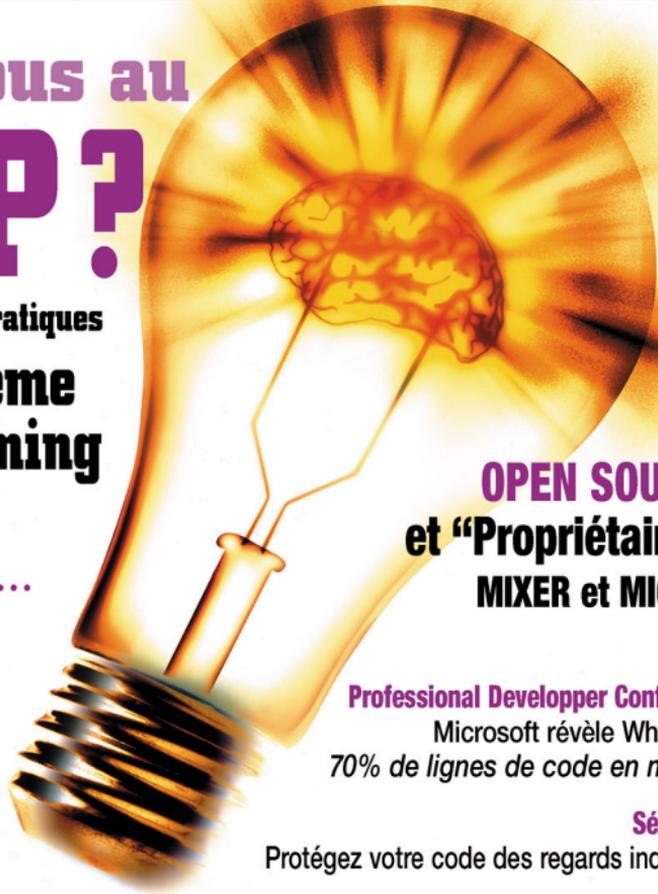


Êtes-vous au TOP?

Les 13 bonnes pratiques
de l'eXtreme
Programming

UP, MDA,
RUP, SOA, etc...
Les nouvelles
méthodes
expliquées

P.32



P.24

OPEN SOURCE
et "Propriétaire":
MIXER et MIGRER

Professional Developer Conference
Microsoft révèle Whidbey :
70% de lignes de code en moins !

Sécurité :
Protégez votre code des regards indiscrets

PHP

P.52

- Développer un Framework avec PHP
- PHP au delà du WEB

PRATIQUE

P.58

- Découvrir les attributs de C#
- Migrer de ASP.Net ou PHP vers Mono
- Java et les Sockets

LUDIQUE

P.78

Développer un Jeu 3D
Rally Race complet !

TECHNO

Le retour du client lourd

Printed in France - Imprimé en France
LUXEMBOURG 6,45 € - Canada 9,95 \$ CAN
www.programmez.com

M 04319 - 59 - F - 5,95 €





PDC 2003 Whidbey et Longhorn à l'honneur



"Longhorn est la plus importante évolution de Windows depuis Windows 95. Il offrira plus de puissance et de simplicité, notamment grâce au nouveau système de fichier WinFS. Ce système de fichier est mon Saint Graal depuis des années",

Bill Gates, Chief Software Architect, Microsoft

Sans annonce produit à court terme, Microsoft a profité de la dernière PDC - Professional Developer Conference - pour présenter sa nouvelle génération d'outils : Yukon, Whidbey et Longhorn. La dernière édition de la Professional Developer Conference (PDC) de Microsoft qui

se tenait du 26 au 30 octobre à Los Angeles a été l'occasion pour les 8.000 développeurs présents de découvrir en avant - première Longhorn - future version de Windows - ainsi qu'ASP.NET 2.0 et la prochaine version de Visual Studio .NET regroupés sous le nom de code Whidbey. Yukon (prochaine version de SQL Server), Object Space (outil de mapping objet/relationnel) et MSBuild ont également été

présentés aux participants du salon. "We were there at the beginning". Ce slogan imprimé sur les tee-shirts des organisateurs résume bien l'ambiance de la conférence. " Cette PDC est le point de départ d'une nouvelle génération d'applications [ndlr : basées sur .NET]. Nous sommes en train de remanier tous nos outils explique Craig Mundie, Directeur technique de Microsoft. L'ensemble des nouveaux outils reposera sur le serveur d'applica-

tion .NET et sera finement intégré à Longhorn. Whidbey sera disponible fin 2004, Yukon devrait être commercialisé courant 2005 et Longhorn en 2006. La stratégie de .NET semblant avoir porté ses fruits, les participants de la PDC sont tous repartis avec des versions alpha de Longhorn et de Whidbey dans leurs valises. Microsoft entend ainsi remonter un maximum de retours utilisateurs avant la finalisation de ses produits. ■ **David Thévenon**

• Le projet Mono mal en point

N'étant pas invité à s'exprimer lors de la PDC, Miguel de Icaza a improvisé une session dans les couloirs du centre de conférence pour faire un point très attendu sur le futur de Mono face à l'avalanche d'annonces telles que ASP.NET 2.0, Avalon, Longhorn, etc. Selon lui, rien n'empêche le projet Mono de continuer et " nous sommes déjà prêts à implémenter .NET V2 " a-t-il annoncé. . Pourtant les relations entre Microsoft et le projet Mono semblent s'être dégradées après le rachat de Ximian (fondée par Miguel de Icaza) par Novell.

• IBM DB2 supporte la CLR

Après Java, IBM vient d'intégrer la machine virtuelle de Microsoft (CLR) à sa base de données vedette. DB2 peut désormais exécuter des procédures stockées écrites en Visual Basic, C#, Visual C++ et l'ensemble des 23 autres langages supportés par .NET. Le SGBD/R d'IBM dispose également d'un connecteur géré (Managed Data Provider) et de plugins pour Visual Studio.NET 2003. Les développeurs peuvent donc manipuler les bases DB2 dans Visual Studio .NET comme ils le font déjà avec SQL Server ou Oracle. IBM DB2 est la première base de données certifiée .NET.

Pour en savoir plus : www.ibm.com/software/data/info/dotnet

• Amberpoint Express inclus dans Whidbey

L'outil d'administration de services web Amberpoint Express de l'éditeur éponyme sera inclus dans la prochaine version de Visual Studio .NET. Amberpoint Express facilite l'instrumentation du code nécessaire à la supervision de services web et s'intègre à Visual Studio .NET sous forme d'un plugin qui écrit en C# qui s'exécute sous .NET.

• MSBuild : la copie conforme de NANT made in Microsoft

Les développeurs .NET attendaient depuis de longs mois l'intégration de Nant (l'équivalent d'Ant dans la communauté .NET) à Visual Studio. En fait, ils devront désormais choisir entre MSBuild et Nant. L'outil de build (construction de projets) de Microsoft ressemble comme deux gouttes d'eau à son homologue open source, jusqu'aux API similaires.

Malgré quelques légères améliorations, on peut légitimement se demander pourquoi les développeurs de MSBuild - qui participent à Nant - ont préféré réinventer la roue plutôt que de capitaliser sur le seul projet qui comblait jusqu'alors le vide laissé par Microsoft.

Ils y étaient

"ASP.NET 2.0 m'a complètement soufflé. Microsoft va sûrement séduire beaucoup de développeurs car c'est un outil plus pratique, productif et intuitif."

→ **Grégory Renard** (Rédo) – Directeur Développement de WygWam
 "Whidbey intègre énormément d'améliorations par rapport à Visual Studio .NET 2003. On sent que Microsoft a écouté les développeurs pour mettre au point cette nouvelle version"

→ **Nicolas Sorel** (Nix), Directeur de Developers Association, fondateur de CodeS-SourceS.com,

"Faisant partie des alpha testeurs de Whidbey, c'est surtout LongHorn qui m'a bluffé. L'intégration de la CLR en tant que base du système, l'interface graphique calculée en temps réel directement par la carte graphique, la couche de communication Indigo et la gestion de " vues " par " axe " des dossiers de l'utilisateur par WinFS : impressionnant !"

→ **Kevin Auch** (kevin), Axyx Consultants,

"Ce fut certainement la rencontre informatique la plus dense et la plus prometteuse de la décennie pour les développeurs Microsoft. Jugez vous même: un OS innovant sur tous les aspects, des langages et un framework implémentant des paradigmes originaux et enfin, un SGBD managé révolutionnaire. I was there at the beginning"

→ **Patrick Smacchia**, consultant indépendant et auteur de " Pratique de .NET et C# "

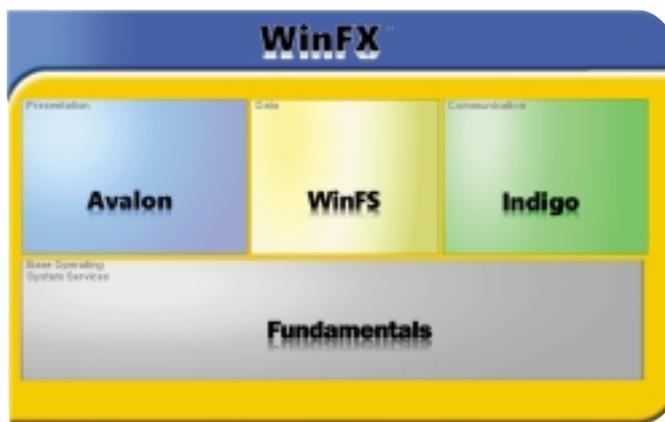
ASP.NET 2.0 devrait faire gagner beaucoup de temps sur les petits projets. Les développeurs vont retrouver le cycle de débogage très rapide d'ASP : modifier son code, tester immédiatement sans recompilation ni déploiement. Les applications RAD seront également plus faciles à maintenir grâce au câblage direct entre la couche de présentation et les classes métier (ObjectDataSource), notamment via ObjectSpaces"

→ **Pierre Couzy**, Winwise, Microsoft Regional Director sur la région parisienne

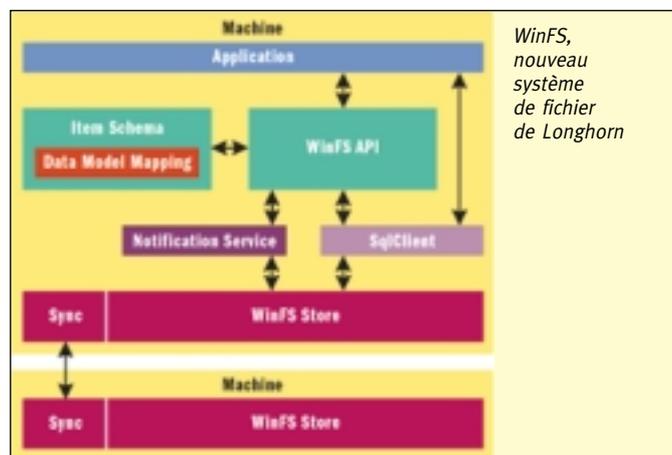
Longhorn : bien plus qu'une nouvelle version de Windows

Achevant la fusion de l'OS de Microsoft avec .NET, Longhorn propose un nouveau système de fichier – WinFS – ainsi qu'une nouvelle couche graphique – Avalon - qui vont révolutionner le développement d'application. Attendu pour 2006 et présenté pour la première fois au public par Bill Gates en personne lors de la dernière PDC, Longhorn marque

une rupture avec les versions précédentes de Windows. Pour les utilisateurs, le principal changement réside dans la capacité de Longhorn à fusionner l'ensemble des données des applications au sein d'un seul système de fichier (WinFS). Le cœur de Longhorn est organisé en trois couches : Avalon (couche graphique), WinFS (système de fichiers) et Indigo (couche



Les trois nouvelles couches de Longhorn exposées par l'API WinFX.



de communication). L'ensemble est exposé sous une API homogène, baptisée WinFX, qui succède à Win32.

WinFS

WinFS se présente comme une API de haut niveau qui permet de manipuler exactement de la même façon des documents (Word, Excel, e-mails, etc.), des fichiers XML ou des sources de données relationnelles. Le nouveau système de fichier de l'éditeur repose sur NTFS ainsi que sur un moteur SQL. Il embarque également des mécanismes de synchronisation de données et un système d'agents permettant de créer faci-

lement des règles de manipulation des fichiers (comme on le fait actuellement dans Outlook). Plus intéressant, WinFS permet de déclarer ses propres types à l'aide de schémas XML afin de personnaliser le système d'exploitation.

Avalon

La couche graphique de Longhorn innove sur trois points. Contrairement aux technologies Windows précédentes qui reposaient sur du bitmap, Avalon propose une couche graphique vectorielle. Et plutôt que de laisser le processeur de l'unité centrale (CPU) calculer l'interface, Avalon s'appuie entièrement sur la carte graphique

(GPU). La description de l'interface peut s'effectuer avec XML Application Markup Language (XAML), un langage XML à mi chemin entre XUL et XForms.

Indigo

Nouveau socle de communication, Indigo propose une couche d'abstraction qui permet de sélection-

ner .NET Remoting ou les web services comme couche de transport des messages. Très proche de WSIF de la fondation Apache, cette couche d'abstraction nécessite moins d'une ligne pour passer d'un protocole à l'autre. " Indigo constitue le serveur d'application du futur pour Microsoft, il est destiné à héberger des

WebServices à travers les standards WS " commente Sami Jaber.

Vers un poste client entièrement programmable

Longhorn se distingue par une intégration complète entre l'interface utilisateur, le système de fichier et le socle d'exécution des applications tierces qui se fondent dans

l'OS. En ce sens, Longhorn préfigure une nouvelle génération de postes utilisateurs entièrement programmables et donc extensibles à l'aide d'API de haut niveau.

■ David Thévenon

Pour en savoir plus :

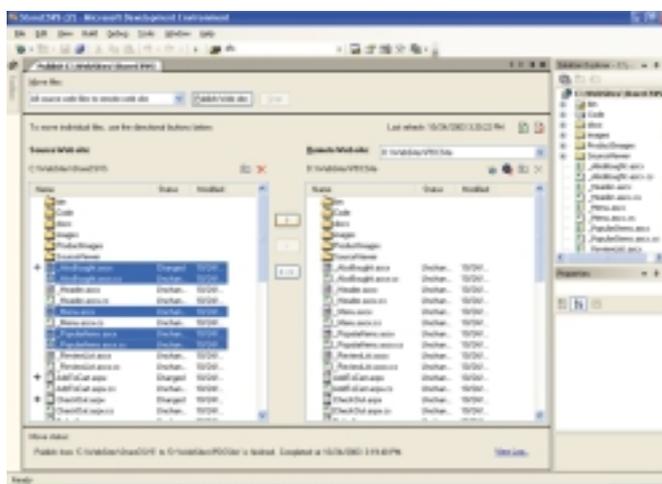
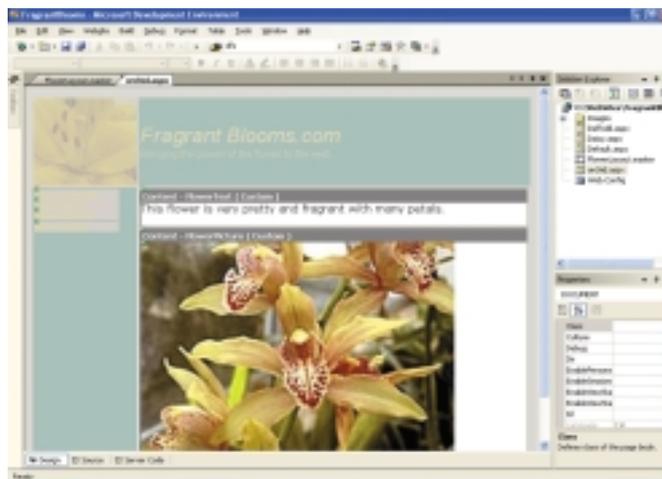
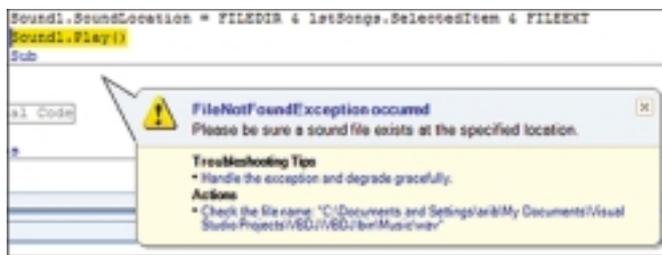
Longhorn : msdn.microsoft.com/longhorn

Whidbey : 70% de lignes de code en moins

La prochaine version de Visual Studio .NET et des frameworks associés a été repensée pour limiter le nombre de lignes de code et pour accompagner toujours plus les développeurs. Selon Microsoft, Whidbey réduira considérablement les efforts des développeurs. L'éditeur a travaillé principalement sur deux points : la réduction du nombre de lignes de code, grâce à une API de plus haut niveau et l'amélioration des fonctions d'assistance au développement.

Building blocks

Une application qui nécessitait 50.000 lignes de code avec WSE n'en nécessitera plus que 27.000 avec Whidbey et 3 avec Indigo ! C'est du moins ce qu'a promis Microsoft en présentant les " building blocks " lors de son keynote technique. Ces API de haut niveau et les contrôles associés reprennent le principe des "application blocks" du .NET framework, en adaptant le concept aux différents langages supportés par VS.NET. Parmi les plus utiles on peut citer par exemple pour ASP.NET une gestion des membres, un moteur de personnalisation, une gestion de la navigation, un cache d'accès aux données, etc. L'API des building blocks joue également le rôle d'une couche d'abstraction, en permettant au développeur de



l'interfacer avec ses propres composants. L'ensemble de l'API est relié à Visual Studio .NET à l'aide d'assistants et de plus de 40 nouveaux contrôles serveurs paramétrables. L'accès et la manipulation des données est plus simple qu'avant, grâce à l'ajout de deux nouveaux contrôles. Il est enfin possible de glisser une source de données (contrôle Data) sur un webform, puis de la lier avec un nouveau contrôle " Grid " particulièrement complet. Quelques paramètres suffisent là où il fallait généralement écrire une cinquantaine de lignes de code pour personnaliser le comportement du DataGrid dans les versions précédentes de VS.NET. VB.NET dispose de son côté d'un ensemble de blocs de code pré-développés.

Guider les développeurs

Intellisens est désormais disponible pour tous les langages (HTML compris) et même les types génériques sont pris en compte. Dans la logique du correcteur orthographique de Word, les erreurs de syntaxes sont soulignées par l'IDE qui propose différentes corrections dans un smart tag. Dans la même logique, tous les contrôles proposent une liste de tâches à effectuer pour les paramétrer et le debugger VB présente une liste de solutions pour corriger les erreurs rencontrées.

La présentation du code ASP progresse également, avec la possibilité de réduire le code HTML, comme on le fait pour le code-behind. Enfin, ASP.NET se conforme aux standards tels que WCAG et XHTML, que VS.NET peut valider.

Déploiement et administration simplifiés

ASP.NET intègre un nouveau module de déploiement très proche de celui proposé par Dreamweaver. Ce module permet également de créer des répertoires virtuels IIS directement depuis VS.NET. Mais c'est surtout

ClickOnce qui a retenu l'attention des participants du salon. Cette nouvelle technologie - très proche de JavaWebStart de Sun - permet de déployer un projet en code managé à l'aide d'une simple URL. L'application peut alors être mise à jour bloc par bloc. Le déploiement d'un projet ASP.NET s'effectue désormais par simple copie de fichiers.

Pour en savoir plus :

VS.NET : msdn.microsoft.com/vstudio/productinfo/roadmap.aspx

ASP.NET : www.asp.net/whidbey

■ David Thévenon

VB.NET : déploiement en un clic

VB.NET Whidbey bénéficie également de nombreuses améliorations, comme la possibilité de créer automatiquement un data-grid par simple glisser-déplacer d'un service web. Le nouveau contrôle DataGrid permet également de passer automatiquement d'un mode liste à un mode fiche. Autre avancée : ClickOnce facilité

Visual Basic .NET 2003	Const GreetingTime As String = "Greeting" Dim iDisplay As Integer Dim ResMgr As ResourceManager ResMgr = New ResourceManager("Resources.Sample.MyStrings", File.GetType.Assembly) iDisplay = ResMgr.GetString(GreetingTime)
Visual Basic Whidbey	My.Resources.MyStrings.Greeting

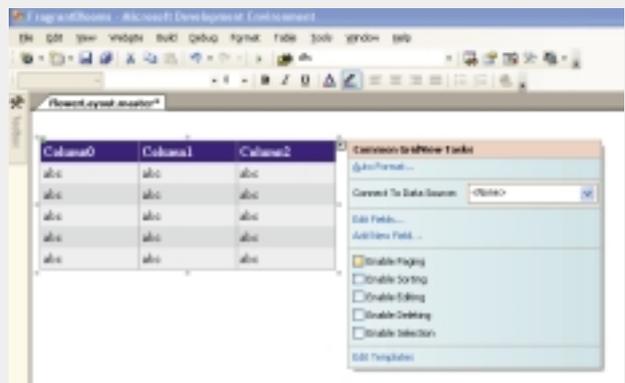
```
Dim nDir As Integer
Dim i As Integer
Dim b As Boolean

If Microsoft.VisualBasic.IIf(nDir = 0,
    ReDim DirNames(nDir)
    DirName = Dir(Path.Combine(
    Do While Len(DirName) > 0
```

le déploiement des applications VB au travers d'une simple URL et permet de mettre automatiquement à jour certaines parties de l'application. Un nouveau debugger, plus ergonomique, fait également son apparition.

ASP.NET : Master Page, thèmes et WebParts

ASP.NET 2.0 intègre un mécanisme de "page principale" qui accélère la création d'un site. Le principe est simple. Une fois la page principale créée (structure du site avec menu, barre de navigation, etc.) il suffit de préciser la zone personnalisable qui affichera les "pages" du site à l'aide du contrôle ContentPlaceHolder. Grâce à cette approche, il suffit d'associer un thème à une master page pour que le site change de look en un clic. Le binding s'effectue dans un fichier de configuration au format XML, paramétrable



par un assistant. L'utilisateur peut ensuite personnaliser son interface (choix d'un thème par exemple) grâce aux building blocks login et personnalisation qui gèrent les accès et stockent l'ensemble de ses paramètres en base de données. Il est maintenant possible d'intégrer des webparts dans l'équivalent d'un ContentPlaceHolder. Aucun développement n'est nécessaire, le contrôle WebPart gérant l'ensemble des interactions avec les autres blocs pour le développeur. ASP.NET inclut également un nouvel outil de modélisation graphique de données disposant d'un mécanisme de synchronisation. Enfin, ASP.NET 2.0 supporte mieux les périphériques mobiles au travers d'une gestion étendue des différents profils.

