de son langage de script (qui porte le nom évocateur d'ActionScript) est bien moins important et puissant que le langage de Director. Un programme codé en Flash est bien souvent 40% fois moins rapide que sous Director, selon certains tests.

Paradoxalement, en combinant les deux environnements de Macromedia, le développeur multimédia obtient un environnement de développement extraordinaire. Depuis quelques années, il est en effet possible d'intégrer des animations Flash directement dans votre programme Director.

En ce qui concerne le développement de jeux vidéo, les coûts de production et les délais de développement sont bien inférieurs pour le studio qui utilise un IDE tel que Director. En effet, il est possible de créer un jeu 3D intéressant avec ce dernier, dans des délais extraordinairement moins importants qu'avec un environnement classique tel que Visual Studio, associé à DirectX. Il est vrai que ce type de production sera bien moins impressionnant techniquement et ne pourrait constituer en aucun cas un blockbuster, mais pour certaines productions du domaine ludo-éducatif ou de la 3D online, Director est probablement l'outil le plus adapté pour une société à taille " humaine " ou pour un indépendant travaillant en freelance. En divisant les délais de développement d'un jeu par dix, Director permet de réduire les budgets fatalement par dix, mais il n'est plus le seul...

Virtools le virtuose non virtuel

Le seul véritable concurrent de Director est français, et porte le nom, anglophone pourtant, de Virtools. Moins connu que Director, Virtools est cependant un extraordinaire outil



Figure 7 : Virtools existe aussi pour Xbox.
Site Internet : <http://www.virttools.com>



Figure 8 : Scol repose maintenant sur un concept " Open Source ", et donc gratuit...
Site Internet : <http://www.scol-technologies.org/>

de conception de jeux, orienté 3D online. Moins polyvalent que Director, il n'en demeure pas moins impressionnant dans ses capacités à produire un jeu 3D à coût réduit et pourtant performant. L'éditeur de jeux vidéo Microïds l'utilise ainsi depuis quelques années avec succès. Virtools constitue donc une autre alternative, mais il a un énorme défaut pour un concepteur de jeu vidéo indépendant : son prix, qui tourne autour de 5000 €. Certes, ce prix se justifie pleinement en raison des capacités fabuleuses de l'environnement, mais il est souvent un frein pour un développeur travaillant en freelance, qui se tournera plus facilement vers Director. (Figure 7)

Scol, le miraculé

Virtools n'est pas la seule technologie française permettant de produire des jeux 3D, l'un de ses concurrents fut longtemps le Scol. Symboliquement, de nombreux concepteurs de jeux 3D online sont très attachés à cette technologie qui faillit disparaître avec la société Cryo, qui déposa son bilan en 2002. Scol signifie " Standart Cryo Online Language " et a été

créé par le génial Sylvain Huet au milieu des années 90. Le rôle premier de Scol était de fournir aux développeurs un environnement de programmation pour concevoir des environnements 3D dédiés à Internet, mais aussi des CD-Rom de jeux originaux hybrides, combinant un univers 3D intéressant, mais complètement orienté online et donc sur Internet. (Figure 8) Techniquement en avance sur ses concurrents, Scol et la société Cryo Networks ne résistèrent pourtant pas à l'éclatement de la bulle Internet en 2001. Après le dépôt de bilan de Cryo Networks, son directeur technique et concepteur de Scol réussit un pari incroyable : récupérer les droits d'exploitation commerciaux de Scol pour le diffuser sous un modèle " Open Source " avec le support d'une association. Scol est véritablement un miraculé, mais il est difficile de miser aujourd'hui sur cette technologie, tant son développement futur reste incertain, étant donné le fait que seuls des bénévoles travaillent aujourd'hui sur son moteur. Mais qui sait, Scol n'a peut-être pas révélé tous ses secrets... ?

Que choisir ?

Ce panorama, non exhaustif, des différentes technologies disponibles pose une question simple : quelle est la solution la plus adaptée à vos besoins ? Investir dans une technologie en achetant une licence développeur à un coût, certes parfois élevé, mais rapidement rentable. En revanche, investir du temps et des ressources humaines dans une technologie peu adaptée à vos besoins ou à ceux de vos clients peut représenter une perte bien plus conséquente à terme. Une petite structure, comme un studio multimédia, qui se lance pour la première fois dans la conception d'un jeu vidéo ne peut pas envisager de concevoir un produit grand public, capable de concurrencer les jeux existant sur le marché. Mais d'autres branches du jeu existent, comme la 3D online, les " Web Games " ou bien les jeux sur plate-forme mobile (téléphones portables ou PDA). Pour finir, nous vous proposons ici un tableau récapitulatif des différentes technologies abordées dans ce dossier, avec les outils que nous préconisons, en fonction du type de jeu développé. Ce tableau récapitulatif ne présente pas, bien entendu, toutes les technologies existantes, mais probablement les principales qui font l'unanimité aujourd'hui pour la création de jeux vidéo.

■ Laurent Jayr

tech@tsm-internet.com

Type de jeu	Cible (client, utilisateur)	Outils préconisés	Autres outils
Jeux 3D de type " blockbuster " pour PC	Windows, Xbox	Renderware, Virtools	DirectX, XNA, OpenGL...
Jeux 3D de type " blockbuster " pour consoles	Xbox, PS2, etc.	Renderware	OpenGL, DirectX, XNA, Virtools (XBox)
Jeux 3D de type " ludo-éducatif "	Windows, Macintosh	Director	DirectX, OpenGL...
Jeux 3D online	Windows, Macintosh	Virtools ou Director	Java 3D, Scol
Jeux 2D de type " ludo-éducatif "	Windows, Macintosh	Director	Flash
Jeux 2D online	Windows, Macintosh	Flash	Director
Jeux 2D pour mobile	Palm, PocketPC	Java	Une solution avec Director et/ou Flash existe également