WhiteHorse

Très attendu, l'outil de modélisation WhiteHorse permet de modéliser la couche logique d'une architecture orientée services en connectant des services web entre eux, tout en vérifiant son adéquation avec l'infrastructure physique existante, en fonction de paramètres tels que le niveau de sécurité attendu, les protocoles utilisés, etc. WhiteHorse sera directement intégré à Visual Studio Whidbey. Ce nouvel outil fait partie de l'initiative Dynamic

C# 2.0 : generics & itérateurs

Parmi les nouveautés très attendues, Microsoft a confirmé l'ajout de plusieurs fonctionnalités clés au langage vedette de .NET. Les génériques, itérateurs, méthodes anonymes et types partiels font leur apparition.

PHPIndex : point de rencontre de la communauté PHP

Fondée en 1999 par Armel Fauveau, cette communauté s'est peu à peu imposée comme un point de rencontre (au travers des forums notamment) de la communauté PHP francophone.

ENTRETIEN AVEC SON FONDATEUR



Programmez ! : Peux-tu nous présenter brièvement PHPIndex.com ?

Armel : PHPIndex regroupe diverses ressources liées à la plate-forme PHP : un fil d'actualités quotidiennes, des

articles originaux mais aussi des relectures d'articles publiés en France ou à l'étranger (avec un résumé pour inviter le lecteur à consulter l'article original). PHPIndex propose aussi des trucs et astuces, des offres d'emploi, des forums, ainsi que de nombreux liens vers des ressources diverses (doc. en ligne, sites majeurs, livres, etc.). Enfin, une FAQ de près de 300 questions/réponses est également disponible. Actuellement PHPIndex affiche 750 000 pages vues chaque mois.

P ! : Qu'est-ce qui t'a poussé à créer cette communauté ?

Armel : Le vide. L'idée m'est venue au printemps 1999 alors que j'animais les forums du support technique d'un grand hébergeur français qui supportait PHP et MySQL. Je répondais aux nombreux messages de clients qui découvriraient PHP. La majorité n'avait pas le réflexe

de consulter la documentation officielle. Elle était pourtant déjà fort complète. Je rappelais donc régulièrement l'existence du site officiel (php.net) ainsi que d'autres liens utiles comme PHPBuilder (phpbuilder.com) et DevShed (devshed.com). En avril 1999, j'ai répertorié les ressources francophones. Le constat fut sans appel : à part une poignée de pages perso ou de listes de diffusion au trafic aléatoire, il n'y avait rien. J'ai alors humblement tenté de regrouper ce que je savais de PHP sur un site, afin d'en faire profiter le plus grand nombre. L'idée de PHPIndex venait de germer.

P ! : Que penses-tu apporter aux lecteurs de PHPIndex.com ?

Armel : De l'information liée à la plate-forme PHP :). PHPIndex apporte aussi énormément aux personnes en charge de le maintenir. PHPIndex constitue une formidable plate-forme de veille et de formation continue. Elle nous force à nous tenir au courant, tant de l'actualité que des évolutions techniques.

P ! : Quel est le point fort de PHPIndex.com ?

Armel : Un seul ? D'abord, PHPIndex est un site qui vit. C'est idiot à dire, mais si PHPIndex est le plus ancien, c'est aussi l'un des seuls sites francophones qui est mis à jour quotidiennement depuis 1999, sans interruption. Ce point me semble essentiel. Ensuite, PHPIndex offre plusieurs niveaux de lecture. Nous mettons régulièrement en avant des actualités qui intéressent d'autres publics que les développeurs. Je pense à des retours d'expériences par exemple, ou encore à des articles de fond. Cela nous a conduit à la rédaction d'un livre blanc sur la "Place de la plate-forme PHP dans l'économie



française" (www.phpindex.com/download/phpen-france.php3). On est loin d'un sujet technique.

P ! : Combien êtes-vous à faire vivre PHPIndex.com ?

Armel : C'est un réel travail d'équipe. Frédéric Hovart est à l'origine des "Sorties du Jour" qui synthétisent l'actualité quotidienne des projets gravitant autour de PHP. Il référence aussi interviews, retours d'expérience et autres actualités liées à PHP. Plusieurs employés de GLOBALIS, la société éditrice de PHPIndex, participent aussi. Je pense en particulier à Catherine Lao ou Arnaud Buchoux. Enfin, certains lecteurs réguliers qu'il serait trop long de citer ici (ils m'excuseront je l'espère) nous apportent également leurs contributions.

P ! : Quels sont tes projets pour PHPIndex.com ?

Armel : Ma priorité c'est de faire vivre le site existant. Ceci étant, je n'écarte pas l'idée de faire évoluer le site en profondeur un jour... dès que j'en aurais le temps ! La motivation, quant à elle, est bien là.

■ Propos recueillis par David Thévenon

"PHPIndex est pour moi la référence PHP francophone (avec Nexen et feu PHPInfo.net). Du point de vue utilisateur de PHP, c'est une mine d'or concernant les ressources PHP (actualités, articles, annuaires, ...) mais ce qui m'intéresse le plus, c'est le bloc 'Actualités' qui est mis à jour très souvent. Du point de vue développeur, PHPIndex me permet de faire passer des news sur mes propres développements ou sur des articles que j'aurais écrit. J'apporte comme cela ma "brique" à l'édifice PHP".

■ David Duret (Pilgrim), auteur du moteur MALA template et rédacteur/modérateur pour Misamatik.net.

Fiche technique

Adresse : www.phpindex.com

Date de création : 1999

Thème : PHP et son écosystème

Technos : PHP, MySQL

Cible : Webmasters, développeurs, architectes

Niveau : débutant, confirmé, expert

Ressources : actualité, articles, liste de diffusion, forums, source, emploi, etc.

Nombre de membres : pas de système de membres

100 000 visiteurs uniques par mois

Nombre de contributeurs actifs : plus de 20

Nombre d'inscrits à la Newsletter : 3 200

Open source et "Propriétaire" Mixer et Migrer

Linux, Open Source, Logiciels Libres, sont trois réalités que l'on rencontre de plus en plus dans les entreprises, chez les développeurs et certains utilisateurs avancés. Dans *Programmez !*, nous parlons régulièrement des outils et langages libres. Ce focus aborde deux problématiques sur la manière de faire cohabiter des outils propriétaires et Open Source et de migrer purement et simplement vers l'Open Source.

Rien n'est jamais blanc ou noir dans l'univers informatique. Dans le cas du système d'information et du développement, les nuances sont grandes et multiples. Que l'on migre ou que l'on décide de mixer, il y a des précautions à prendre. Si théoriquement et dans les discours, cela semble facile, la réalité impose restrictions et compromis. Oui, il est possible de migrer totalement, voire de mettre en place un "mixte". Non, cela n'est pas possible dans tous les cas.

Différencier Open Source et Linux

Trop souvent encore, quand on pense Linux, on pense Open Source, Logiciels Libres. C'est un réflexe exagéré. Car, on peut utiliser du Libre et de l'Open Source sans que l'on ait besoin d'installer un système Linux. Il est pos-



sible de rester avec son Solaris, son MacOS X ou son Windows. Nous verrons que cette précision, jamais inutile à rappeler, est importante à garder en mémoire. Si vous optez pour de l'Open Source, vous n'êtes pas obligé de migrer votre environnement système !

Vive l'Open Source dans le développement !

L'utilisation d'Open Source dans le développement pour un développeur, une SSII, une entreprise ne sera peut-être pas la même. Utiliser des outils Open Source signifie tout d'abord un coût de licence nul. Quand un IDE coûte des centaines, voire des milliers d'euros, l'économie n'est pas négligeable. Nous reviendrons plus loin sur les nuances à apporter. Dans le domaine Java, Eclipse est une référence incontournable, même s'il lui manque toujours un RAD. N'allez pas croire que le monde Windows et notamment .NET, soit le parent pauvre de la communauté Open Source. Il existe un test unitaire, Nunit (un clone de Junit), un IDE gratuit (SharpDevelop), des bibliothèques Open Source (ex. : iTextSharp ou ReportNet, permettant de générer des PDF), un générateur de documentation (Ndoc), un OpenGL adapté à .NET (CsGL), un ANT .NET (Nant), une version de l'API log4j, etc. Ne parlons même pas de l'offre C, C++, Python, PHP... Le langage C# semble inspirer les projets Open Source. On trouve ainsi des bibliothèques et outils très diversifiés : du SQL, SVG, convertisseur de code Delphi vers C#, un IDE C# (DrC#), ou encore un compilateur, des moteurs 3D, etc. Si vous avez besoin de générer du PDF, de créer dynamiquement des cartes géographiques, ou de faire du mapping, l'offre est vaste. Reconnaissons



tout de même que Java / J2EE est plus largement utilisé dans les projets Open Source. Vous aurez moins de difficulté à trouver un outil Java que C#. Plusieurs serveurs J2EE sont Open Source / Libres (JBoss, JonAS). De plus en plus, les éditeurs (hormis Microsoft) intègrent des solutions Open Source dans leurs offres. On le voit parfaitement pour l'offre entreprise. Certains éditeurs redéfinissent une partie de leur stratégie grâce à l'Open Source. C'est notamment le cas de Novell. D'autres, complètent les gammes (Sun, HP, IBM).

Mixer propriétaire / open source

L'un des grands intérêts de l'Open Source est de pouvoir mixer les deux univers. Il existe

divers plug-ins Open Source pour Visual Studio .NET. Divers outils Open Source sont déjà abondamment utilisés par les IDE Java : Ant, Junit, Struts... Pour des fonctions d'un projet, vous pouvez implémenter des bibliothèques libres, à condition qu'elles existent pour votre IDE propriétaire... Cette mixité est une des tendances actuelles. Le CVS est largement mis en œuvre pour les petits projets, pour les gros, un environnement plus lourd sera peut-être nécessaire. Bref, techniquement, rien n'interdit un tel mixage. Dans les faits, il faudra tout de même envisager quelques tests d'intégration, afin de vérifier le bon fonctionnement.

Le mixage des solutions permet aussi de ne pas être lié uniquement à un éditeur propriétaire. Vous pouvez aussi mettre en place des couches Open Source pour la compilation, les tests, la documentation... limitant ainsi le lien de dépendance. Il peut aussi constituer une étape douce pour envisager à terme une migration totale. Ce n'est pas un hasard si on rencontre régulièrement un frontal Open Source avec une base propriétaire du genre Oracle. Le "mixte" a l'avantage de pouvoir garder des fonctionnalités "propriétaires" que l'on n'a pas (ou mal) dans l'Open Source. Dans le cas d'applications VB, VB.NET, il sera nécessaire de rester sous Windows. D'ailleurs, il n'existe pas en Open Source un IDE capable de rivaliser avec la richesse fonctionnelle d'un JBuilder ou d'un Visual Studio .NET.

De plus, le mixage a l'avantage de ne pas casser un environnement applicatif fonctionnant parfaitement. Pourquoi refaire les applications si celles-ci fonctionnent depuis des années ? Dans ce cas, on peut introduire des modules Open Source, par exemple dans les couches basses (voir le cas ADP).

Attention aux absents

Aussi diversifiée que soit l'offre, il demeure quelques manques. Sur les méthodes de développements et les nouvelles architectures (UP, MDA, UML, SOA...), l'Open Source est encore loin d'offrir les mêmes fonctionnalités que les outils Rational ou Together. Concernant UML, il n'existe pas encore de modèleur digne de ce nom. Cependant, pour les petits et moyens projets, cela devrait suffire. Côté SGBD, si les solutions Open Source n'ont plus rien à prouver, elles ont encore du mal à tenir la compa-

raison avec des outils de type Sybase, IBM, Oracle, Microsoft.

Migrer du propriétaire à l'Open Source

Plus encore que dans le cas d'un "mixte", la migration nécessite de la méthode et du pragmatisme. Il faut déterminer ce que l'on souhaite migrer, choisir les outils Open Source que l'on utilisera. Dans le cas d'outils de développement, les solutions Open Source choisies ont-elles les fonctions nécessaires ? Comment reprendre l'existant ? Ces questions doivent être étudiées avant toute migration.

Il est évident que pour certains types de projets, il paraît difficile de ne pas réécrire l'application. Les projets VB ne sont pas migrables tels quels. Dans le cas d'ASP, il existe des convertisseurs. ASP2PHP est le plus connu. Mais attention, l'ASP première génération n'obligeait pas à une stricte séparation entre le contenant et le traitement serveurs / données. S'il est possible de faire tourner de l'ASP.NET sous Apache, cela nécessite une bonne maîtrise.

Plus le code sera de mauvaise qualité, plus il y aura de travail à faire. D'une manière générale, toutes les fonctions "très" propriétaires (API, bibliothèques, fonctions, interface, appels systèmes, etc.) causeront obligatoirement des



re aujourd'hui de migrer un EJB d'un IDE à un autre, d'un serveur à un autre.

Dans le cadre d'une migration, il faudra aussi regarder si les projets Open Source choisis sont pérennes ou non. Il s'agit d'un point important, surtout en entreprise. L'autre problème éventuel concerne les compétences. Les développeurs sont-ils compétents sur les nouveaux outils ? Si ce n'est pas le cas, la formation paraît inévitable. De la bonne maîtrise des outils dépend la réussite ou l'échec de la migration.

SSII à la rescousse ?

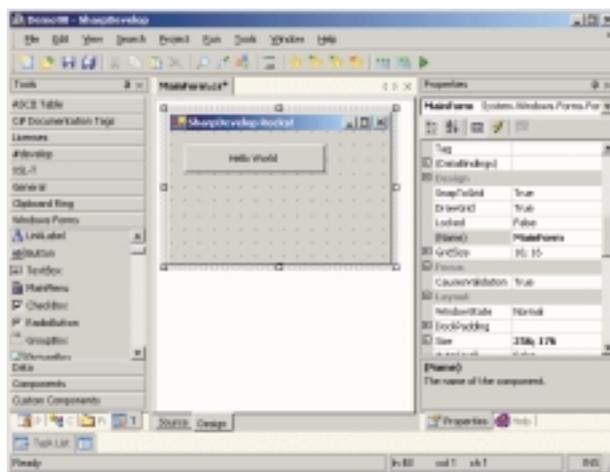
Si aujourd'hui, les migrations et l'implémentation d'outils Open Source sont monnaie courante, il ne faut pas pour autant croire que cela se déroule sans problèmes. Si l'entreprise n'a pas de compétences en interne, un prestataire extérieur est vivement conseillé au moins pour établir un audit des besoins, et apporter les conseils nécessaires dans les choix des outils et dans la méthode. Si vous êtes un développeur indépendant, une auto-formation est un minimum. Vous pouvez aussi envisager une formation auprès d'un spécialiste.

Bref...

... que l'on opte pour un "mixte" ou une migration, rien ne doit être fait au hasard. Il ne faudra pas oublier de prendre en compte l'éventuel problème de la licence de l'outil Open Source. Si l'application est à usage interne, pas de souci à avoir. Aujourd'hui, et encore plus demain,

l'entreprise, le développeur, souhaite être de plus en plus indépendants des éditeurs. Même si on ne souhaite pas fondamentalement changer d'IDE, on peut ici et là, rendre plus indépendantes certaines fonctions. Bref, vive l'Open Source dans le développement !

■ François Tonic



soucis dans une migration. D'où la nécessité de bien déterminer les fonctions nécessaires dans les outils Open Source / Libres. Si dans le cas d'un C / C++ ANSI, le problème se posera peu, idem pour le code Java / JavaBeans, pour les composants métiers de type EJB, l'affaire est toute autre. Il est (très) difficile enco-

"L'Open Source évite de se faire piéger par le propriétaire"

IdealX réalise constamment des projets de migrations d'environnement propriétaires vers l'Open Source. Programmez a interrogé David Barth, Directeur du département Infrastructure, pour en savoir un peu plus sur l'intérêt d'utiliser l'Open Source dans la chaîne de développement.

Programmez ! : On a parfois un peu de mal à voir de quelle manière intégrer des outils de développement Open Source dans un environnement existant. Quelle est votre expérience ? Que peut apporter l'Open Source dans son développement ?

David Barth : Il ne faut être lié à un environnement. C'est l'avantage de l'Open Source. Par exemple avec du C / C++, on aura du mal à changer d'environnement, d'un IDE Borland à un outil Microsoft. L'Open Source évite de se faire " piéger " par le propriétaire. On constate une concentration des éditeurs et le risque d'en être dépendant. Avec l'Open Source, c'est l'inverse. On peut le faire évoluer et le maintenir. Il est possible, grâce à l'Open Source de mettre en place une chaîne de compilation "indépendante" de l'environnement de développement. Sur le makefile, c'est identique. On peut utiliser autoconfig pour créer ses makefiles et éviter ainsi d'être lié à un environnement.

PI : Comment cela se passe-t-il si on possède une application VB, SGBD ou Unix ?

DB : On réalise beaucoup de migrations d'Unix à Linux. Les deux systèmes sont très proches. Tout le patrimoine Unix est dans les API de Linux. Les applications Windows sont un peu à part. Dans le cas des bases de données, on a rencontré plusieurs projets liés à une migration d'Oracle vers Postgresql. C'est relativement transparent. S'il y a du code C embarqué dans l'application Oracle, on a la même chose de l'autre côté. Les problèmes de ce type de migration sont bien connus. La passerelle est finalement assez simple. Sur les langages scripts, il n'y a pas beaucoup de soucis. Les plus populaires étant en Open Source. Concernant, VB ou ASP, il existe un gros héri-



tage. Je pense que les entreprises n'investiront plus dans ces technologies et qu'elles vont laisser dépérir les applications existantes. Les nouveaux développements se faisant en Open Source.

PI : Java apporte une certaine abstraction par rapport aux outils et aux systèmes. Cela se vérifie-t-il ?

DB : Dans Java, il n'y a pas le problème de la portabilité. Là aussi, il faut essayer de ne pas être lié à un environnement, à un IDE. Pour moi, Eclipse est un événement majeur. En développement, on peut choisir de nombreux outils / utilitaires Open Source : Ant, Solex, Cactus, Junit, etc. L'opposition Propriétaire / Open Source est estompée dans un contexte Java. Des API aussi importantes que Struts sont en Open Source.

PI : Malgré tout, migrer des EJB d'un IDE à un autre ou d'un serveur à un autre, pose toujours beaucoup de problèmes. L'Open Source ne semble pas régler cela. Qu'en pensez-vous ?

DB : C'est en effet un problème. Mais, Xdoctet est sans aucun doute une avancée majeure pour résoudre cela.

PI : Aujourd'hui, la notion de cycle de vie de l'application est primordiale. On le voit avec les méthodes de développements : RUP, UP, XP. Dans le monde Open Source, il n'existe pas d'outils " lourds " de type Rational ou Together, ni de gestion de cycle de vie aussi complète. Il existe aussi un certain manque autour d'UML. Cette absence pénalise-t-elle l'Open Source ?

DB : En Open Source, il nous manque effectivement un vrai modelleur UML à l'image des outils Rational. On ne peut pas modéliser de

lourds projets avec des outils Open Source. Mais, pour moi, le code demeure le plus important.

Sur ce type d'outils (ndlr : les modelleurs), il y a moins d'adhérence. Pour la gestion documentation, le CVS, les tests, les traces, on a ce qu'il faut en Open Source. On sait aussi gérer les exigences. Au lieu de dépenser pour acheter des outils du genre de Rational, on peut réduire la facture en utilisant de l'Open Source. Par exemple, à IdealX, on utilise de l'UP avec de la programmation XP, sans pour autant utiliser des outils propriétaires.

PI : Admettons que je souhaite implémenter une librairie 3D open source dans un IDE propriétaire, cela pose-t-il un souci ?

DB : Techniquement, non. Je n'y vois pas de problèmes particuliers. Dans ce cas, il faut faire attention aux éventuels problèmes juridiques liés à la licence des librairies.

PI : Vous venez d'évoquer rapidement le problème des licences. Pouvez-vous préciser la situation et les problèmes éventuels ?

DB : C'est une des premières questions qu'une entreprise se pose. Si j'utilise une librairie GPL, le programme réalisé doit être lui aussi GPL. Si l'application est à un usage uniquement interne, là, il n'y a aucun problème avec la licence. On ne force pas à reverser le code de l'application à la communauté.

PI : Le fait d'utiliser des outils Open Source favorise-t-il le retour sur investissement dans un projet ? Le développeur ou l'entreprise économise-t-il réellement ?

DB : Je n'ai pas de chiffres. Sur des projets utilisant Samba et/ou LDAP, le retour sur investissement est assez rapide. Sur le développement, c'est avant tout une question de stratégie et de choix. Ici, l'expérience guide l'entreprise. Quoi qu'il en soit, l'Open Source est un gisement où l'on peut puiser pour sa chaîne de développement. L'entreprise et le développeur l'ont compris.

■ *Propos recueillis par François Tonic*



ADP aime Linux et l'Open Source

Aéroport de Paris (ADP) cherchait à améliorer son infrastructure logicielle de paiement de ses parkings. Il y avait dès le départ volonté d'ouvrir l'architecture à Linux et au monde Open Source, encore fallait-il préserver l'existant. Un défi pour Thalès e-Transactions et OpenWide...

ADP possède de nombreux parkings à Roissy et Orly. L'idée de départ était d'améliorer la gestion et la communication du système de paiements. Pour ce faire, ADP a fait appel à Thalès e-Transactions, spécialisé dans les solutions monétiques. Dès le départ, Linux et Open Source faisaient parties des solutions envisagées. Pour réaliser une telle intégration, Thalès e-Transactions chercha un spécialiste dans l'intégration de l'Open Source. OpenWide arriva alors dans le projet.

Du DOS 6.22...

ADP disposait d'un environnement de paiement automatique (Largo) fonctionnant sous DOS 6.22. Deux limitations sont apparues au fil des années : système propriétaire, environnement peu évolutif. Mais ce système avait fait la preuve de sa robustesse et de sa fiabilité... Deux critères vitaux pour ADP qui voulait garder cette même stabilité dans la nouvelle

Windows. ADP souhaitait préserver les habitudes d'utilisateur du poste Windows. La couche de communication était du " banal " TCP/IP.

... à Linux

Dire que l'on veut du Linux, c'est bien, mais choisir la bonne distribution est une autre affaire. Le choix se porta sur Linux 2.1 de MontaVista. OpenWide avait acquis une solide expertise sur la distribution de MontaVista. Une fois le package Linux choisi, il fallut ré-architecturer l'application tout ainsi que les packages.

Sus à l'assembleur !

L'assembleur utilisé initialement était là pour contourner la limitation de 640 Ko du DOS. Comme sous Linux, cette limitation n'existe pas, lors du projet, "l'assembleur a donc pu être supprimé sans vergogne pour des raisons évidentes de simplicité de maintenance et d'amélioration de portabilité" explique Gilles Chanteperrix d'OpenWide.

Le code initial des applications et composants avait été fait par la société CGA, appartenant désormais au groupe Thalès. Pour OpenWide, il n'y a pas eu de contrainte particulière concernant l'IDE précise Gilles Chanteperrix. Pour le développeur, les outils classiques de l'Open Source ont été utilisés : CVS, doxygen (documentation) et auto-tools (système de compilation). L'objectif n'était pas de tout " chambouler ". Les applications ne furent pas réécrites. Une couche intermédiaire fut réalisée pour faire communiquer les anciens composants et les nouveaux. Finalement, les difficultés techniques ne furent pas spécialement importantes. " Puisque nous disposions des sources du logiciel que nous portions, la difficulté ne provenait pas tant du



passage d'un système propriétaire à un système "Open Source" que du changement de plate-forme. Les difficultés techniques proviennent, lorsque l'on passe de MS-DOS à Linux x86 : du passage d'un compilateur 16 bits à un compilateur 32, des jeux de caractères différents et enfin de la nécessité de pouvoir récupérer les fichiers de données d'origine " précise Gilles Chanteperrix.

Un projet concluant

Finalement, si la pression du délai était grande, OpenWide a eu une quinzaine de jours d'avance. ADP a mis en test la solution sur le parking Po d'Orly avant un déploiement plus global. Sur une durée de 6 mois au total du projet, OpenWide n'est réellement intervenu que sur 3 mois. Cela a mobilisé deux personnes à plein temps. Dès le départ, ADP voulait de l'Open Source dans ce projet. Il fallait trouver une société spécialisée dans ce genre d'outils. Patrick Bénichou, PDG d'Open Wide, ajoute : " Nous sommes très heureux d'être intervenus auprès de Thalès e-Transaction sur ce projet significatif. Cette réalisation commune traduit notre capacité de répondre à des problématiques industrielles, en faisant bénéficier nos clients des atouts de l'Open Source en matière de robustesse et ce, avec des gains financiers appréciables". Au final, ce projet démontre aussi la capacité à mixer une architecture système basée sur linux avec des applications qui pour leur part n'ont pas changé de modèle. ■ F.T.



Aéroport Roissy-CDG : vue aérienne de l'aérogare 2, au premier plan à gauche le terminal 2E Paris-Charles-de-Gaulle Airport : aerial view of terminal 2, in the foreground terminal 2E to the left

architecture logicielle. Les applications avaient été écrites en plusieurs langages. L'application principale était en C++ et intégrait 3 composants : un noyau multi tâche " maison " en C et assembleur, un composant de gestion de base de données propriétaire en C (avec interface C++) et une bibliothèque graphique " faite maison " en C et assembleur. Pour compléter cela, la caissière disposait d'un poste sous

Projet ADP : modifier le moins possible l'application

Un projet, c'est le choix d'un ensemble de composants, plus, éventuellement un développement spécifique, et enfin l'intégration de l'ensemble. Le calcul du coût est donc simple, c'est la somme du coût des composants, du coût des développements, et du coût de l'intégration.

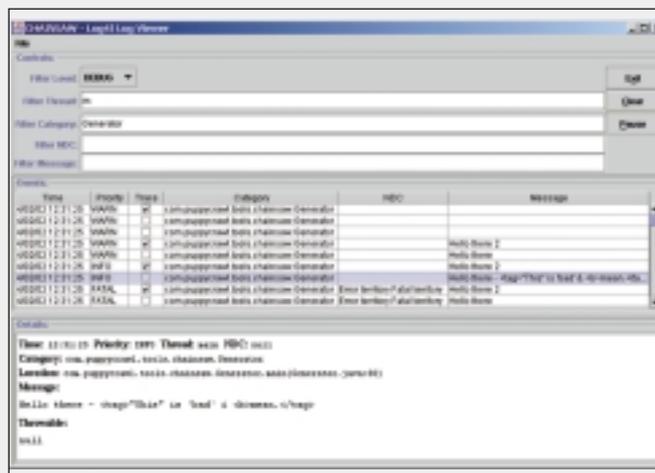
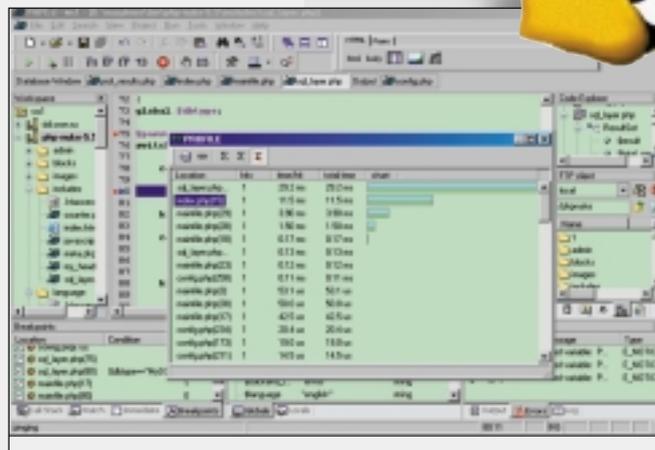


À partir de là, vous pouvez imaginer tous les compromis :

- composant Propriétaire / composant Open Source → le composant Propriétaire n'est intéressant que s'il permet d'économiser sur les développements (j'inclus la "prise en main" des composants inconnus au début du projet dans les développements)
- composant "sur étagère" / développement spécifique → Un développement spécifique est parfois moins coûteux (à court terme, puisque le logiciel a aussi un coût de maintenance) que la prise en main d'un composant sur étagère généraliste.

La deuxième première contrainte (ex-aequo), c'est le délai, mais c'est indirectement compté dans le coût, par l'intermédiaire de pénalités de retard.

Le point de vue technique maintenant... Objectivement, et en tout cas si l'on se restreint à des composants mûrs, le composant sur étagère l'emporte presque toujours sur le développement spécifique, parce que les développeurs qui l'ont fait ont certainement résolu des problèmes auxquels on ne pense pas d'emblée. Pourtant, personnellement, je trouve plus amusant de tout développer "from scratch", que d'assimiler et d'accepter un design que je n'ai pas choisi. Sur le compromis Propriétaire/Open Source, personnellement, je trouve insupportable de travailler avec un logiciel dont on ne dispose pas des sources (Propriétaire ne signifie pas forcément que l'on ne dispose pas des sources), parce qu'une documentation n'est pas forcément à jour, parce qu'un logiciel contient toujours des bugs.



Sur les outils, je pense qu'il ne faut pas perdre de vue qu'un ingénieur n'est pas une bête de somme, il peut donc préférer tel ou tel outil, il peut s'adapter à un outil s'il a la preuve qu'il est plus adapté que celui qu'il utilise. La valeur ajoutée d'un ingénieur n'est pas, fort heureusement, sa capacité à se servir de tel ou tel outil.

Après ces généralités (pour ne pas dire banalités), le cas particulier du projet largo (voir cas ADP). Tout d'abord, les raisons de la migration sont essentiellement techniques : le projet avait atteint les limites de la plateforme DOS, qui est elle-même moribonde (pas au sens marketing des éditeurs, qui voudraient nous faire croire qu'un produit ne remplit plus sa fonction parce qu'il est trop vieux, mais au sens où on ne développe plus guère sous DOS).

Ensuite, pour nuancer les généralités qui précèdent, un portage

laisse beaucoup moins de latitude dans le choix des composants, puisque le but est de modifier le moins possible l'application. L'application s'appuyait sur un composant propriétaire, qui avait été porté sous Linux (mais avait évolué entre temps). Pour le reste, les fonctionnalités de Linux (POSIX threads, frame buffer, driver série, accès direct en utilisant iopl/ioperm) ont pu être utilisées pour remplacer les mécanismes bas niveau sur lesquels s'appuyait l'application, en apportant la sécurité et le confort de la MMU.

■ Gilles Chanteperdix - OpenWide

Open Source : une compétence qui peut faire la différence



L'Open Source est un sujet dans le vent. Pour autant, les compétences liées à cet univers sont encore rarement recherchées en tant que telles. Si à expertises égales, elles peuvent faire la différence pour accéder à un emploi, elles ne sont ni un accélérateur de carrière ni la garantie d'une meilleure rémunération.

Les logiciels libres touchent peu à peu tous les secteurs de l'informatique. Administrateurs, mais aussi développeurs et bientôt utilisateurs, s'y intéressent. La montée en puissance des projets Open Source est incontestable. "Une à une, les différentes couches de l'infrastructure des logiciels d'entreprise sont touchées", explique Alain Delafosse, consultant chez Objet Direct, filiale de Homsys. Après Linux et Apache dans les domaines des systèmes d'exploitation et des serveurs http, c'est au tour de JBoss et de Tomcat d'être reconnus comme des projets incontournables dans le domaine de serveurs d'applications Java. Quel sera le prochain domaine à être touché ? Sans doute celui des bases de données où Oracle, DB2 et SQL Server pourront trouver une nouvelle concurrence.

Cap sur Linux et les logiciels libres

Si le marché de la formation informatique n'est pas spécialement euphorique, l'Open Source semble faire figure d'exception. La demande en formation Linux et Logiciels libres croît fortement aux dires des professionnels du secteur. Les entreprises commencent à s'équiper en solutions Open Source et il leur faut donc former leurs informaticiens à ces nouveaux systèmes. Les formations sur l'administration des systèmes Open Source sont historiquement les plus anciennes. "Actuellement les for-

mations Java, Tomcat, Struts ont la cote, confirme Cyril Pierre de Geyer, responsable des formations au sein de kaptive formations. Ces demandes sont une grande constante depuis plusieurs années et s'accroissent depuis deux ou trois ans malgré l'arrivée de .Net. Elles sont fortement plébiscitées par des profils techniques de type ingénieur ou chef de projet." A noter également l'arrivée de demandes autour de MySQL et Postgresql pour l'administration et l'utilisation des bases de données. La demande progresse aussi largement depuis ces deux dernières années autour de la plateforme PHP. "Sans oublier Linux Administration, pour qui la demande est intense depuis deux ans, précise-t-il. De nombreux informaticiens spécialisés dans le langage propriétaire cherchent à se former sur cet outil."

Un accélérateur de carrière ?

"Les carrières "Open Source" sont les mêmes que pour un développeur classique, elles répondent de la même manière au jeu de l'offre et de la demande, estime Cédric Barbier, directeur marketing du site lesjeudis.com. Il est certain que plus les entreprises utiliseront l'Open Source, plus ces informaticiens seront demandés." Pour ce qui est des salaires, c'est la même chose, il n'y a pas vraiment de différence entre un développeur ASP et un déve-

loppeur PHP. "En fait, deux profils sont à distinguer, précise Rodolphe Quiédeville, fondateur de Lolix. Ceux qui travaillent dans des SSII qui ne font que de l'Open Source et dans ces cas là, leurs compétences Open Source ne sont qu'une normalité. Et tous les autres, pour qui ce type de compétence constitue une corde de plus à leur arc. Ce ne sera pas à proprement parler un accélérateur de carrière mais certainement un " plus " pour décrocher un poste actuellement." Les entreprises embauchant à moindre frais, il n'est pas rare de voir se développer des offres de stage. "C'est une réalité criante dans la sphère de l'Open Source, constate-t-il. Il n'y a pas pléthore d'offres d'emplois dédiées, et à leur côté coexistent des propositions de stage exigeant des compétences que seules 5 ans d'expérience sauraient valider." En conclusion, l'Open Source se déployant, il est judicieux de s'ouvrir à cet univers et de se former, avec toutefois un bémol : "Tout évoluant extrêmement vite, ce type d'avance n'a pas un effet durable, tempère Bertrand, Guillin coordinateur technologique du Groupe SQLI. Certes, cela reste un atout supplémentaire, mais l'expérience du collaborateur et son développement potentiel sont une meilleure garantie pour les entreprises qui recrutent actuellement."

■ Anne-Françoise Moal

Kaptive, l'esprit Open Source



Cyril Pierre de Geyer est chef de projet et responsable des formations de la branche Kaptive Formations. Il nous donne ici un aperçu des profils qu'il recrute pour sa SSII

et des tendances de la formation liée à l'Open Source.

Kaptive est un studio de développement informatique spécialisé dans les technologies Internet et la création de sites web. Cette SSII permet aux entreprises, en direct ou en sous-traitance, de profiter des avantages du logiciel libre. Les profils d'informaticien ayant une compétence Open Source sont les bienvenus. "En début de mois, nous recherchions, trois profils d'informaticiens maîtrisant PHP, explique Cyril Pierre de Geyer. Principalement pour travailler avec les services WEB et l'interfaçage avec des systèmes d'informations existants. La demande de profils Java, J2EE reste une constante et cela, malgré l'arrivée et le tapage médiatique autour de ".Net ". Tous les salariés de Kaptive bénéficient d'une formation autour des compétences Open Source. "L'esprit Open Source est également présent chez nous au travers d'une participation active aux efforts communautaires, souligne-t-il. Contrairement à ce que certains pourraient penser, ce n'est pas du temps volé à nos projets en interne car le retour est toujours au rendez-vous et nous permet de progresser."

Est-ce que ces compétences liées à l'Open Source sont désormais incontournables ? Selon lui tout dépend des secteurs. "A titre d'exemple, dans le monde de la bureautique, il est clair que Linux a encore des progrès à faire par rapport à Microsoft. Actuellement, maîtriser Open Office est moins intéressant sur un curriculum vitae que maîtriser Ms Office. A l'inverse, dans le monde des serveurs, Linux est très bien établi et des connaissances de Linux Apache Perl PHP ou Python seront plus intéressantes. Apache représentant plus ou moins 60% du marché sur les serveurs Web, et PHP étant installé sur 30% du parc mondial

de serveurs Web, devant ASP de Microsoft". Quant aux évolutions de carrières, il ne manque pas de rappeler que la maîtrise de l'Open Source est plus orientée sécurité et systèmes d'informations, mais que globalement, une personne maîtrisant l'Open Source aura la même carrière que celle maîtrisant des technologies propriétaires. Son conseil : "il ne faut surtout pas s'enfermer dans un monde ou dans l'autre. Chacun a ses avantages et ses inconvénients." Quant à un différentiel possible entre les rémunérations liées aux compétences de l'univers propriétaire et Open Source, il ne décèle aucune différence notable.

■ AFM

Le point de vue de... Bernard Lang, directeur de recherche à l'INRIA



Actuellement, le marché des serveurs est très largement ouvert à l'Open Source, cela génère donc des emplois avec ce type de compétence. Il en va de même en ce qui concerne le temps réel et les systèmes embarqués, où l'Open Source se développe rapidement. En fait, Linux ou BSD étant des variantes d'Unix, les formations sont essentiellement les mêmes pour les systèmes Open Source et les Unix propriétaires, ce qui représente environ 60% du marché des serveurs – contre 40% à Microsoft – si l'on y inclut Mac OS X, dont le cœur est un Unix BSD, donc Open Source. IBM, Novell ou encore Bull développent une stratégie résolument Linux. Les activités réseau restent majoritairement dans les environnements

Unix/Linux/BSD, et pour les serveurs web, le logiciel libre Apache vient de passer la barre des 67% du marché. Une bonne connaissance de Linux ouvre des horizons, car de nombreuses entreprises réfléchissent à un passage à l'Open Source. Elle est sans doute même indispensable dans le monde des réseaux.

Dans le secteur de l'informatique embarquée, les contraintes du marché ont créé une nette préférence pour les systèmes libres, au besoin en finançant leur développement. A titre d'exemple, nous pouvons citer eCos, qui est un système d'exploitation en temps réel libre, développé par Cygnus, société maintenant intégrée à Redhat. Il a été entièrement financé par un consortium de grands groupes utilisateurs. Pour un professionnel de l'informatique embarquée, les systèmes libres sont incontournables, même si la variété de l'offre libre et propriétaire reste grande.

Compte tenu de la domination de Microsoft sur le poste de travail, il y a sans doute globalement autant de jobs autour de Windows qu'autour de Linux ou Unix. La situation peut cependant évoluer. De grands industriels comme Alcatel ou Thalès envisagent d'étendre leur usage de Linux, certains ayant déjà commencé, de même que nombre d'administrations. Il y a clairement une réaction à ce qui est perçu comme une dépendance excessive des choix techniques ou économiques de Microsoft, réaction qui concerne même le poste de travail et la bureautique.

■ *Propos recueillis par Anne-Françoise Moal*

Pour aller plus loin...

INRIA : Institut National de Recherche en Informatique et en Automatique
www.inria.fr

AFUL : Association loi 1901 d'utilisateurs et de professionnels du logiciel libre
www.iful.org

APRIL : Association pour la promotion et la recherche en informatique Libre

Fr.lolix.org : forum d'emploi spécialisé dans les logiciels libres

Fr.joinux.org : site spécialisé dans les métiers ayant trait aux systèmes d'exploitation à base de noyau Linux spécifiquement et la famille des Unix libres plus généralement comme FreeBSD, NetBSD et Open BSD.

Nouvelles méthodes de développement

Êtes-vous au top ?

XP, MDA, RUP, SCRUM, UP, SOA, les notions de méthode de conception et de développement ne manquent pas. On les qualifie de "méthodes de développement agiles". Elles doivent fournir un développement plus harmonieux entre les utilisateurs finaux, les concepteurs et les développeurs et améliorer le taux de réussite des projets arrivant à terme.



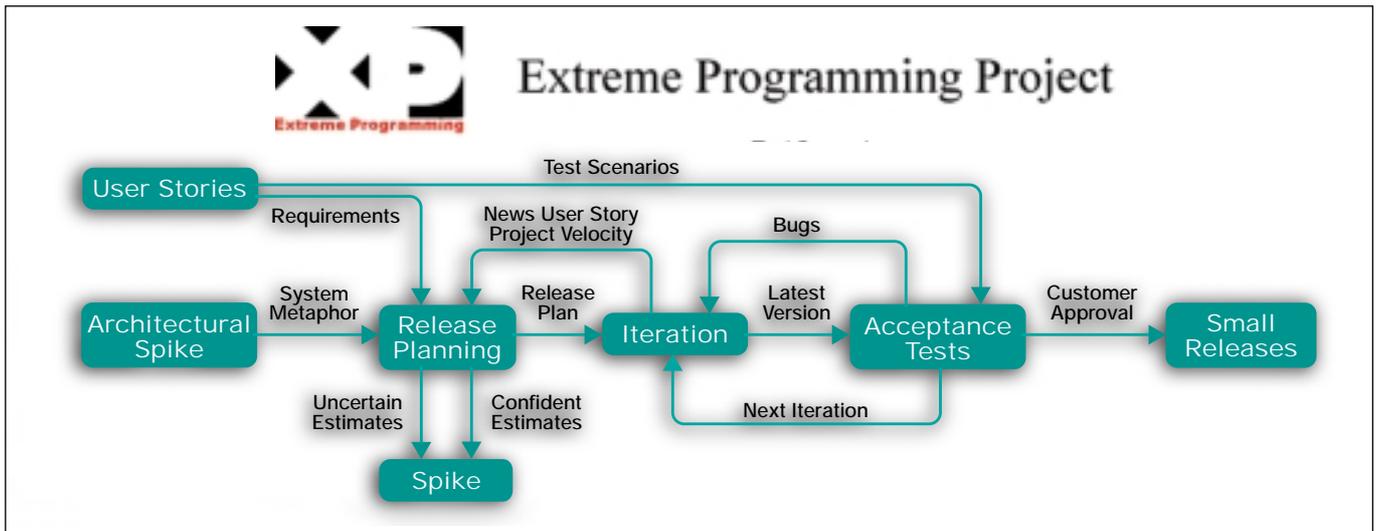
Aujourd'hui, il est clair que les utilisateurs ne peuvent pas concevoir eux-mêmes des applications complexes. Les langages et les outils étant trop compliqués à maîtriser. Seuls les petits programmes plus ou moins bureautiques, peuvent être conçus par un utilisateur avancé. Il existe toujours une certaine incompréhension entre les utilisateurs, le développeur, le concepteur et le cahier des charges. Il faut alors rendre cette communication plus souple, plus fluide, plus réactive. Plus que jamais, il y a nécessité à partager l'information et les connaissances. La réussite d'un projet (petit ou grand) dépend d'une multitude de facteurs techniques, technologiques, fonctionnels mais aussi humains.

Un développement s'il n'est pas structuré et ordonné, dérape rapidement. Ensuite, difficile de rectifier le tir. Rien n'est moins aisé que de respecter les délais et les demandes fonctionnelles. Nous avons volontairement mis l'accent sur trois méthodes : RUP, XP et SCRUM. Ces méthodes de développement "agiles" sont très utilisées. Chacune répondant à des problématiques précises et différentes. Pourquoi utiliser de telles méthodes ? C'est avant tout une question d'efficacité, de maîtrise et d'adaptation du développement. On peut faire le rapprochement avec la notion de projet itératif : être capable d'adapter le projet en cours de route, selon les nouvelles fonctions ou les nouveaux besoins. Comparées aux

anciennes méthodes (cycle en V ou Merise), les notions agiles privilégient l'interaction et les individus. On tente de donner un modèle défini, pour mener à bien le projet et mieux intégrer les différents intervenants en parlant leur langage. Le but final est toujours le même : une application fonctionnelle répondant aux demandes, une documentation technique, le respect des délais et des coûts. Naturellement, c'est plus facile à dire qu'à faire...

XP replace le développeur sur le devant de la scène

L'eXtreme Programming (XP) est-il le cauchemar des directeurs informatique ? Est-il un bonheur pour le développeur quel qu'il soit ?



Force est de constater que le XP a su devenir en peu de temps une méthode de référence ou tout du moins, quasi incontournable. Les détracteurs du XP mettent en avant le manque d'outils de contrôle et de structuration de la méthode. Et si le XP est mal contrôlé, il aboutit à l'effet inverse de son objectif : gaspillage de ressources et de temps. En fait, le XP diffère des autres méthodes en remettant le développeur sur le devant de la scène, car celui-ci est au cœur de la programmation extrême.

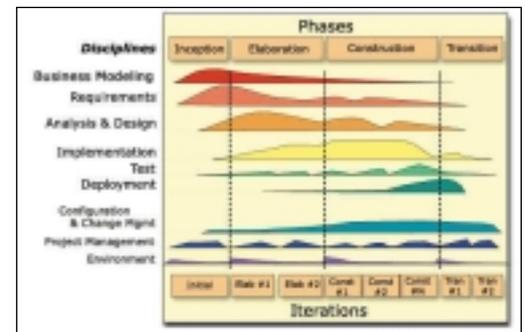
Le fonctionnement du XP repose sur les échanges de vues entre le client et le développeur sur le projet, un code simplifié, un processus incrémental. À cela, on ajoute aussi un meilleur partage de la connaissance, grâce à une communication active et constante. En fait, le développeur est au cœur du XP. Le but n'est pas de fournir un projet complet mais de se concentrer sur l'essentiel, les fonctions de base prioritaires du projet. L'objectif est avant tout de fournir rapidement une base fonctionnelle. Cela peut être très utile dans le cadre d'un maquetage applicatif, afin de juger telle ou telle fonction et de la corriger si besoin était.

Le partage de connaissance est un élément central du XP. Pour Kent Beck, le créateur de la méthode, il est inutile de faire travailler un développeur 10 heures sur une fonction, il est plus intéressant de faire travailler deux développeurs durant 5 heures chacun. Que doit changer cette " organisation " : le développeur sera plus productif et surtout, il y aura un bien meilleur partage de la connaissance du code source... Ce dernier point est non négligeable dans la maintenance future du projet.

XP, comme dit plus haut, repose aussi sur une communication directe entre le client et le développeur. Il y a tout d'abord un partage de facto des rôles de chacun : le client prend les décisions fonctionnelles, le développeur a les décisions techniques. Dans le but d'aller vite et de se concentrer sur le développement pur et dur, il y a contact direct entre les deux parties. Cette communication favorise bien entendu le retour d'expérience mais surtout, elle permet une réactivité optimale.

XP se focalise donc sur la productivité du développeur. Il permet de coder et de tester des fonctions en très peu de jours (on développe

uniquement des petits morceaux). XP se concentre donc uniquement sur le code et non sur l'environnement global du projet comme le fait un RUP qui se focalise sur l'utilisateur et la validation. XP permet aussi de tester et de valider bien plus rapidement les fonctions. Bien que jeune, XP n'en est pas moins une méthode originale de développement, tout en proposant une adaptabilité maximale aux



changements. Surtout, XP peut être utilisé par tous les développeurs, même si cela nécessite une rigueur de codage et la compréhension du fonctionnel final de l'application. XP permet d'aller vite mais il ne faut pas pour autant négliger la qualité du code.

L'avis de Philippe Marot

CONSULTANT SENIOR OBJET – NSK TECHNOLOGIES

La méthode XP est très intéressante pour les petites équipes, mais rien n'est défini pour les projets de taille importante. La communication entre les différents acteurs est primordiale. La méthode ne peut s'appliquer si le client n'est pas assez disponible pour participer aux différentes étapes du projet. Les fréquents retours entre l'équipe et le client lui assurent une livraison de l'application à chaque itération, conforme à ses attentes.

Le travail en binôme, qu'il soit sur la même machine ou à distance (toujours un développeur-un testeur), procure au code une qualité et une robustesse accrue par rapport aux autres méthodes.

Le rituel UML

Difficile aujourd'hui de ne pas faire intervenir d'une manière ou d'une autre UML dans un projet. Ce n'est pas à proprement parler une nouvelle méthode de développement, mais, il est souvent utilisé. UML sert à la modélisation. Un grand nombre d'éditeurs intègre un modèleur UML directement dans leur IDE. Dans le cycle de vie, UML a une place importante. Grâce à lui, on peut définir le modèle (en quelque sorte le squelette) de son application

et générer ainsi le code de base nécessaire. Cependant, UML, comme les autres méthodes, il faut le maîtriser et établir des modèles clairs et compréhensibles. UML oblige à une réflexion sur son modèle. UML a un bénéfice s'il est compris et bien utilisé.

RUP qui peut

RUP (Rational Unified Process) a été créé par l'éditeur Rational. Il s'agit d'une méthode centrée sur l'architecture itérative et incrémentale. Il repose sur les composants applicatifs et propose des contrôles de qualité. Théoriquement, RUP ouvre aussi le management des besoins initiaux et changeants de l'utilisateur. Grossièrement, il est possible de définir 4 motivations envers RUP :

- 1 les erreurs coûtent cher, il faut tenter de savoir ce que l'on veut avant tout développement concret
- 2 Tout bon système, tout bon projet, toute bonne application, repose obligatoirement sur une architecture adaptée, évolutive et saine.
- 3 Livraisons régulières de releases du projet, afin de vérifier la bonne conduite du projet et d'avoir un retour utilisateur très rapide
- 4 Intégrer les changements avec l'itération et l'incrémentale, afin d'adopter rapidement tout changement fonctionnel et/ou stratégique

La vie de l'application (que l'on peut aussi nommer cycle de vie de l'application) s'articule autour de plusieurs cycles : développement, amélioration, maintenance. À cela s'ajoutent des phases d'itérations et enfin un certain nombre d'activités formalisées pour chaque phase de la méthode.

On apprécie RUP car il s'agit d'une méthodologie fournissant une visibilité sur le projet aux différents intervenants. Surtout, RUP offre une définition claire du ou des rôles de chacun. De plus, RUP (s'il est bien implémenté et maîtrisé) donne un suivi du projet et du respect des besoins initiaux. Ainsi, tout au long de la conception et du développement, on peut facilement détecter toute anomalie ou dérapage. Comme nous l'avons déjà écrit dans Programmez !, le cycle de vie d'une application est une nécessité et les principaux éditeurs d'outils de programmation l'ont compris et intégré dans leur stratégie. Cependant, RUP apparaît un peu lourd et surtout, cette méthode laisse peu de place à l'initiative.

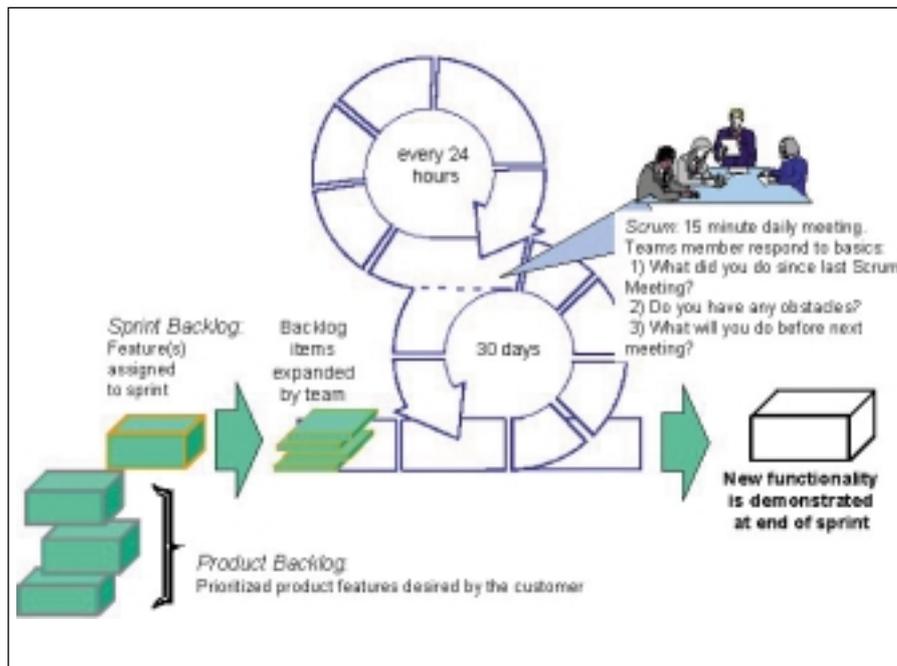
RUP dérive en fait d'UP (Unified Process). UP répond aux attentes suivantes :

- qui participe au projet

- qu'est ce qui est produit durant le projet
- comment doit-il être réalisé
- quand doit-il être livré

UP comme RUP est un processus de développement logiciel. Il est itératif et incrémental. UP se base sur UML pour créer les plans d'élaboration et de construction. Pour ce faire, on utilise des cas d'utilisation. Il "travaille" en étroite lien avec l'architecture. UP propose 4

d'un projet, gros ou petit, nécessite une grande rigueur dans l'approche et la résolution des problèmes. Bien gérer le côté humain, c'est sans aucun doute aider à la réussite du projet, d'où un succès croissant pour SCRUM. Une des utilisations classiques de SCRUM est de la coupler avec une programmation XP. Ainsi on allie la gestion du facteur humain à la rapidité de développement. Grâce à cela, il est possible



phases : inception, élaboration, construction et transition. Chaque phase se découpe en itération. Côté release, UP propose deux types : livraison d'ingénierie, livraison de gestion. On a l'habitude de définir avec UP les 4 P : personnes, projet, produit, processus. À cela, s'ajoutent les outils. UP et RUP ne sont pas exactement les mêmes méthodes. RUP propose une discipline de modélisation métier et diverses subtilités absentes d'UP. RUP fournit une plus grande précision dans la conduite du projet et sa définition. Les deux méthodes ne proposent qu'un cadre générique qu'il faut savoir adapter à ses besoins.

SCRUM

Pourquoi parler ici de SCRUM, alors qu'il s'agit plutôt d'une méthode de management et non d'une pure méthode de développement ? Car, au-delà des aspects techniques du projet, il ne faut jamais oublier le côté humain du développement. Finalement, un projet est autant un développement, qu'un projet humain qu'il faut savoir gérer. Gérer les ressources humaines

d'envisager avec le couple XP – SCRUM des projets importants, qu'avec le seul XP, il serait hasardeux de réaliser.

Que représente SCRUM ? Il faut retenir trois points principaux :

- 1 Les changements étant difficiles à réaliser, il faut les identifier le plus tôt possible
- 2 Le développeur sait programmer, laissez-le faire son travail
- 3 Faire des itérations de 30 jours. Cela permet de laisser du temps aux développeurs de produire assez de code sans prendre en défaut le management.

Comme RUP, SCRUM est un processus incrémental que l'on peut utiliser pour développer une application. Il "facilite" ou tout du moins aide à gérer les problèmes et conflits humains. SCRUM met en avant la communication et la coopération, via des réunions très régulières et des releases à intervalle régulier...

MDA, SOA : concurrent ?

Depuis quelque temps, la presse informatique n'a qu'un seul à mot à la plume : SOA (Service

Oriented Architecture). Et MDA (Model Driven Architecture) ? L'aurait-on déjà oublié ? Que se soit SOA ou MDA, ces deux " méthodes d'architecture " sont promises à prendre une part importante dans le futur, sur la définition des projets et applications. Ces méthodes " new generation " visent l'architecture globale du système d'information, rien de moins ! MDA est officiellement supporté par OMG et fournit une approche ouverte et indépendante utilisant des standards. C'est un méta modèle. MDA sépare la logique applicative métier et la partie purement plate-forme technique. Bref, MDA vise tout simplement à rendre indépendante l'application, en mettant en avant trois notions : portabilité, interopérabilité et réutilisation. Ainsi, avec MDA, on définit son système, sans se soucier de la plate-forme technique supportant au final l'application. À partir de là, on utilisera les modèles MDA (ex. : CIM, PIM) pour implémenter l'application sur la, ou les plates-formes cibles.

SOA est plus médiatique que MDA en ce moment. SOA n'a pas pour objectif de remplacer une autre méthode. SOA doit permettre de réutiliser l'existant et de le transformer en services plus agiles. Soit ! SOA peut se résumer à quatre préceptes :

- 1 Assurer un faible couplage entre le code et les aspects métiers.
- 2 Modèle distribué. Il est possible qu'un service d'une application soit physiquement réparti sur différents systèmes.
- 3 Les services sont invocables et publiables quel que soit le système utilisé
- 4 Orientation métier

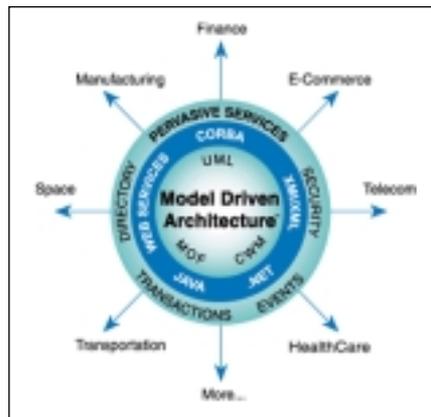
En soi, SOA n'apporte aucune nouveauté dans son concept et va moins loin que MDA. Surtout, SOA convient mieux à un projet Web Services ou à une application distribuée que dans le cadre d'une application desktop ou mobile. De toute manière, SOA emprunte des aspects propres à MDA et à UML. SOA a pour but de fournir des services les plus universels possibles. Un service SOA est une unité de travail accomplie par un fournisseur de service à un client. Le concept SOA est donc plutôt abstrait. Il s'oppose totalement au modèle de l'OO qui oblige à faire du binding. Cependant, même en programmation OO, il est possible de découpler les couches et de distribuer des composants, sans pour autant avoir besoin de SOA. De plus, il existe au moins deux grandes contraintes : utiliser un jeu réduit d'interfaces

(universelles) et des messages (selon un schéma de communication défini).

On peut avoir l'impression que l'on tente de limiter la portée de SOA à la simple notion d'intégration. Même si SOA a un fort rôle à jouer dans l'urbanisation du système d'information via les services réutilisables, SOA n'a pas à être limité à l'intégration. Encore faut-il que les environnements SOA implémentent totalement les promesses de départ, et ça, ce n'est pas encore totalement le cas... Surtout, il faut inclure une architecture SOA dès le départ et ne pas sous estimer le surcoût que cela peut engendrer. D'autre part, SOA n'a de sens que dans une utilisation à long terme du projet. Donc, il ne faut pas s'attendre à un ROI rapide.

Combiner plusieurs méthodes : attention aux faux espoirs

Aucune méthode n'est universelle et ne gère tous les aspects d'un projet. Est-il pour autant possible de combiner plusieurs méthodes tout en restant réaliste ? Finalement, chaque



méthode a des avantages et des inconvénients. RUP sait manager les projets importants, XP fournir le code et SCRUM s'occupe de l'humain...

Chaque méthode est suffisamment complémentaire pour pouvoir en combiner plusieurs et couvrir ainsi un large spectre des besoins dans le cycle de développement de l'application. Ainsi, on peut parfaitement envisager d'utiliser dans une même projet : du RUP pour la gestion au quotidien, de l'UML pour modéliser, XP pour coder le cœur des fonctions, du MDA (voire du SOA) pour l'orientation du logiciel. Le tout intégré dans un environnement de cycle de vie. Sans oublier d'y greffer l'architecture, les utilisateurs, les tests, la documentation et la maintenance !

Le dire ne représente pas un risque énorme, mais mettre tout cela en pratique est une autre paire de manches ! Chaque méthode nécessite plusieurs impératifs de départ :

- Pourquoi utiliser cette méthode ? A quoi va-t-elle concrètement servir ?
- La méthode répond-elle à mes impératifs techniques, humains, fonctionnels et technologiques ?
- Comment intégrer et mettre en œuvre la méthode ?

Si UML est maintenant largement répandu dans le processus de modélisation, ce n'est pas encore le cas pour les autres méthodes. Il ne faut surtout pas se précipiter. La précipitation est un facteur d'échec dans un projet et dans tout développement. L'implémentation d'une méthode dans un cycle de développement doit être analysée, testée, éprouvée par les responsables et les développeurs. De toute manière, comme nous l'avions précisé dans notre dossier " développer en équipe", un projet est un tout, où chaque élément s'ajuste aux autres. Donc, dans cette logique aussi bien technique, que technologique et humaine, la méthode de développement / conception n'est qu'un élément. Il faut que les outils du cycle de développement soient capables de supporter les méthodes choisies et que les développeurs et les chefs de projet et architecture soient aptes à les utiliser. On ne s'improvise pas développeur XP et encore moins docteur ès RUP ou MDA, par sa seule volonté...

La réussite d'une méthode dans un projet dépendra bien entendu de sa pertinence mais aussi, de l'aptitude des personnes à l'utiliser. Une méthode mal maîtrisée aboutira à coup sur à la faillite du projet, au mieux, à une application non conforme. La maîtrise de la méthode est un élément capital. Si cela nécessite une semaine de formation intensive, mieux vaut s'y soumettre. Une phase d'explication, aussi bien à la direction informatique qu'aux développeurs, permettra d'intégrer la méthode dans les consciences et d'avoir un échange de vues. Il ne faut pas heurter les développeurs ou un architecte. Si tel est le cas, la méthode sera mal utilisée, voire snobée, tout le contraire du but recherché. La méthode nécessite aussi un suivi attentif. Dans le cas du XP cela est encore plus criant, car on produit rapidement du code. Il faut éviter toute dérive.

Communiquez !

Au bout du compte, on dispose de tout plein de bonnes choses originales pour son projet et le mener à la réussite. Les personnes demeurent tout de même la clé incontournable du succès ! Forcer les acteurs du projet à utiliser du XP, de la MDA ou du RUP ne sert strictement à rien, bien au contraire. Ces méthodes apportent certes une vision globale et plus précise, mais complexifie de facto le modèle d'architecture projet à mettre en place et donc à gérer. Avant toute utilisation d'une méthode, on ne le répétera jamais assez : jouez la communication, approche informatique. Il faut que l'équipe du projet adhère aux éléments techniques de la gestion. En théorie, avec ces méthodes, on peut espérer, à terme, des économies et une meilleure productivité, mais

attention, n'attendez pas de miracles immédiats. Si dans l'abstrait, ces nouvelles méthodes posent les pierres d'une nouvelle programmation des applications, la réalité demeure éloignée de cette séduisante théorie. Pour une simple PME, parler de RUP ou XP est sans aucun doute inutile dans la majorité des projets. Ces structures souhaitent juste concevoir des applications rapidement, à un coût minimal, sans réflexion forcément approfondie. Par contre, les responsables de PME sont très sensibles à l'écoute " rapprochée " des utilisateurs : les cas d'utilisation apportés par UML sont donc un très bon outil pédagogique. A l'inverse, le Grand Compte et son équipe informatique cherche davantage à piloter ses multiples projets et à normaliser le plus possible les relations MOA-MOE. Il ne faut pas hésiter

dans ce cas précis, à mettre en avant RUP qui apparaît maintenant mature. Par contre, XP est plus difficile à défendre surtout quand l'approche " deux programmeurs " sur une fonction donnée est discutée : les responsables des études ont encore du mal à comprendre que ce processus XP est réellement une grande force et non un gaspillage de ressources. Utiliser RUP ou XP fait-il faire des économies ? Cela est évident si tous les acteurs acceptent les règles. On ne peut que regretter que XP soit timidement implémenté à des projets non stratégiques. Or, ce n'est pas ainsi que l'on peut prouver à la hiérarchie la pertinence des nouvelles méthodes.

RUP a été testé sur de gros projets et a prouvé son intérêt.

■ François Tonic

Un vrai travail de réflexion sur les pratiques et les outils de développement



Qui mieux que Nasser Kettani (marketing manager EMEA, IBM Software Group Rational Software) pouvait répondre à nos questions sur les nouvelles méthodes de développement ? Plongé dans le monde RUP, UP, XP depuis des années, Nasser Kettani a enseigné à l'Université de Paris-Dauphine le génie logiciel et a publié chez Eyrolles : "De Merise à l'UML".

Programmez ! : Parmi les nouvelles méthodes de développement, quelle est pour vous la plus utilisée ?

Nasser Kettani : Tout dépend ce que l'on entend par méthode. Une méthodologie en génie logiciel couvre l'ensemble du cycle de développement. C'est le cas d'UP et de RUP. UP est la plus utilisée en tant que méthode, ou plutôt comme cadre méthodologique. Développer des logiciels est un métier comme un autre. Il nécessite des processus, une démarche, de l'ingénierie. Durant longtemps, les directeurs informatique n'ont pas considéré ces besoins. Dans le développement logiciel, on trouve des gens pour développer, pour tester, pour documenter, etc. La méthode de développement englobe l'ensemble des besoins de chaque corps de métier. C'est pour cela que nous avons besoin d'une méthode

globale. Il y a encore une dizaine d'années, l'industrie n'en avait pas. À l'époque, on parlait volontiers de processus pour les activités : tests, conception, UP est l'unique méthode globale de bout en bout. Toutefois, j'y ajouterai un petit bémol. On trouve des processus de développement logiciel spécifiques à certaines industries. Par exemple à l'Agence Spatiale Européenne, au ministère de la Défense... Mais finalement, cela ne règle pas le problème des différents métiers. Ces méthodes définissent un cadre général du projet. Une méthode globale intègre tout le cycle et les spécificités de chaque métier.

P ! : XP est une méthode de développement très en vogue actuellement. Mais est-il adapté à tous les types de projets ?

NK : Non. XP est très bien pour de petits pro-

jets. Par contre, pour des projets plus importants, RUP est tout indiqué. C'est aussi pour cela que l'on a créé un sous ensemble RUP pour XP. Je reviens un instant sur UP et RUP. UP est un cadre méthodologique. Rational a repris " ce socle " pour en tirer un produit, RUP. Au même moment, XP est né. Il est issu de la même idée. XP part du développeur. Il s'agit d'une méthode de développement rapide. Il y a environ trois ans, on a défini une instantiation de RUP en XP. Nous pensons que XP est en fait un sous-ensemble de RUP. On conçoit en quelque sorte que XP est comme UP. On a alors taillé dans RUP pour pouvoir y faire du XP !

P ! : Rational est le créateur de RUP héritant d'UP. De quelle manière pourriez-vous le définir ?

NK : Je dirais que RUP est comme un énorme manuel de cuisine. Avec lui, on peut tout cuisiner. Il couvre l'ensemble des problèmes d'un projet. Les gens peuvent prendre RUP et utiliser tel ou tel sous-ensemble. C'est pour cela que l'on a voulu le rendre le plus configurable possible. RUP définit les différents métiers.

P ! : Ok, RUP est une méthode globale pour développer un projet informatique. Mais

aujourd'hui la réalité n'est pas aussi joyeuse que cela. L'entreprise l'a-t-elle compris ? L'intègre-t-elle ?

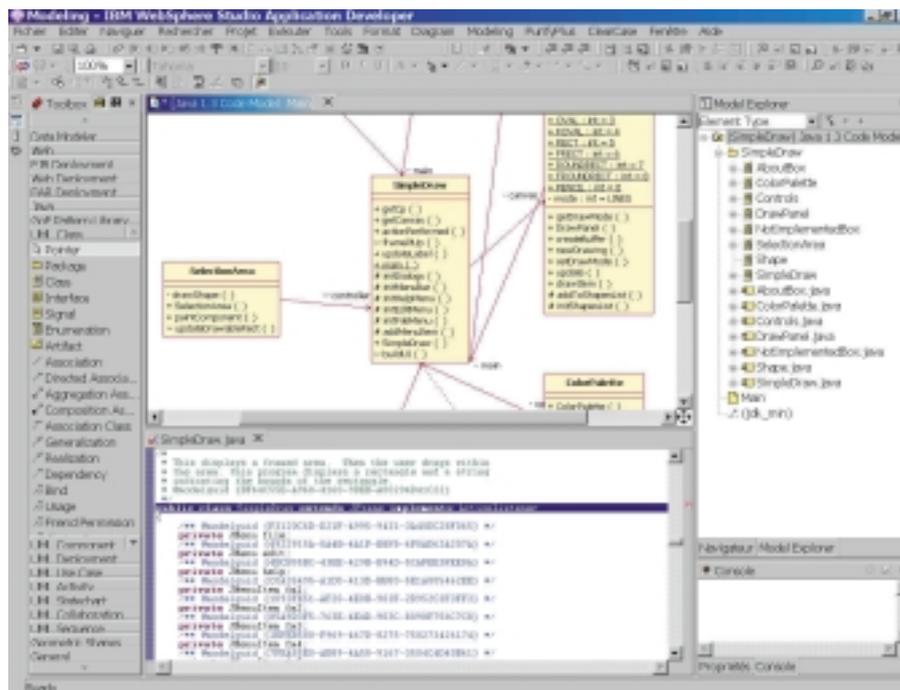
NK : Je reprendrais votre première question sur la méthode la plus utilisée. Je dirai que la méthode la plus utilisée est le couteau suisse ! Une sorte de bricolage... En même temps, les DSI ne cessent de dire que les projets sont en retard, coûtent cher ou ne répondent pas aux besoins. Répondre à ces problèmes passe nécessairement par la mise en place de pratiques et processus de développement logiciels, il faut cesser de "bricoler"

P ! : Certes. Mais le coût des licences d'outils de méthodologies ou de cycles de vie n'est-il pas un facteur dissuasif ? Est-ce un faux débat ?

NK : Prenons un exemple. Sur 100 dollars d'un projet informatique, l'achat des produits ne représente que 5 % ! Vous me direz, où sont passés les 95 % restant ? Principalement dans la masse salariale ! La vraie question serait plutôt la suivante : en investissant 5 % peut-on être plus efficace, plus productif ? Si le produit choisi apporte moins de 5 % de productivité, ce n'est pas la peine de l'acheter. En même temps, est-il envisageable dans une entreprise que les employés utilisent des outils de messagerie ou de bureautique différents et non compatibles, ou que les commerciaux utilisent des outils de CRM différents ? Et pourtant, c'est ce qui se passe avec les développeurs. Les DSI doivent opérer un vrai travail de réflexion sur les pratiques et les outils de leurs équipes de développement

P ! : Si on regarde le taux de réussite des projets informatiques, on ne peut pas dire qu'il soit très glorieux. Les 3/4 sont, soit en retard, soit non conformes, soit tout bonnement arrêtés. Souvent, on s'aperçoit qu'il y a une mauvaise prise en compte des besoins et on ne pense pas toujours à faire évoluer le projet au fur et à mesure. Les nouvelles méthodes évoquées plus haut répondent-elles à de telles problématiques ?

NK : Oui. UP, RUP, XP permettent de prendre en compte les changements des besoins. Mais ce n'est pas uniquement un problème d'outils. Les outils automatisent des processus métiers.



Ils sont là pour nous aider. Si on a des outils de sécurité, mais que l'on n'a pas de politique de sécurité, ces logiciels ne serviront à rien ! Les entreprises sont demandeuses d'outils de méthodologie, pourvu qu'ils prennent en compte leurs spécificités ... Un éditeur comme Rational, doit répondre à cela

P ! : Si je comprends bien, l'outil ne fait pas le moine. Et pourtant, les entreprises utilisent parfois à tort et à travers ces méthodes. Car, ici comme ailleurs, la formation n'est-elle pas un facteur de réussite ou d'échec ?

NK : Nos clients ne sont pas dupes : un bon outil de CRM ne rend pas un mauvais commercial très performant. Les éditeurs ont fait un extraordinaire travail pour fournir des outils de développement logiciels (IDE) de très bonne qualité et faciles à utiliser, et il faut continuer. Mais il faut toujours former ! Au-delà de la formation, il y a l'accompagnement. Il faut aider l'entreprise à comprendre les méthodes et les outils. Nous travaillons avec nos clients pour identifier les problèmes de développement logiciel qu'ils rencontrent et nous tentons d'y répondre en mettant en place les processus et les outils adaptés. Il faut aussi impliquer les utilisateurs de ces outils. Il ne faut pas avoir

de rejet par rapport aux nouveaux outils. Un de nos axes de travail concerne l'acceptabilité. D'autre part, Rational travaille beaucoup à simplifier l'utilisation de nos environnements mais cela ne signifie pas absence totale de formations !

P ! : En dehors des méthodes purement de développement et de cycle de vie, il existe aussi une véritable lame de fond autour de l'orienté service. On constate une sorte de frénésie autour de MDA et SOA. Certains tentent de les opposer, d'autres, au contraire, parlent de complémentarité. Qu'en pensez-vous ?

NK : Je ne les mets pas en opposition. Il y a une notion fondamentale à retenir : les entreprises doivent disposer d'un système flexible qui résiste aux changements. MDA et SOA sont complémentaires. On passe d'un modèle de composant à un modèle de services, c'est notamment une des principales évolutions liées à l'arrivée des Web Services. MDA est une logique complémentaire. Si je veux un système d'information, je peux l'architecturer d'un point de vue métier. Ensuite, je génère l'application à partir d'un modèle de haut niveau, le niveau technique étant caché. Si je sais que j'utilise du WebSphere, du DB2... je peux alors modéliser l'application à très haut niveau et générer le code automatiquement. On ne peut pas générer pour une architecture que l'on ne connaît pas.

■ *Propos recueillis par François Tonic*

Quand on introduit un nouvel outil, il y a toujours un changement de comportement.

L'eXtreme Programming : les 13 bonnes pratiques

Les nombreux échecs de projets informatiques ont conduit ces dernières années à l'émergence de nouvelles méthodologies dites "agiles". La plus populaire d'entre elles, l'eXtreme Programming (XP), est souvent perçue comme une solution utopique prêchée par des informaticiens idéalistes. Il s'agit pourtant de pratiques assez simples à adopter et à adapter progressivement selon le contexte de son projet.

L'eXtreme Programming en pratiques

XP est un ensemble cohérent de 13 pratiques. Pratiques de programmation, de collaboration ou de gestion de projet, il n'est pas nécessaire de les utiliser toutes en même temps pour en tirer profit. Mais leur cohérence fait qu'on vient à l'une à partir d'une autre, et que leur association les rend plus efficaces. Nous vous invitons donc ici à les découvrir l'une après l'autre pour comprendre comment elles s'enchaînent et s'utilisent.

Au début du projet, toute l'équipe de développement définit **les règles de codage** qu'elle va utiliser. Même si les "coding standards" sont déjà familiers aux développeurs expérimentés, cette pratique permet de gommer leurs petites différences d'habitudes et de mettre les débutants à niveau. Il est aussi important d'établir un degré d'exigence pour les commentaires qui accompagnent le code. Celui-ci gagne ainsi beaucoup en homogénéité et en lisibilité, ce qui facilite considérablement sa reprise. Une fois que la syntaxe à respecter est définie, chaque programmeur en reçoit un exemple et tous veillent régulièrement qu'il en est fait bon usage. Cette pratique est très facile à initier, mais demande à être suivie avec attention tout au long du projet.

Si besoin, rien de tel qu'une séance de remaniement du code pour corriger ses défauts !

Pratiquer le **remaniement** (ou "refactoring"), c'est améliorer systématiquement le code sur lequel on repasse. Ne pas laisser de code qui ne sert plus en commentaires, nettoyer, simplifier voire reprendre la conception si elle

n'est pas satisfaisante. Pour s'en souvenir, on gardera en tête le conseil de John F. Woods : *"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live"*. Même dans une équipe animée des meilleures intentions, cette pratique a tendance à être vite écartée quand la pression monte ou que la date de livraison se rapproche. Il est donc fortement recommandé de planifier des séances fréquentes et régulières de remaniement du code pour toute l'équipe (par exemple une demi-journée par

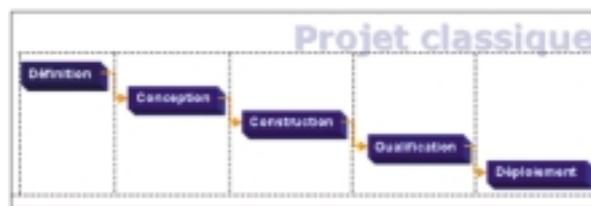


figure 1

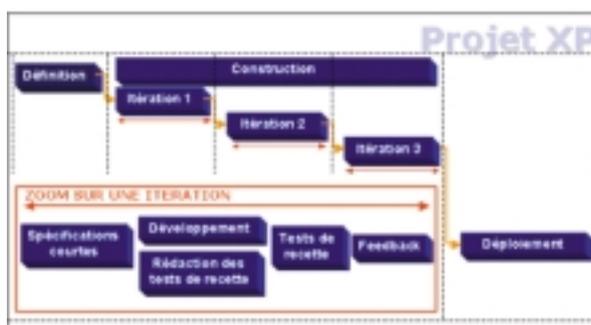


figure 2

semaine, le vendredi matin), et de ne pas y déroger. On récupère largement le temps passé à remanier le code lorsqu'on passe en phase de test ou que l'on souhaite réaliser des évolutions sur l'application.

Cela s'accorde parfaitement avec le choix de la **conception simple**, qui consiste à toujours envisager la solution la plus simple qui répond à la demande du client, ni plus ni moins. On

ne cherche donc pas à anticiper des besoins non encore spécifiés ou à développer des composants plus génériques que nécessaire. Contrairement à ce que l'on peut penser, cette démarche n'est pas évidente à suivre quand on est justement habitué à produire des composants techniques et métiers génériques et réutilisables. Pour adopter une conception simple, il faut réussir à se convaincre qu'on peut probablement perdre du temps et compliquer les développements sur des fonctions qui ne serviront peut-être jamais. Bénéficier du regard critique des autres développeurs ou du chef de projet sur sa conception du code est alors d'une aide précieuse. Plus généralement, l'échange et la communication au sein de l'équipe projet sont des valeurs clés des pratiques XP.

De là découle la fameuse et controversée **programmation en binôme** : la théorie voudrait que les programmeurs travaillent à deux sur un même poste, les binômes changeant fréquemment. Ici, notre expérience tend à prouver que les développeurs doivent travailler par paire au minimum lorsqu'ils abordent des points difficiles ou font du refactoring, mais pas nécessairement en permanence si cela nuit à la productivité de l'équipe. Le coach XP veillera donc avec attention à ce que le meilleur niveau de collaboration soit atteint entre les différents programmeurs.

La **responsabilité collective du code** est d'ailleurs là pour inciter toute l'équipe à participer à l'ensemble des développements. En effet, chaque développeur est considéré comme responsable de la totalité du code produit par l'équipe et doit le maîtriser suffisamment pour pouvoir intervenir dessus efficacement.

Cette pratique est très utile pour gérer un turnover imprévu dans l'équipe. Mais bien sûr, elle est d'autant plus difficile à appliquer que l'équipe est grande.

Autre pratique de collaboration, **l'intégration continue** a un double objectif : réduire au maximum les conflits entre les développements réalisés par les membres de l'équipe, et

disposer en permanence d'une version de référence de l'application. Les problèmes rencontrés lors de la synchronisation du code, par exemple avec un CVS (Concurrent Versions System), sont en effet bien moins nombreux et bien plus simples à résoudre si cette synchronisation est quotidienne. D'autre part, on se trouve alors toujours en mesure de livrer une version du code source intégrant les derniers développements.

Cela tombe très bien puisque procéder à des **livraisons fréquentes** est également une pratique XP. On met ainsi à disposition du client un produit qu'il peut voir et tester très tôt dans le projet. L'effet tunnel (absence de visibilité du client pendant la phase de développement) et ses effets négatifs disparaissent alors, et toute mauvaise compréhension des besoins peut être rectifiée presque en temps réel.

Pour faciliter encore les choses, le fonctionnement de l'application sera décrit à l'aide de **métaphores** utilisées par tous les participants. Comparée à d'autres pratiques, celle-ci peut paraître anodine mais elle permet de s'accorder réellement sur une compréhension commune des besoins et d'en explorer toutes les facettes... et elle est si simple qu'il serait dommage de s'en priver.

La **planification itérative** est une pratique beaucoup plus structurante de XP, à manier avec plus de précautions. Classiquement, un projet est découpé en phases successives (cascade, cycle en V... - voir [figure 1](#)). Ici, XP propose un découpage très différent : la phase de développement est divisée en itérations auxquelles la conception et la recette sont intégrées ([figure 2](#)). Concrètement, le projet se déroule alors de la manière suivante : tout d'abord l'expression de besoins n'est pas détaillée mais traduite en brefs scénarii fonctionnels. Chaque scénario représente ainsi une fonction à implémenter, mais à ce stade il ne s'agit pas d'en spécifier le contenu de manière détaillée. En revanche, on s'attache à trier cette liste par ordre de priorité. Pour cela, on commence par classer les scénarii selon le degré de nécessité exprimé par le client (la fonctionnalité la plus indispensable étant placée en premier). On révisé ensuite ce classement en tenant compte des dépendances entre scénarii : certains devront sûrement être réalisés avant d'autres, par exemple pour des rai-



sons techniques. On obtient ainsi la liste exhaustive et ordonnée de toutes les fonctionnalités identifiées au début du projet. On répartit alors les scénarii en itérations de durées égales (2 à 4 semaines en général). L'équipe projet étant généralement de plus en plus performante au fil des itérations, il est conseillé d'en tenir compte à ce moment-là en allégeant la quantité de travail prévue pour la première itération et en l'augmentant progressivement pour les itérations suivantes (à durée constante).

On rentre alors dans le processus de développement itératif à proprement parler. Chaque itération débute par une réunion de lancement



avec le client et l'équipe de développement pour détailler, chiffrer et arbitrer les scénarii à réaliser durant l'itération. On prépare ensuite les tests de recette afin de les fournir rapidement aux développeurs. L'itération est terminée lorsque tous les tests de recette sont validés.

Les avantages de ce type de planification itérative sont considérables : apporter une réponse à l'utopie des spécifications complètes et immuables puisque les scénarii sont spécifiés et affinés au fil des itérations, pouvoir ajuster le reste à faire en fonction des consommés et des délais, mettre à profit le " feedback " sur une itération pour améliorer la suivante...

Durant chaque itération, après le lancement spécifiant le détail des scénarii réalisés, le client élabore les **tests de recette** correspondants. Pour cette pratique, il peut se faire aider du coach car ces tests doivent être les plus complets et précis possibles, et être mis à disposition des développeurs avant qu'ils aient réalisé 50% des développements correspondants. Les tests de recette peuvent être confrontés à la description des scénarii pour que l'ensemble définisse parfaitement le besoin réel du client. Ce dernier est donc occupé presque en continu durant le projet : lorsqu'il ne détaille le contenu des scénarii fonctionnels avec les développeurs, il rédige des tests de recette avec le coach ou valide que l'application les passe correctement.

On comprend alors bien la raison d'être de la pratique **du client sur site**. Idéalement, le client XP intègre les locaux de l'équipe de développement pour favoriser au maximum la communication et la réactivité sur le projet. Ceci est d'autant plus important que l'on n'a pas rédigé de spécifications complètes au début du projet, et qu'il faut donc que le client soit présent pour préciser ses besoins, répondre rapidement aux questions des développeurs et réagir par rapport aux livraisons fréquentes de l'application. Si le client ne peut, ou ne souhaite être présent à temps plein sur le site de développement, il faudra compenser cette absence par tous les moyens possibles : réunions ou conférences téléphoniques très fréquentes, utilisation d'outils de collaboration, ouverture d'un accès à distance à la version de référence de l'application, etc. Si la disponibilité du client n'est pas suffisante ou que la pression qu'il va subir pour remplir cor-



rectement son rôle risque d'être trop forte, il est fortement conseillé d'allonger la durée des itérations et de revoir à la baisse la dimension de l'équipe de développement. De cette manière, celle-ci ne restera jamais en attente d'informations ou de jeux de test de la part du client et conservera un bon niveau de productivité tout au long du projet.

Quoi qu'il en soit, il est capital de sensibiliser le client au rôle particulier qu'il devra jouer sur le projet, avant même le lancement de celui-ci et le choix de certaines pratiques XP. L'engagement du client n'est pas forcément beaucoup plus important sur un projet XP que sur un projet classique, mais il est réparti de manière continue sur toute la durée du projet au lieu d'être concentré au début (définition et spécification) et à la fin (recette).

Pour exploiter au mieux les pratiques de développement vues précédemment, on pourra mettre en œuvre **le développement piloté par les tests** avec les tests unitaires. Le principe est de tester de manière unitaire les classes développées, en associant à chaque classe une classe de test qui vérifie son bon fonctionnement de manière automatique. Cela impose d'aborder le développement objet avec la méthodologie du " Test First " qui propose les étapes suivantes : écrire en premier la classe de test et vérifier que le test ne passe pas, puis écrire la classe à tester, et enfin vérifier que le test passe maintenant correctement.

La mise en place des tests unitaires n'est pas évidente : elle nécessite un outillage adéquat (JUnit par exemple), l'apprentissage d'une

méthode de codage et de test, bref un certain investissement. Mais bien utilisée, la charge de travail supplémentaire engagée a toutes les chances d'être récupérée par la suite et d'apporter des améliorations certaines. En effet, au-delà de la meilleure fiabilité des composants testés, l'utilisation des tests unitaires ne peut, par exemple, se concevoir sans l'utilisation des interfaces. Elle améliore par conséquent le design du code, et rend donc l'application plus facile à maintenir (notamment parce qu'elle nécessite de casser les dépendances entre packages). De plus, l'automatisation de ces tests permet de vérifier très rapidement la non-régression du code après y avoir apporté des modifications. Attention cependant : l'utilisation des tests unitaires n'affranchit pas d'effectuer les tests fonctionnels (voir la pratique des tests de recette).

Forte de toutes ces bonnes pratiques, une équipe XP adopte un **rythme durable**, qui lui permet de travailler efficacement du début à la fin du projet.

Organisation d'une équipe projet XP

Globalement, l'organisation d'une équipe XP est un peu différente de celle d'une équipe classique. Le rôle de chef de projet est remplacé par celui de **coach**, qui coordonne le travail de tous les participants, les aide notamment à utiliser les pratiques XP et veille au bon déroulement du projet. Le coach est présent à toutes les réunions et reste l'interlocuteur privilégié du client. Les **développeurs** participent aux réunions de lancement et de spé-

cification des scénarii au début de chaque itération, codent et testent en utilisant les pratiques de programmation. Le **client XP** participe à la rédaction et à la spécification des scénarii fonctionnels, prend part à la planification au début de chaque itération, rédige et valide les tests de recette avec l'aide du coach. Son pouvoir de décision et sa disponibilité sont des facteurs clés du succès du projet. Le **manager** participe aux réunions clés, s'informe du bon déroulement du projet et fournit les ressources nécessaires. Le **tracker** suit avec un regard extérieur le bon avancement des développements, anticipe et remonte les problèmes qui risquent de survenir. Dans une équipe réduite, ce rôle peut être joué par le coach ou un développeur.

Pour que chacun comprenne bien son rôle et puisse le remplir correctement, l'idéal est bien entendu qu'une ou deux personnes déjà expérimentées sur XP puissent former les autres au cours du projet. Si ce n'est pas le cas, mieux vaut choisir quelques pratiques pour un premier projet puis essayer les autres progressivement.

Chez SQLI, XP rime avec qualité

Quelqu'un qui ne connaîtrait pas XP et participerait pour la première fois à un projet XP pourrait se sentir un peu perdu ou inquiet de ne pas retrouver ses repères habituels. Le client peut notamment être gêné du peu de formalisation des documents utilisés (fiches scénarii, etc.), ou le manager trouver la gestion de projet un peu floue. Concrètement, XP véhicule parfois une image de manque de rigueur ou de processus mal définis. Pourtant, il n'en est rien. Nous avons, chez SQLI, entamé en 2002 une démarche de certification de qualité CMMI® (Capability Maturity Model Integration) qui repose notamment sur l'adoption de bonnes pratiques adaptées au contexte du projet. Et les pratiques XP ont naturellement été inscrites à cette initiative.

Bien comprendre les objectifs et les exigences de chaque pratique, savoir quand les utiliser et les adapter au contexte du projet nécessite bien entendu un apprentissage qui a un coût. Mais celui-ci semble très raisonnable en regard des promesses qu'il permet finalement de tenir : jouer sur le périmètre fonctionnel pour garantir le respect des charges, des délais et de la qualité des projets informatiques.

■ **Alban DALLE**, chef de projet (Groupe SQLI)

Quelques outils UML

L'apparition d'UML a profondément modifié l'approche que l'on avait dans la modélisation des projets. UML, grâce à son architecture modulaire et puissante, s'est rapidement imposée comme LA méthode de modélisation. Les principaux IDE et environnements de conception ou presque implémentent un module de modélisateur UML. Si la liste est loin d'être exhaustive, les outils UML cités vous donneront quelques possibilités.

Outils Open Source		
Nom	Système	Remarques
UML Object Modeller for Linux http://uml.sourceforge.net/	Linux	Un des meilleurs outils Linux en Open Source (fonctionne sous KDE)
Python UML Tool http://pyut.sourceforge.net/	Windows, Linux, Unix	Éditeur de classes UML écrit en Python
Dia2Code http://dia2code.sourceforge.net/	Linux	Outil permettant de générer du code à partir d'un diagramme UML. Il supporte Ada, Java, PHP, Python, C++, etc.
Jug (Java UML Generator) http://jug.sourceforge.net/	Windows	Convertisseur de classes Java en diagrammes UML.
ESS-Model http://www.essmodel.com/	Windows	Pour faire rapidement du reverse engineering à partir de code Kylix, Delphi, Java
Omondo http://www.omondo.com	Java	Plug-in de modélisation UML pour l'IDE Eclipse. Version entreprise prochainement disponible
ArgoUML http://argouml.tigris.org/	Java	Modèleur en licence BSD. Implémentation UML presque complète.
Outils commerciaux		
Simple Objects Standard Adaptive Arts http://www.adaptive-arts.com	Windows	Reverse engineering pour Delphi, Smalltalk, Java, Corba, VB, C++
Real-time Modeler Artisan http://www.artisansw.com	Windows	Modèleur temps réel avec support multi-utilisateur
Cittera CanyonBlue http://www.canyonblue.com	Windows	Outil collaboratif UML en temps réel. Idéal pour les groupes de développeurs dispersés géographiquement.
AllFusion Component Modeler Computer Associates http://www.ca.com	Windows	Modèleur UML dédié au eBusiness en entreprise. Un des outils les plus complets du marché
Describe Embarcadero http://www.embarcadero.com/	Windows	S'intègre aux IDE Java. Supporte le développement en groupe.
Rhapsody I-Logix http://www.ilogix.com	Windows	La gamme Rhapsody permet d'utiliser UML dans le cadre de développement d'applications embarquées
Jsequence Objective Ideas AB http://www.obj.se	Java	Génération de diagrammes à partir d'un code Java
MagicDraw http://www.magicdraw.com/	Java, MacOS X, Linux, Unix, Windows	Outil UML avec module de travail de groupe. Il est compatible avec Corba. Il peut générer des DDL
Gamme Rational Rose / XDE http://www.rational.com	Windows .NET Java	Rational édite les outils UML les plus complets et une large gamme d'éditeurs. L'éditeur propose des versions spécifiques Java et .NET. Il possède aussi plusieurs déclinaisons de Rational Rose.
Objectteering UML Modeler http://www.objectteering.com	Windows, Unix	Environnement de conceptions UML se décline en plusieurs versions. La version personal est gratuite. Seule la version entreprise inclut le travail en groupe
PowerDesigner Sybase http://www.sybase.com	Windows	Outil de conception UML orienté entreprise. Support du ebXML
Together Solo http://www.borland.com	Java .NET	Modèleur UML pour les développeurs indépendants et les petites équipes. Existe différentes déclinaisons.
Visual Paradigm http://www.visual-paradigm.com/	Windows, Solaris, Java, Linux, Unix	Outil visuel UML. Il incorpore un IDE en plus du modèleur UML.
Visual UML Standard Edition Visual Object Modelers http://www.visualuml.com	Windows	Génération de diagrammes et reverse engineering pour C++, C# et Java
Visual UML Standard Edition for VB Visual Object Modelers http://www.visualuml.com	Windows	Module UML pour VB Existe une version Visual UML Plus Edition for VB incluant VBA et MS Repository 2.0
PowerAMC http://www.sybase.com	Windows	Environnement de développement intégrant UML
Poseidon for UML http://www.gentleware.com/	Java, C#, VB.NET...	Environnement UML bien connu. Prix accessible, existe en plusieurs versions.
ModelMaker http://www.modelmakertools.com/	Delphi	Outil UML dédié exclusivement à Delphi
TAU http://www.telelogic.com	Windows Linux	Environnement UML 2.0. Existe en 2 déclinaisons, possibilité de temps réel

Biztalk Server : l'EAI façon Microsoft

Intégration, ce mot revient dans tous les projets informatiques. Cette préoccupation n'est bien sûr, pas nouvelle. La nouveauté réside plutôt dans le fait que l'intégration donne lieu dorénavant à des projets à part entière et a fait naître une nouvelle offre logicielle : les solutions d'E.A.I. Nous allons voir quelles sont les problématiques d'une solution d'EAI et comment *BizTalk*, l'offre E.A.I. de Microsoft, se propose d'y répondre.

L'EAI peut se définir comme un ensemble de techniques permettant d'intégrer des applications indépendantes, dans le but de créer un système plus étendu. Cette intégration implique la mise en place d'une architecture permettant à ces applications de dialoguer entre-elles.

L'EAI constitue une approche systématique des problématiques d'intégration d'applications. Elle met l'accent sur une intégration métier et non plus uniquement technique.

Cette industrialisation de l'intégration a fait naître de véritables "Ateliers de Génie Logiciel" dédiés à l'intégration. Ces solutions fournissent aussi bien les outils nécessaires pour la modélisation des processus métier, pour la définition et la conversion des formats de documents, et pour l'orchestration des flux de données.

L'EAI permet de passer d'un S.I. spaghetti où les applications échangent directement des informations les unes avec les autres, à un S.I. intégré où les échanges sont gérés par un hub central. (figure 1 et 2)



Figure 2 - Le système d'information intégré

les avoir préalablement identifiés, formalisés et normalisés. Il est intéressant de noter que les projets d'EAI sont souvent l'occasion de formaliser les processus intra ou inter-entreprises.

BizTalk permet de modéliser graphiquement les Business Process. Il fournit à cet effet un module spécifique dédié à la modélisation. Dans ce module, appelé *BizTalk Orchestration Manager*, l'écran est divisé en deux parties : une partie métier et une partie technique. Le principe de ce module est de modéliser des événements métier (ex : réception de commande) et de leur faire correspondre des événements techniques (ex : réception d'un fichier via le Web).

La modélisation de ces processus avec *BizTalk* s'avère très intuitive. En effet, le module de modélisation permet de dessiner les processus métier. Ce module est basé sur un outil de dessin existant appelé *Microsoft Visio*.

Une fois que l'on a d'un côté, les actions métier et de l'autre, les événements techniques, il faut les relier entre eux. Pour ce faire, BizTalk a défini un certain nombre de notions et d'objets de communication (entrées/sorties). Ces objets sont les ports et les canaux. Ils indiquent à *BizTalk* le mode de transport (Fichier, email, ftp, ...) et le format des données qui entrent ou sortent de *BizTalk*.

Définition des formats de documents échangés

Les applications qui sont amenées à échanger des informations n'ont bien évidemment pas connaissance les unes des autres. En revanche, elles doivent définir les données qu'elles mettent à disposition. Cette définition de données doit être publique afin que toute application désirant dialoguer avec une autre ait accès à cette définition.

BizTalk Server prévoit à cet effet deux modules. Le premier permet de définir des formats de données et le second permet de les déposer dans un référentiel.

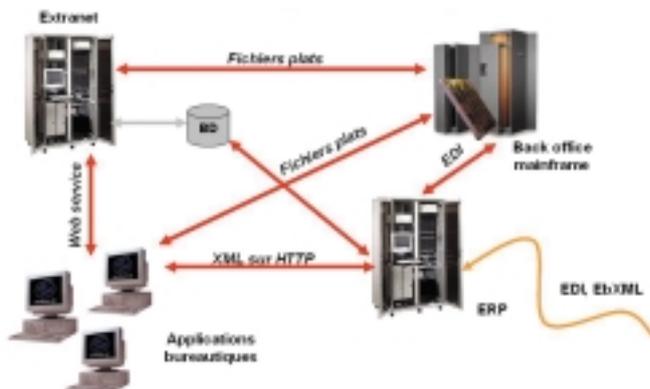


Figure 1 - Le système d'information spaghetti

BizTalk face aux challenges de l'EAI

Après avoir passé en revue les fonctionnalités d'un outil d'EAI, nous allons nous attacher à regarder comment les mettre en œuvre avec BizTalk Server 2000.

Ce dernier offre une approche centrée, non pas sur les données, mais bien sur les processus métier, en cela BizTalk est un outil de deuxième génération.

Modélisation graphique des Business Process

Les processus métier déterminent la logique d'échange entre les différentes applications. Cette approche centrée sur les processus nécessite :

- de posséder des processus clairement identifiés et définis,
- de disposer d'un outil pour les modéliser.

La modélisation des processus métier n'est possible qu'à condition de

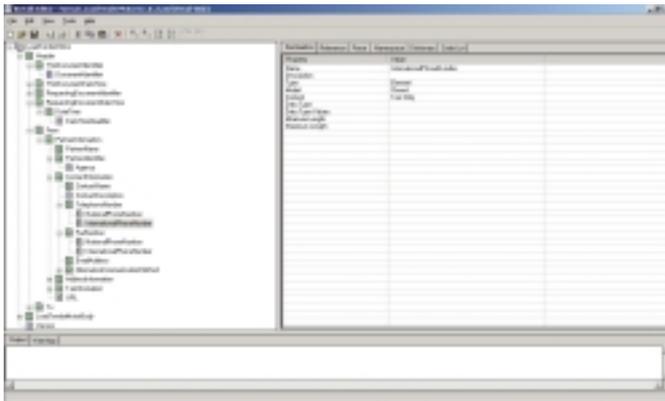


Figure 3 - L'éditeur de documents de Biztalk

BizTalk utilise très largement la norme XML pour la définition des données. Cette définition est réalisée dans un module BizTalk appelé BizTalk Editor. Ce module permet de définir la structure des messages que l'on utilise pour échanger des informations. En plus du format XML, cet éditeur supporte les formats issus de l'EDI : EDIFACT et X12. Pour désigner cette définition de document, BizTalk utilise le terme de "spécification".

Une fois que la structure est définie et sauvegardée au format XDR (l'export au format standard XSD n'est supporté qu'à partir de la version 2002 de BizTalk), on la dépose dans le référentiel de BizTalk. Ainsi, elle devient accessible pour tous les processus métier modélisés dans BizTalk. BizTalk utilise le terme de "Document Definition" pour désigner une spécification déposée dans le référentiel.

C'est l'ensemble de ces définitions de documents qui constitue le référentiel BizTalk. Nous constatons donc qu'en aucun cas, ce ne sont les applications qui ont la charge de fournir les formats de données. Ces définitions sont centralisées hors des applications. (figure 3)

Transformation des documents d'un format vers un autre

Chacune des applications possède un format de données qui lui est propre. Pour que ces applications puissent malgré tout dialoguer entre elles, BizTalk va donc devoir procéder à des transformations de format. Ces transformations peuvent être des transformations simples ou des transformations plus complexes qui font appel à des fonctions.

Pour plus de clarté, prenons l'exemple du Business Process de passage de commande en ligne. Une des étapes de ce processus sera l'émission d'une facture à partir d'un bon de commande. Cette opération va être réalisée par des applications différentes, toutes deux reliées et pilotées par BizTalk. Apparaît donc le besoin de transformer un bon de commande en facture. Les spécifications de la facture et du bon de commande sont déclarées dans le référentiel BizTalk. Il faut indiquer à notre Business Process comment reconstituer l'adresse d'une facture à partir des informations présentes dans le bon de commande. Pour réaliser cela, BizTalk a prévu un module spécifique, nommé BizTalk Mapper qui permet de dessiner, puis de générer les transformations nécessaires.

Là encore, les transformations se définissent de manière visuelle et sont ensuite générées au format XSLT. BizTalk permet également de développer ses propres fonctions de transformation. Ces fonctions de transformation sont appelées des functoids. (figure 4)

Implémentation des échanges de données à travers plusieurs protocoles

Toutes les applications sont reliées à BizTalk grâce à un connecteur. Quelle que soit la forme que prend ce connecteur, il utilise un mode de transport entre l'application et BizTalk. Ce mode de transport repose sur un protocole. BizTalk permet de communiquer avec des applications grâce à des protocoles comme HTTP/HTTPS, SMTP, MOM, ...

L'intérêt d'une solution d'EAI est de permettre le transport d'informations à travers plusieurs protocoles. En effet, les applications à relier sont souvent de nature hétérogènes et situées sur des sites différents. BizTalk prend en charge la gestion des protocoles et permet donc de se concentrer sur les données et non sur les couches de plus bas niveau.

Nous verrons plus loin, que les applications ne sont en réalité pas directement connectées à BizTalk, mais utilisent pour cela des canaux et des ports. Il est intéressant de noter que BizTalk, comme les autres solutions d'EAI, favorise l'utilisation de middlewares asynchrones de type broker de messages (ex : MQSeries, MSMQ, ...). En effet, la nature des besoins d'intégration impose souvent une utilisation massive du mode asynchrone. Il est néanmoins possible de réaliser des appels RPC, via DCOM ou des connecteurs développés spécifiquement pour une application (AIC). Notons également que BizTalk ne fournit pas de moyen de transport des données mais s'appuie sur des middlewares existants. Il permet de définir l'itinéraire des documents à travers les différents systèmes.

Orchestration des flux de données à travers les différentes applications

Les principales différences entre un projet d'intégration simple et un chantier d'EAI sont le couplage faible entre les applications et la possibilité d'appliquer **des règles de routage** sur les flux d'informations échangées par les applications.

La première génération d'outils d'EAI était fondée sur l'utilisation de middlewares asynchrones, principalement orientés Messages (MOM). Cette génération d'outils permettait la mise en place d'un bus d'échange de données, avec un faible couplage entre les applications. En revanche, la gestion des flux (route, règle de routage) était à la charge d'une ou plusieurs applications tiers, développées spécifiquement pour chaque nouvelle règle. Cette implémentation des règles dans du code rendait le *Business Process* rigide et nécessitait un développement qui n'aurait dû être qu'un paramétrage.

Aujourd'hui, les solutions d'EAI telle que BizTalk, permettent de modé-

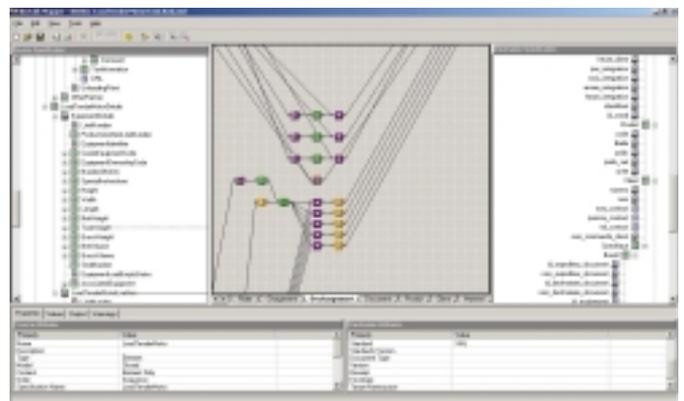


Figure 4 - Le Mapper de Biztalk

liser le flux de l'information à travers les différents sites ou applications. Les outils de deuxième génération ne se bornent plus à fournir un bus de données passif, mais ils intègrent dorénavant une intelligence qui leur permet d'orchestrer les flux.

Nous avons vu que BizTalk permet de modéliser graphiquement les Business Process dans un module nommé Orchestration Designer. C'est ce *Business Process* qui, en définissant des règles métier, va aiguiller les documents et répartir les messages vers les différentes applications, via les connecteurs.

Dans un premier temps, l'Orchestration Designer permet la modélisation de la route d'un document, puis dans un second temps d'indiquer quelles informations vont passer sur cette route.

Administration et supervision du déroulement de ces échanges

L'EAI est l'industrialisation de l'intégration logicielle. Cette industrialisation sous entend que les volumes échangés sont importants et croissants. Une solution d'EAI doit donc permettre de tenir la charge et fournir des outils d'administration et de supervision.

Cette administration peut être de deux types : une administration système ou une administration des échanges des données.

L'administration système concerne les paramètres techniques de la solution d'EAI : dimensionnement des bases, pooling de ressources, gestion de droits, ... Ce type d'administration n'est pas spécifique aux solutions d'EAI. En revanche, l'aspect orienté métier adopté aujourd'hui par les solutions d'EAI nécessite de pouvoir superviser les échanges de messages entre les applications et les sites. Ce suivi est facilité par le fait que tous les messages, bien qu'utilisant des canaux de communication différents, passent par BizTalk. Ainsi, ce dernier est capable de garder une trace de chacun d'eux.

A cet effet, BizTalk propose un module de suivi de documents appelé BizTalk Document Tracking. Ce module permet la consultation de tous les échanges qui ont eu lieu à travers BizTalk. Il se présente sous la forme d'une interface Web qui affiche des informations telles que : l'application de départ, l'organisation d'arrivée, les heures ... Il est également possible de journaliser des informations métier contenues dans les messages XML échangés.

Pourquoi mettre en place BizTalk dans votre système d'information ?

BizTalk couvre les principaux besoins en matière d'EAI et de B2B. Si les fonctionnalités techniques de BizTalk sont facilement identifiables, son domaine d'application précis est quant à lui, large et donc plus difficile à définir. En effet, les produits d'EAI, tels que BizTalk, peuvent être utilisés pour atteindre plusieurs objectifs.

Faire dialoguer des applications intra-entreprise (EAI)

BizTalk est utilisé dans ce cas comme un middleware de haut niveau qui permet de relier entre elles plusieurs applications internes, et cela, avec un couplage faible (sans que les applications n'aient connaissance les unes des autres). De plus, BizTalk permet d'utiliser plusieurs canaux de communication et de définir des règles de routage de l'information. Son principal intérêt réside dans la possibilité d'appliquer ces règles de routage, quel que soit le mode de transport de l'information.

Mettre en place des échanges de données inter-entreprises (B2B)

Les pratiques commerciales évoluent et les partenariats se multiplient.

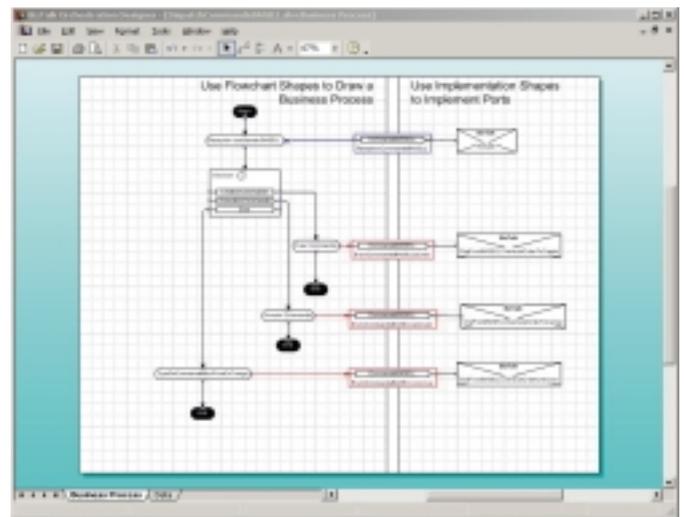


Figure 5 - Orchestration de processus avec Biztalk

Cette augmentation des échanges entraîne le besoin d'ouvrir son système d'information vers l'extérieur, tout en évitant de trop le coupler avec celui des partenaires. On ne parle plus alors d'intégration d'applications mais d'intégration étendue. BizTalk répond également à cette problématique.

Pour dissocier les processus métier et leur implémentation (Business Process Integration)

Les entreprises n'ont pas attendu l'émergence des outils d'EAI pour échanger de l'information entre applications. En revanche, ces échanges sont bien souvent réalisés par des programmes " batch ". Ces programmes emprisonnent la logique métier et nécessitent du développement pour tout changement dans les règles métier. La modélisation des processus ne remplace pas leur implémentation. En revanche, le fait de modéliser les processus permet de ne pas se déconnecter de la logique métier en l'enfouissant dans un programme informatique.

Conclusion

Les réalités qu'englobent la notion d'EAI sont nombreuses : intégration d'applications, interfaçage des systèmes d'informations entre partenaires, orchestration de processus métier. Les outils d'EAI sont entrés dans une deuxième génération, qui leur permet de couvrir tous ces aspects. BizTalk fait partie de cette génération d'outils, qui se présentent comme des solutions globales aux problèmes d'intégration.

Les concepts de ports et de canaux appliqués à l'EAI s'avèrent très pratiques et permettent d'isoler l'information de son mode de transport.

L'un des intérêts de l'offre EAI de Microsoft réside dans sa simplicité d'utilisation. La modélisation graphique des Business Process rend leur définition possible par les équipes fonctionnelles et non techniques. Bien que BizTalk ne dispense pas d'implémenter les processus métier, il rend le contrôle de ces processus à la maîtrise d'ouvrage.

La modélisation des processus métier est en cours de normalisation avec des initiatives telles que BPML (*Business Process Modeling Language*). Cette normalisation pourrait aboutir à rendre la description des processus métier indépendante des moteurs d'intégration qui vont les orchestrer.

■ Eric K'Dual - Expert EAI chez Neoxia,

cabinet de conseil en architecture des systèmes d'information.



Protégez votre code des regards indiscrets avec des obfuscateurs

Ces nouveaux logiciels, très en vogue avec Java ou .Net, sont censés empêcher la rétro génération de votre code. Une motivation pas toujours très saine comme nous allons le voir.

En prenant l'angle du code source, quel est le problème commun entre un programmeur Java, PHP, ASP.NET, ou même VB.NET ? Les développeurs qui revendent des solutions basées sur ces langages auront du mal à empêcher leurs clients de visualiser le code de leurs applications. Par exemple avec l'"ASP ancienne génération", un simple éditeur de texte comme le "bloc-notes" suffit à dévoiler les arcanes du logiciel. Avec ASP.NET le code est compilé, c'est-à-dire transformé au final en "assemblages". Mais Microsoft a développé un outil du nom de ILDASM qui permet de les décompiler. Cet outil produit des instructions MSIL (*Microsoft Intermediate Language*) et par conséquent, quelqu'un qui comprend ce langage peut facilement déchiffrer votre code original. Maintenant, il y a mieux, ou pire selon le point de vue, car il existe des outils capables de réaliser automatiquement la transformation des assemblages en code source de haut niveau (C#, VB.NET ou code managé C++.NET).

Si vous n'êtes pas convaincu, nous vous invitons à surfer à l'adresse : <http://www.remotesoft.com/salamander>. Cet outil "Salamander .NET Decompiler" a été développé par un Monsieur Huihong, Docteur en informatique de son état, et éminence grise de l'Université de Stanford.

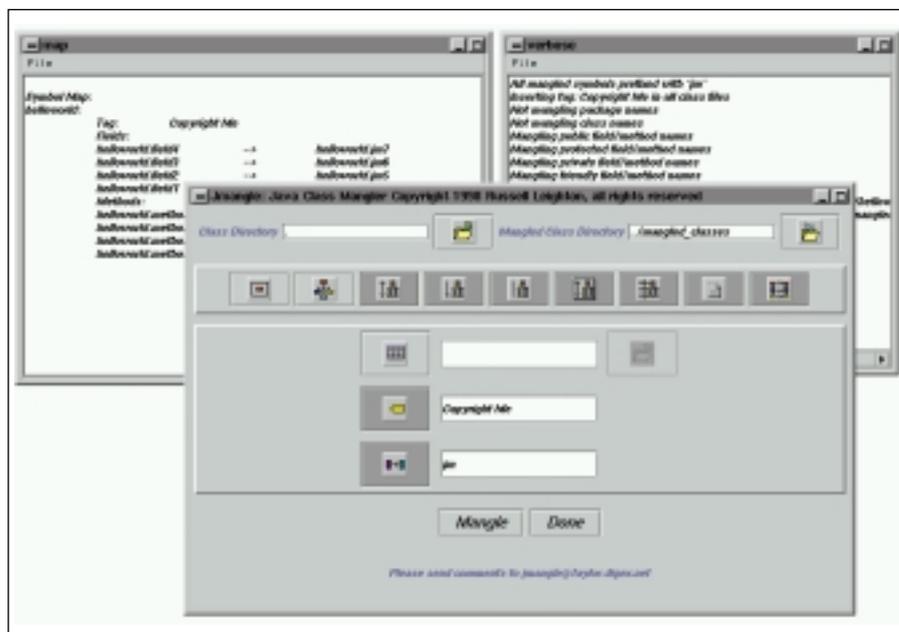
A l'invite "I'm an Advanced User (size limit: 1.5Mo, decompiling 10 methods in each class) File Name:", chargez un assemblage (prenons le cas d'un "exécutable .NET d'extension ".exe"). Le résultat est étonnant : vous retrouverez votre code tel quel, et dans certains cas, le code sera même plus lisible car convenablement reformaté ! Pour votre information, le coût de cette petite merveille tourne quand même autour des 1 000 €.

Si vous êtes au désespoir de garder votre code confidentiel, sachez que notre bon docteur a eu la brillante idée de développer un outil

complémentaire, baptisé de manière générique "obfuscateur". Celui-ci a pour rôle de "brouiller les cartes" au maximum, afin de rendre le résultat d'une décompilation le moins clair possible. L'obfuscateur transforme les noms de variables et de fonctions. Par exemple le nom de fonction "MaFonctionDeLecture" deviendra "\$ouijknn". Vous pouvez aussi tester cet outil on-line (<http://www.remotesoft.com/salamander/obfuscator.html>).

tion du code ASP sous Unix est possible, via la solution proposée par l'éditeur ChiliSoft (<http://www.viaverio.com/support/virtual/web/asp/chili/>).

Pour le code managé, ASP.NET et autre, vous pouvez transformer le code MSIL en instructions binaires machine, ce que l'on appelle du code pre-JIT (Just In Time). Le Framework .NET propose l'outil "Native Image Generator" dans ce but. Le gros inconvénient est que vous perdrez tout bénéfice du code MSIL. Autrement dit, en théorie, sur une plate-forme quelconque le code MSIL est exécutable, vu qu'il est interprétable à un plus bas niveau (c'est le cas du code MSIL généré par le projet MONO sous Linux par exemple). Mais le code pre-JIT



Evidemment ici, l'avantage principal de cet obfuscateur est qu'il est impossible de le décompiler à l'aide du "Salamander .NET Decompiler"... Toujours pour votre information, si vous désirez utiliser cet "empêcheur d'y voir clair", vous devrez déboursier environ 500 €. Sinon quelles autres possibilités avez-vous ? Vous pouvez encapsuler votre code "ASP classique" au sein d'un objet COM, mais en contrepartie vous perdrez en portabilité. En effet, il ne faut pas perdre de vue que l'exécu-

implique une compilation obligatoire pour un processeur ou/et une plateforme cible...

Obfuscateur C/C++ et C#

COBF signifie "C-Obfuscator". Vous pouvez le télécharger gratuitement à l'adresse <http://home.arcor.de/bernhard.baier/cobf/>. Cet outil tourne sous Windows (9x à XP) et aussi sous Linux ou Unix. Il est livré avec son code source, ce qui est intéressant pour déterminer son fonctionnement.

Avec COBF le code :

```

/* test.c - simple test program for COBF */
#include <stdio.h>
#ifdef unix
#define MAX_COUNT 10
#else
#define MAX_COUNT 20
#endif
int main()
{
int i;
for (i = 0; i < MAX_COUNT; ++i)
printf("Hello %d!\n", i);
return 0;
}
    
```

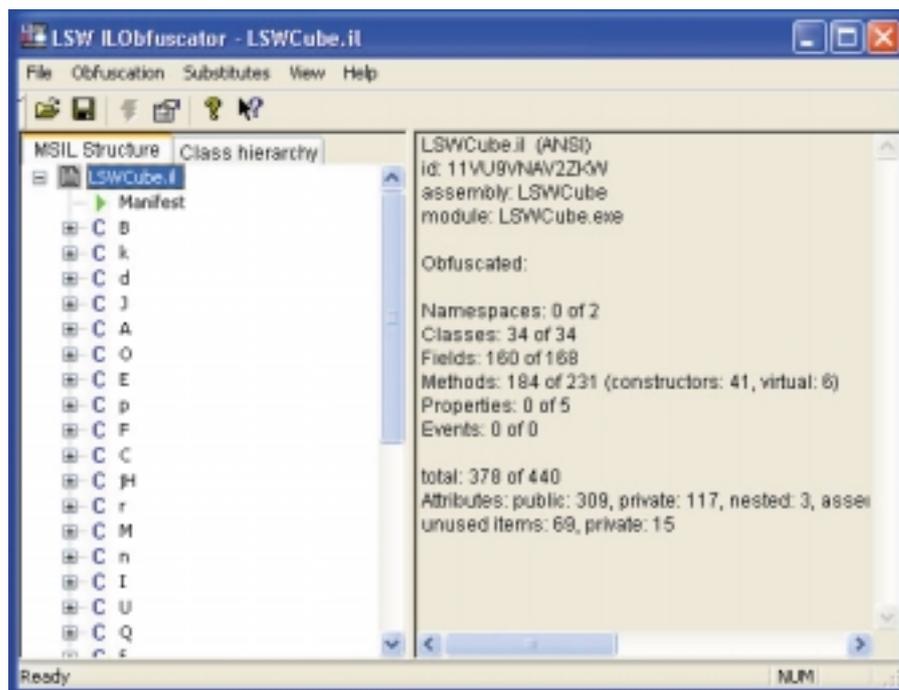
Devient :

```

/* COBF by BB - 'test.c' obfuscated at Sun Jan
21 18:44:05 1996
*/
#include<stdio.h>
#include"cobf.h"
#ifdef unix
#define b 10
#else
#define b 20
#endif
c e(){c a;d(a=0;a<b;++a)f("\x48\x65\x6c\x
6c\x66\x20\x25\x64\x21\n",a)
:g O;}
    
```

On peut difficilement imaginer un code moins lisible !

Avec C# de dot net, les obfuscateurs permettent de coder directement le byte-code, afin de



rendre illisible le fichier source lors de la décompilation. Un produit reconnu pour C# est l'"IL-Obfuscator" (http://www.lesser-software.com/en/content/products/LSW%20DotNet-Tools/LSW_DotNet_IL-Obfuscator.htm). Vous compilez le code normalement, puis vous exécutez l'outil qui renommera variables, méthodes et autres types. Pour rappel, le comportement de l'application n'est en rien modifié.

proposé par Russell Leighton est freeware. L'interface est excellente et le logiciel est fort bien documenté. Il y a aussi Retroguard, mais qui ne fonctionne pas avec les servlets (<http://www.retrologic.com/retroguard-main.html>). Enfin, un outil du nom de JAX, développé par IBM vaut le détour : <http://www.alphaworks.ibm.com/tech/JAX/>.

Dans Programmez n°57, Frédéric Mazué a signé l'article "Un chargeur de classes personnalisé pour dissimuler votre code java". Sans le savoir (...), il démontre qu'il est très facile de coder un obfuscateur en java... A découvrir, ou à relire.

Obfuscateurs PHP

Un des meilleurs est sans doute POBS (<http://pobs.mywalhalla.net/>). Il est distribué sous licence GPL. D'autres obfuscateurs pour PHP existent comme SourceGuardian (200 € : <http://www.phpguardian.com/>), ou le commercial Zend (<http://www.zend.com/>). Notez aussi un projet open source du nom de Microcode, qui a été développé à l'origine pour proposer une alternative à Zend (<http://sourceforge.net/projects/microcode>).

Obfuscateurs Java

Il en existe de nombreux. Les classes java pouvant être facilement décompilées (exemple : le décompilateur GPL jode <http://jode.sourceforge.net/>), le problème s'est posé très tôt. Sans oublier qu'un obfuscateur optimisera votre code en réduisant la taille des classes.

Si vous devez commencer par en choisir un, testez d'abord Jmangle qui réussit à empêcher le reverse engineering (<http://www.elegant-software.com/software/jmangle/>). Ce logiciel

Obfuscateurs DOT NET

Nous avons déjà parlé de Salamander mais il y en a d'autres comme XENODE (<http://www.xenocode.com/fr/>), dont une version de démonstration est téléchargeable on-line ; dotfuscator (<http://www.preemptive.com/dotfuscator/index.html>) ; ou encore dynuobfuscator (<http://www.dynu.com/dynuobfuscator.asp>). Du côté des logiciels gratuits existe Anakrino (<http://www.saurik.com/net/exemplar/>), mais son développement semble ne pas avancer d'un iota pour l'instant.

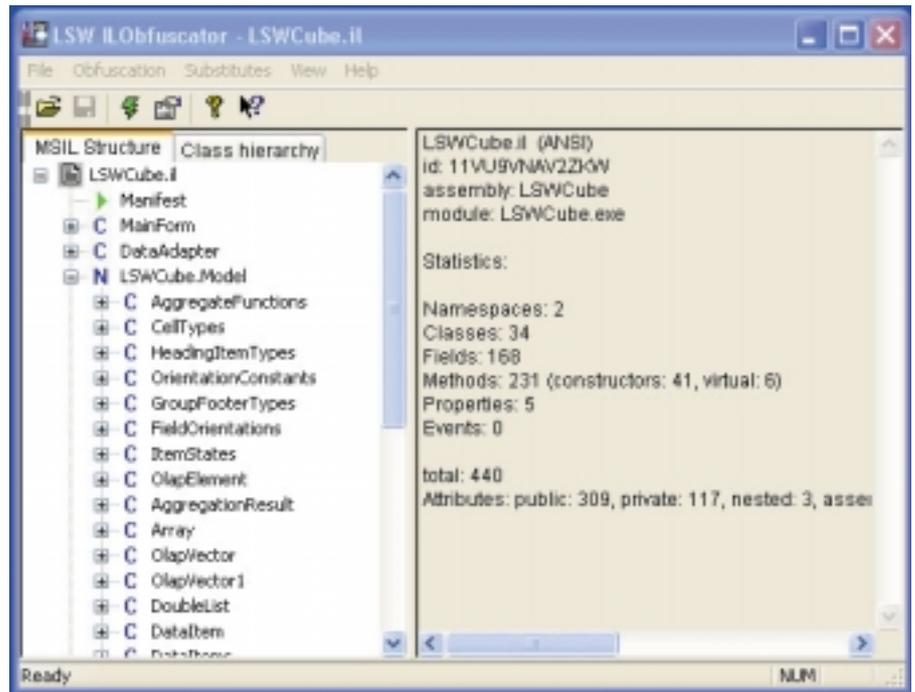
Pour quelles raisons ne PAS utiliser d'obfuscateur

Nous pensons qu'il s'agit d'une perte de temps et d'argent. D'abord la majorité des logiciels

commerciaux ne comporte pas de "code révolutionnaire". La valeur d'un logiciel s'axe sur le fait que l'ensemble de son code constitue un tout utilisable. La décompilation représente un avantage, et non pas un inconvénient, pour la maintenance du code. Si vous perdez les sources de votre propre logiciel, vous pourrez revenir en arrière et les reconstituer.

La plupart du temps un logiciel est de toute manière incompréhensible sans documentation, alors pourquoi ajouter une nouvelle couche d'incompréhension ? Sans conteste, l'utilisation d'un obfuscateur est contre-productif pour le développeur.

Vous avez en réalité bien plus de raisons de livrer votre code source avec votre produit commercial. Vous en donnez plus à vos clients sans que cela vous coûte. N'oublions pas qu'avec le code managé il n'y a plus d'appels API obscurs et propriétaires. Un concurrent pourra de toute manière parvenir au même résultat, car il possède le même matériel de base (le framework dot net). Il copiera votre code source pour gagner du temps, et non pas dans le but de "pirater votre propriété intellectuelle". En ce qui concerne la protection intellectuelle, les grands éditeurs utilisent des brevets logiciels et des avocats pour les défendre. Rien à voir avec des obfuscateurs.



Si un de vos clients découvre "un bug" il pourra peut-être résoudre le problème à votre place, à condition qu'il dispose des sources. Le modèle Open Source le prouve : plus il y a de gens qui regardent votre code, mieux ce sera pour la solidité de celui-ci. Si votre code est

"bien fait" vous gagnerez en respect de la part de vos clients. Votre expertise ne sera plus à prouver, elle découlera de la haute qualité de votre code. Bref, le fait de fournir votre code deviendra vite facteur de réduction de coûts, et fidélisera vos clients.

■ Xavier Leclercq



L'univers du développement :

- L'actualité
- Le téléchargement
- **Le forum technique : questions réponses**
- L'emploi • Les offres • Votre CV
- Les archives du magazine : les articles publiés en 2002-2003

www.programmez.com
www.programmez.com
www.programmez.com
www.programmez.com
www.programmez.com
www.programmez.com

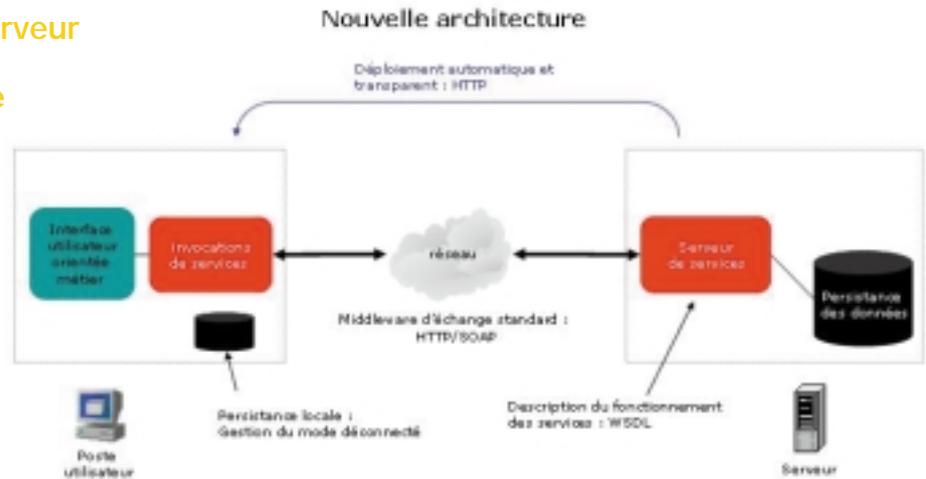
Le client-serveur n'est pas mort !

L'idée novatrice du client serveur était de répartir les tâches entre un serveur et un poste utilisateur devenu intelligent. Le rôle du serveur était, dans la plupart des cas, de centraliser les données et de gérer une partie des traitements, tandis que le client gérait l'autre partie des traitements et l'interface utilisateur.

La communication entre ces deux tiers s'effectuait au travers d'une couche logicielle spécifique, souvent appelée "middleware". L'architecture client/serveur a été massivement utilisée dans la plupart des systèmes d'information, mais elle a fini par montrer ses limites. En effet, la non standardisation du middleware rendait difficile la gestion des flux. De plus, la non standardisation du frontal client a confronté les directeurs informatique à la délicate problématique du déploiement sur les postes utilisateurs. Tirant les enseignements des limites du client/serveur, on s'est alors intéressé à un autre type d'architecture, fondé sur un client léger et standard : le navigateur Web. Le navigateur a été cantonné aux tâches de présentation. HTTP, un middleware simple mais efficace a été choisi comme standard pour les échanges. L'ensemble des traitements a été reporté sur le serveur, reprenant en partie l'ancien modèle des grands systèmes. Sont ainsi apparues les architectures Web.

Les limites du client léger

Si le client léger a résolu la problématique du déploiement, il reste assez limité en termes fonctionnels. En effet, s'il sait effectuer quelques contrôles de format sur les données grâce au langage JavaScript, il ne peut prendre en charge des traitements sophistiqués. De plus, l'interface HTML a une ergonomie limitée ; elle est orientée navigation et non manipulation. Elle n'est pas adaptée à la saisie de masse, et ne sait pas gérer correctement le multi fenêtrage. D'autre part, de par la nature



déconnectée du protocole HTTP, il est impossible de gérer des opérations contextuelles sur une page HTML (comme par exemple l'auto complétion du code postal à partir d'un nom de ville).

Du reste, le client léger ne sait pas prendre en charge le mode déconnecté. Ainsi un utilisateur ne peut travailler sans contact permanent avec le serveur. Cette limite est particulièrement sensible dans le cadre de l'accès par des terminaux mobiles utilisant des connexions instables (GSM/GPRS).

Enfin, l'interface client léger est inadaptée aux applications nécessitant de hautes performances et des temps de réponses très rapides (comme par exemple les applications de salle de marché).

Vers une nouvelle architecture ?

Pour les raisons précédentes, le monde informatique recherche un nouveau modèle d'architecture combinant les avantages du client / serveur et du client léger.

Ce modèle s'appuie sur une interface cliente vélocité, permettant le multi fenêtrage et la manipulation. Cette interface est orientée métier. Si elle s'appuie sur des composants standardisés, elle nécessite néanmoins un développement spécifique au travers d'un environnement de développement. Elle est capable de gérer le mode déconnecté grâce à une persistance locale, en général en XML. Son déploiement est transparent, et elle est capable de se mettre à jour automatiquement au travers du protocole HTTP. Pour permettre une plus grande liberté dans les échanges

avec le serveur, le nouveau modèle d'architecture s'appuie sur des services et sur un langage standard et sophistiqué (SOAP). Ces services savent décrire leur mode d'utilisation grâce à un format de description appelé WSDL. Ce modèle d'architecture est aujourd'hui rendu possible grâce aux avancées des standards que sont SOAP et WSDL. Mais les environnements J2EE et .NET permettent-ils le déploiement de telles architectures ? Avec la standardisation du protocole de communication, est-il envisageable d'utiliser d'autres technologies plus interactives sur le poste client, comme la technologie Flash ?

Peut-on envisager ce type d'architecture en environnement J2EE ?

La plate-forme J2EE fait aujourd'hui office de référence dans le monde des technologies Web ; Sun Microsystems et les éditeurs qui l'accompagnent ont su séduire les entreprises en éditant des standards technologiques autour du langage Java. Mais l'environnement J2EE donne-t-il des outils efficaces pour implémenter une architecture Web de type client riche ?

Comme tout langage de programmation, le langage Java permet l'implémentation d'interfaces riches, à base de fenêtres, de panels, et d'une multitude de contrôles graphiques. Ces interfaces sont la plupart du temps autonomes, mais elles peuvent être embarquées dans des applications Web sous la forme d'applets. Néanmoins, Java souffre, dans ce domaine, d'un historique mouvementé. Dès le JDK 1.0, il était possible de développer des inter-

faces utilisateurs à l'aide du package AWT. Même si cette API était portable (une interface AWT pouvait être déployée sur plusieurs systèmes d'exploitation), le résultat l'était beaucoup moins. Comme l'API déléguait la présentation des objets graphiques au système d'exploitation, l'apparence des interfaces créées pouvait diverger.

Cette petite entorse à la portabilité a poussé Sun à proposer une nouvelle API appelée Swing. Pour garantir le rendu graphique, cette API gère elle-même le rendu des contrôles utilisateurs (un bouton n'est plus dessiné par le système d'exploitation, mais par l'API elle-même, en utilisant des fonctions de dessin 2D). La conséquence de ce choix d'implémentation est une perte de performance sensible entre AWT et Swing, pour réussir à produire des interfaces identiques sur de multiples systèmes d'exploitation.

Cet historique chaotique a empêché le langage Java de combler son retard face aux applications natives Windows. Les environnements de développement Java mettent la plupart du temps Swing en avant, à cause de la complexité et du "look and feel" discutable de AWT, aux dépens des performances et donc de la qualité de l'expérience utilisateur. Des projets comme SWT (Standard Widget Toolkit, un sous-projet de Eclipse) montrent pourtant qu'il est possible de créer des interfaces Java efficaces, autant sur l'apparence que sur la consommation de ressources.

Le deuxième aspect structurant d'une architecture Web avec client riche est le déploiement et la mise à jour de l'interface graphique. L'environnement J2EE propose, pour simplifier cette tâche, une technologie appelée Java Web Start. Cette technologie est composée d'un ensemble d'outils, dont une application Java qui doit être installée sur chaque poste cible du déploiement. Une fois Java Web Start installé, son rôle sera de gérer la distribution des applications Java publiées sur un serveur Web, et de fournir à ces applications des services de cache, de mise à jour automatique, et de sécurité. Le déploiement d'une application Java Web Start est configuré dans un fichier XML, d'extension ".jnlp", et se déclenche par simple clic sur un lien dans une page Web : Java Web Start est une technologie particulièrement efficace pour le déploiement des clients riches Java. La gestion des mises à jour est transparente pour l'utilisateur ; il peut exécuter ses applications de manière classique ou

en cliquant sur le lien Java Web Start correspondant, et l'environnement se charge de détecter l'existence de nouvelles versions de composants et de les déployer si besoin. Les applications Java Web Start sont naturellement sécurisées puisqu'elles s'exécutent par défaut avec un niveau de permissions très bas (comparable à celui des applets). Pour déployer une application avec plus de permissions, comme l'accès au système de fichiers de l'utilisateur, le développeur devra signer ses fichiers JAR, et proposer un certificat électronique de vérification aux utilisateurs.

Le client riche Java, une fois déployé sur le poste de l'utilisateur, doit pouvoir fonctionner en mode autonome comme en mode connecté. Comme nous l'avons vu précédemment, la communication entre un client riche et un environnement serveur doit reposer sur les Web Services, qui proposent une infrastructure de communication à base de standards technologiques (HTTP et SOAP).

Les Web Services, dont la spécification a été produite conjointement par Microsoft et IBM, ont été intégrés tardivement à la plate-forme J2EE, par l'intermédiaire de l'API JAX-RPC (Java API for XML-based RPC). La version 1.4 de la spécification de Sun imposera aux serveurs d'applications l'implémentation de JAX-RPC, complétée par une spécification appelée Web Services for J2EE, encore en travaux, qui standardisera le déploiement des Web Services.

Dans la pratique, la création, comme la consommation de Web Services en Java restent des tâches complexes, qui dépendent encore fortement de l'implémentation JAX-RPC choisie, compte tenu de l'avancement tardif des principales spécifications. Dans ce domaine, la plate-forme J2EE est encore loin du niveau de productivité de son principal concurrent, à savoir .NET.

L'environnement de Microsoft est-il un meilleur candidat pour les architectures Web avec client riche ?

Comme nous l'avons vu, l'environnement J2EE est – pour l'instant – un piètre candidat pour les architectures Web avec client riche. La lourdeur des interfaces graphiques et la complexité du développement et du déploiement des Web Services sont autant de freins à la mise en œuvre d'une telle architecture. Ce sont d'ailleurs ces mêmes facteurs qui expliquent un certain échec de Java Web Start, technologie pourtant aboutie et efficace.

Dans le domaine des interfaces graphiques riches, les environnements Microsoft ont toujours été un modèle de productivité et d'efficacité. Les outils de développement de la gamme Visual Studio font office de référence dans ce domaine, et le dernier venu – Visual Studio .NET – est un aboutissement remarquable puisqu'il permet d'unifier au sein du même outil l'ensemble des développements en technologies Microsoft. Quoi qu'en disent les détracteurs de Bill Gates, les outils Visual Studio sont les plus productifs quand il s'agit de produire des interfaces graphiques évoluées, essentiellement grâce à leur intégration fine au système d'exploitation Windows.

L'environnement .NET n'est pas en reste, quand il s'agit de déployer des applications via le Web. L'équivalent de Java Web Start en environnement .NET est appelé NTD. Il repose sur les mêmes principes, à savoir : déploiement, exécution, et mise à jour par une URL, gestion d'un cache d'applications local et application d'une politique de sécurité restrictive par défaut. La technologie NTD est intégrée directement au framework .Net ; aucun outil supplémentaire n'est requis, contrairement à Java Web Start qui est un package indépendant du JRE. Le modèle de sécurité est aussi contraignant, puisqu'il oblige le développeur à signer ses assemblies et à présenter un certificat pour obtenir un jeu de permissions élevées.

Dernier élément fondamental, les Web Services et les standards technologiques qui les accompagnent sont nativement intégrés au framework .NET. Le développement et le déploiement de Web Services .NET est facilité par un ensemble d'outils particulièrement productifs : génération du fichier de description WSDL, génération automatique de "proxys" à partir de ce fichier, API de haut niveau pour la création et la consommation de ces services, etc. Ces outils sont le reflet des travaux de Microsoft, qui positionne les Web Services au centre de sa stratégie technique et commerciale.

Flash : une solution limitée à l'interface utilisateur

La technologie Flash a été créée en 1996 par la société Macromedia, pour permettre d'intégrer des animations vectorielles de faible poids au sein de pages HTML. Les premiers usages de Flash ont été des animations audiovisuelles (marketing, démonstration, première visite, etc.)

Avec l'évolution de la technologie Flash et

grâce à son formidable taux de pénétration (97% des postes connectés à Internet), Macromedia a réorienté très tôt sa stratégie : l'objectif de Flash est devenu d'offrir une alternative à HTML et de fournir une technologie d'interface riche. Ainsi, bien avant J2EE et .NET, Flash a permis de créer des objets d'interface faciles à distribuer et aptes à échanger avec un serveur. Profitant de son avance, Macromedia a tenté de développer des mécanismes d'échanges propriétaires en proposant le serveur FLASH GENERATOR, pour finalement se réorienter vers des mécanismes et protocoles standards.

Flash permet de créer des interfaces très abouties avec du multi fenêtrage, des contrôles sophistiqués, à tel point qu'il est possible de reproduire une interface windows-like (www.ego7.com). Il se différencie de J2EE et .NET par son orientation Rich Media : sa capacité à gérer des objets audio, vidéo, etc.

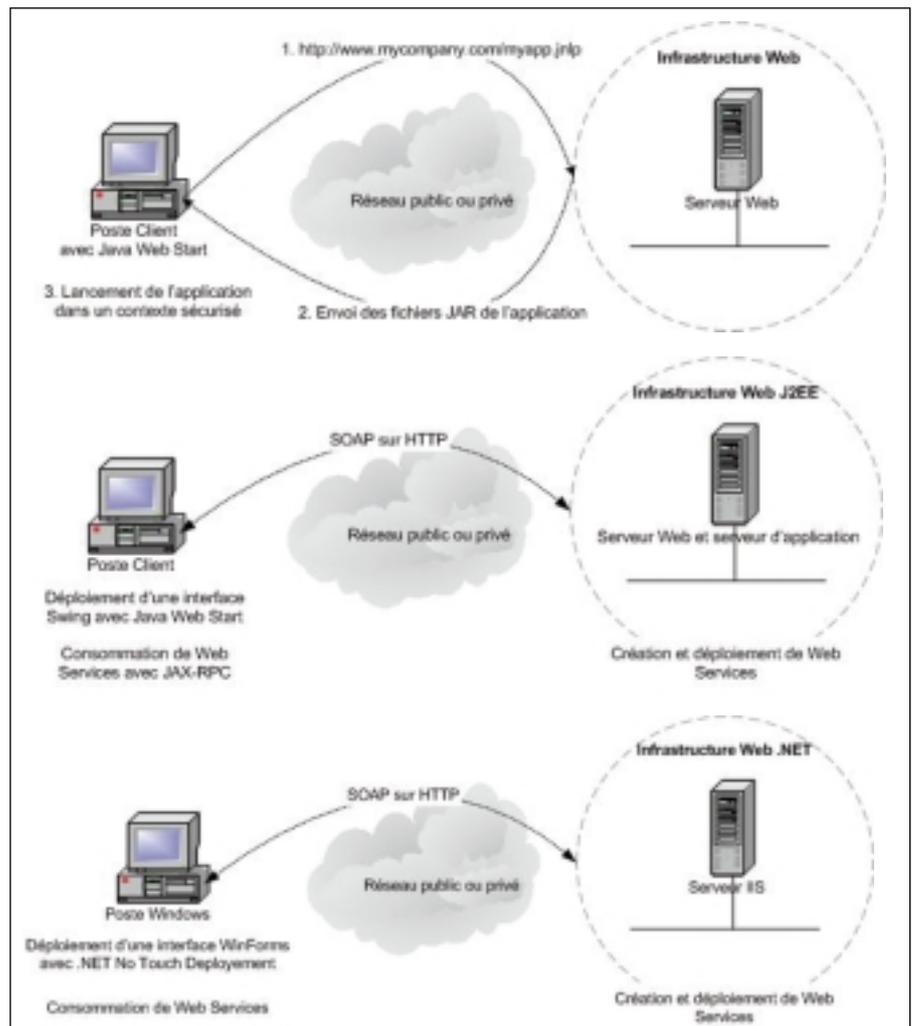
L'environnement de production Flash, destiné à ses débuts à des graphistes, a été repensé pour un usage par des développeurs, à partir de 2002. Flash MX Professional 2004 permet maintenant de créer des formulaires à la manière de Visual Basic. Il est basé sur Action Script, un langage réellement objet depuis la version 2.0. Il propose des API d'accès à des bases de données, des fichiers XML, et surtout à des Web Services.

La persistance sur le poste client, utile pour le mode déconnecté, est basée sur un format de fichier propriétaire à l'extension .SOL

Les objets flash sont compilés par l'environnement de développement et distribués au format SWF, pour être exécutés sur le Runtime Flash. Ce Runtime est disponible pour un grand nombre de plates-formes : Windows, Mac OS, Linux, Palm OS, Windows Mobile, téléphones Symbian, etc.

Il est distribué sous la forme de plug-in avec la plupart des navigateurs Web. Ainsi, on peut exécuter des objets Flash insérés en page HTML sur la quasi-totalité des postes utilisateurs. Par contre la version du Runtime fonctionnant sans navigateur est distribuée uniquement avec l'environnement de développement et sa pénétration est anecdotique. Ainsi, les applications Flash autonomes n'offrent aucun mécanisme de déploiement.

Enfin, Macromedia a plusieurs fois essayé de se positionner sans succès sur les technologies serveur. Si Flash est une technologie intéressante pour la création d'interface utilisateur,



elle n'adresse aucun autre besoin, à la différence de J2EE ou .NET.

Le navigateur n'est pas mort

Comme nous l'avons vu en introduction, le modèle d'architecture présenté dans cet article est pertinent pour couvrir certains besoins, et en particulier pour améliorer la productivité de l'utilisateur. Pour autant, les architectures Web classiques à base d'un client léger, sous forme de navigateur, ne sont pas à remettre en question pour l'instant. Il reste encore beaucoup de chemin à parcourir pour que les architectures Web avec client riche puissent être industrialisées. Au-delà des travaux encore en cours dans le domaine des Web Services, c'est toute l'architecture applicative – c'est-à-dire la répartition des traitements et leurs interactions – qui est encore à penser et à éprouver (rappelons que le design des architectures Web avec client léger est une activité encore en évolution, ce qui laisse présumer des travaux à réa-

liser pour un modèle d'architecture sensiblement différent).

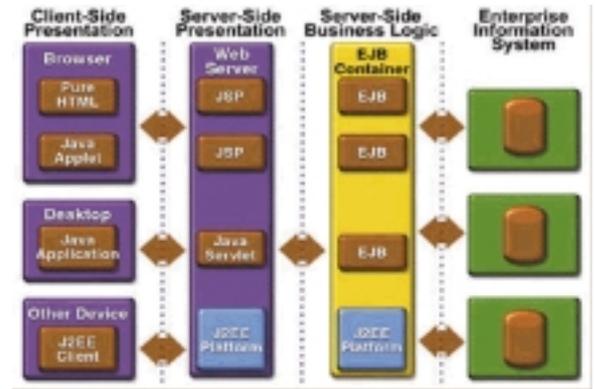
Contrairement aux architectures Web classiques qui mettent en avant l'environnement serveur, et donc indirectement favorisent une implémentation de type J2EE (seule plateforme à proposer une réelle intégration au système d'information), les architectures Web avec client riche remettent au goût du jour les développements côté client, et donc les technologies Microsoft. Ce qui nous pousse à nous interroger : tous les travaux de l'éditeur de Redmond en faveur de Web Services ne seraient-ils pas un moyen de favoriser l'émergence du nouveau modèle d'architecture dont nous venons de parler ? Et par la même occasion, de gagner du terrain face aux défenseurs de J2EE, en les amenant sur un terrain où ils ne peuvent se battre, à savoir le poste client, monopolisé par Windows ?

■ **Guillaume Plouin & Julien Soyer**

Pôle Conseil Technologique, Groupe SQLI

Développement J2EE : de l'intérêt d'utiliser un framework technique

J2EE s'impose comme une plate-forme de référence pour le développement d'applications d'entreprises. En effet, les sociétés envisagent massivement d'utiliser cette solution pour leurs projets futurs. J2EE permet de couvrir tous les besoins des applications, des problématiques de stockage de données à celles de présentation de l'information. De la même façon, il laisse beaucoup de liberté sur le choix de l'infrastructure matérielle.



Tous ces avantages de J2EE ont un coût :

- l'environnement est riche : le maîtriser implique des développements plus coûteux que dans des environnements traditionnels.
- Les aspects d'architecture sont complexes. Il est difficile de trouver des ressources expertes dans le domaine de l'architecture technique. Cette pénurie se retrouve également dans la population des développeurs d'applications.
- J2EE donne une grande latitude aux développeurs dans les choix de mise en oeuvre. Ces derniers étant souvent peu expérimentés, les applications risquent d'être difficiles à maintenir et d'avoir une structure peu évidente. En conséquence, les phases de transfert de connaissances sont difficiles et coûteuses.
- La technologie est en constante évolution. Pour rester efficaces et réactives, les équipes doivent consacrer une part non négligeable des budgets à la veille technologique. De même, les coûts de maintenance liés au maintien de la compatibilité des solutions avec les standards sont élevés.

Aujourd'hui, les entreprises qui travaillent dans l'environnement Java ont expérimenté et identifié les problèmes soulevés par la mise en oeuvre de cette technologie. Si les bénéfices associés sont indéniables, il est néanmoins nécessaire de répondre à ces problématiques.

Les solutions : état de l'offre

Les besoins fondamentaux des équipes de développement J2EE s'articulent autour des problématiques suivantes :

- Architecture et Conception,
- Développement et codage de la solution,

- Déploiement dans une architecture distribuée,
- Gestion de la persistance des données.

L'offre logicielle existante permet de répondre à ces différents points de façon complète :

- Les aspects architecture et conception sont couverts par des outils de modélisation,
- Les IDE permettent de faciliter le codage des applications,
- les serveurs d'applications autorisent le déploiement des solutions,
- les gestionnaires de base de données stockent les données et facilitent leur manipulation.

Cependant, par rapport aux préoccupations énoncées précédemment, il apparaît qu'un besoin fondamental n'est pas couvert : la cohérence nécessaire entre l'architecture établie et le développement proprement dit. En effet, il existe autant de façons de mettre en oeuvre une fonctionnalité qu'il existe de développeurs. Ceci ne facilite ni les tâches de chiffrage, ni les tâches de maintenance. De plus,



des fonctionnalités récurrentes, indépendantes de la problématique métier, doivent sans cesse être redéveloppées ; elles le sont souvent de façon hétérogène.

La structuration des développements d'une part et l'accélération de la mise à disposition des composants techniques d'autre part sont les problématiques auxquelles répondent les frameworks techniques.

L'intérêt d'un framework technique

On peut définir les frameworks comme un ensemble de règles et d'outils qui structurent et accélèrent le processus de fabrication et de maintien en condition opérationnelle d'un projet informatique.

Le principal bénéfice de la mise en oeuvre d'un framework technique est l'amélioration globale de la qualité des projets par la standardisation des architectures et la réutilisation de composants.

Un bon framework technique doit fournir les services suivants :

- Un ensemble de composants standardisés : Quelle que soit l'application développée, des besoins récurrents existent : processus de connexion à l'application, gestion de droits utilisateurs, affichage de fenêtres standards (telles que des interfaces de recherche), respect d'une charte graphique partagée entre les différentes applications, accès à des données stockées dans des formats différents, log d'événements, gestion d'exceptions, ...
- Une architecture standardisée : L'une des clés de la réutilisation de composants est leur standardisation. Il est donc nécessaire que

l'architecture proposée par le framework technique s'intègre complètement aux standards, mais également qu'elle ait déjà été éprouvée en situation de production. De même, la standardisation des architectures favorise le turn-over des développeurs au sein des équipes de développement et améliore les phases de maintenance.

- Une documentation complète : L'un des buts recherchés dans la mise en œuvre d'un framework technique est l'accélération des développements. Il serait dommage qu'une mauvaise documentation vienne ralentir la phase de prise de connaissance et retarde le début des développements.
- Un support réactif : Toujours dans l'optique de raccourcir les phases de développement, mais aussi de sécuriser l'exploitation, la réactivité du support et la rapidité à résoudre les problèmes rencontrés par les utilisateurs sont des critères essentiels dans le choix de l'outil.
- Une gamme de services associés : Outre le support du logiciel, les services de formation et d'assistance au démarrage de projet permettent aux équipes de démarrer plus rapidement les phases de développement proprement dites, en respectant dès le début les meilleures pratiques définies.
- Une intégration au fil de l'eau des évolutions technologiques : Comme nous l'avons vu précédemment, l'un des gros avantages de la plateforme J2EE est sa capacité à évoluer et à

proposer des innovations technologiques au fur et à mesure que les besoins apparaissent. Malheureusement, cet avantage peut rapidement se transformer en inconvénient : pour en tirer complètement profit, il est nécessaire de conserver un poste de veille technologique actif, sans être assuré que les évolutions étudiées aient réellement un intérêt. Le fournisseur du framework doit donc assurer l'intégration de ces évolutions, de la façon la plus transparente possible pour les utilisateurs.

Les solutions

A l'heure actuelle, il existe plusieurs types de réponses logicielles :

- Les frameworks " maison " : De nombreuses sociétés pionnières dans les développements J2EE ont déjà conclu à l'intérêt d'utiliser un framework technique comme socle pour l'ensemble de leurs développements. Après une première expérience de réalisation d'application, la nécessité de réutiliser certains des composants dans les applications futures est clairement apparue comme une priorité, afin d'éviter de nouveaux surcoûts. Seulement, un composant développé dans une optique très particulière doit souvent être retravaillé pour devenir générique. Cette phase de refonte est souvent très consommatrice de ressources (il faut revoir la conception de la brique technique, imaginer a priori les différents cas d'utilisation, les implémenter, les tester, etc...). Les frameworks maison arrivent assez bien à atteindre l'objectif de réutilisation des composants et à proposer une architecture standardisée complète. Cependant, les aspects complémentaires, qui apparaissent comme étant nécessaires, sont la plupart du temps ignorés : la documentation est réduite, le support peut devenir défaillant si l'équipe n'est pas maintenue, la formation est souvent transmise de façon informelle. Ce dernier point peut devenir critique lorsque les équipes sont renforcées par de nouveaux collaborateurs, ou par des consultants externes. Enfin, l'aspect veille technologique est assuré par les équipes de développement. A nouveau, des ressources sont affectées à des tâches non rémunératrices pour l'entreprise.
- Les frameworks open-source : Ce type de framework constitue une possibilité intéressante pour disposer de composants techniques fiables et en constante évolution. Ils offrent en effet une réponse techniquement très



aboutie sur des problématiques très ciblées. Les composants techniques qu'ils proposent correspondent à des problématiques déjà rencontrées dans des contextes similaires et sont parfois déjà validés dans des environnements de production. Une documentation plus ou moins complète est disponible : elle permet de faciliter les processus de formation et de mise en œuvre. Cependant, il est souvent nécessaire de rechercher activement l'information, au travers de forums d'utilisateurs par exemple. Cette solution n'est pourtant pas exhaustive : en effet, la spécialisation de ces frameworks sur un aspect de l'application en particulier (framework de présentation, d'accès aux données, ...) implique la juxtaposition de composants hétérogènes pour répondre aux besoins globaux des applications. Il reste nécessaire de maintenir une équipe d'architecture efficace, dont la mission principale sera de garantir la cohérence de ces briques techniques dans les applications produites. De plus, même si la documentation est riche, l'utilisateur doit chercher activement les informations. En cas de bug, ou de besoin urgent de solution, la réactivité n'est pas garantie. Le problème est le même pour ce qui est de la formation : afin d'assurer une prise en main rapide et homogène dans l'entreprise, il est nécessaire soit de développer ses supports de cours soi-même, soit de recourir à une société tierce réalisant ce type de prestation. Enfin, l'aspect veille technologique reste à la charge des équipes : même si le framework évolue,



les développeurs doivent continuer à estimer par eux-mêmes la pertinence de l'intégration des nouvelles versions dans leur environnement, les risques induits par les montées de version restant à la charge des équipes.

- Les frameworks progiciels : Un nouveau type d'offre apparaît aujourd'hui dans le domaine des frameworks techniques. Ce sont les frameworks progiciels. Ils sont produits et supportés par des sociétés commerciales et offrent à la fois les composants techniques évolués et les services associés. Les frameworks progiciels doivent apporter une réponse à chacun des points présentés précédemment. Ils offrent, en plus d'un ensemble complet de briques logicielles, l'assurance de développer une application correctement structurée dans sa globalité. Ainsi, les phases de maintenance, ou d'intégration de nouveaux membres dans l'équipe sont améliorées. Les modules respectent des standards de développement, sont testés fonctionnellement, ont subi des tests de charge et sont validés dans différents environnements techniques (serveurs d'applications, bases de données, ...). Ils sont livrés avec une documentation complète qui facilite leur prise en

main et sont accompagnés de services de maintenance et de formation qui accélèrent les phases non productives du cycle de développement.

La démarche " progiciel " semble la plus apte à répondre aux problématiques rencontrées.

Comment choisir entre ces différents outils ?

La plupart des sociétés qui réalisent des développements J2EE savent aujourd'hui que la réussite de leurs applications passe par la réutilisation des composants et la structuration homogène de leurs architectures. Comme nous l'avons vu, l'adoption d'un framework technique est la réponse la plus adaptée à ce double objectif. La question principale est maintenant de savoir comment s'équiper.

La réponse doit être guidée par trois critères principaux : la couverture fonctionnelle de la solution : afin de garantir des applications homogènes, il est primordial de s'appuyer sur des architectures techniques suffisamment complètes pour être réutilisées dans des contextes fonctionnels différents. L'homogénéité des applications permet d'optimiser le turn-over et simplifie la maintenance et les évolutions des applications.

- le coût d'acquisition du produit : il intègre le coût du framework lui-même (coût de développement ou coût des licences), celui des services associés (installation et mise en œuvre, formations, temps de prise en main, support et corrections), mais également des activités annexes auxquelles on ne pense pas forcément lors du choix (veille technologique nécessaire pour assurer la pérennité du framework).

- la fiabilité du produit : la finalité d'une application est bien sûr d'être utilisée en production. Plus vite une application est exploitée, plus vite elle devient rentable. Développer sur des socles techniques non validés par ailleurs peut conduire à de gros désagréments lors des déploiements et des retards importants dans la mise en place des solutions.

Aujourd'hui, afin de pallier aux manques des unes et des autres, des solutions progicelles apparaissent. L'offre est donc maintenant complète et permet à tous les créateurs d'applications de trouver une solution adaptée à leurs besoins.

■ Stéphane Génin

Responsable de la validation des choix techniques et de l'organisation des équipes de recherche et développement, avant-vente et support au sein d'Ideo Technologies

DEVENEZ Développeur PHP

(certifié Anaska Formation)

Formation Apprentissage PHP (3 jours) : 990 € ht

Profitez du PACK Apprentissage pour
Apprendre à réaliser des projets InterNet,
Apprendre à créer des espaces sécurisés,
Apprendre à utiliser des bases de données,

Autres formations PHP :

Maîtrise PHP (4 jours) - Expert PHP (2 jours) -
Sur mesure (étude de projet, PHP/JAVA, templates,...)

VENEZ DECOUVRIR TOUTES NOS FORMATIONS* SUR :
<http://www.anaska.com>

Anaska : 117 rue de Crimée - 75019 - PARIS
Tél. : 01 43 81 92 91 - Fax : 01 41 53 00 74

*PHP, ASP, JSP, Sécurité, Serveurs, Sendmail

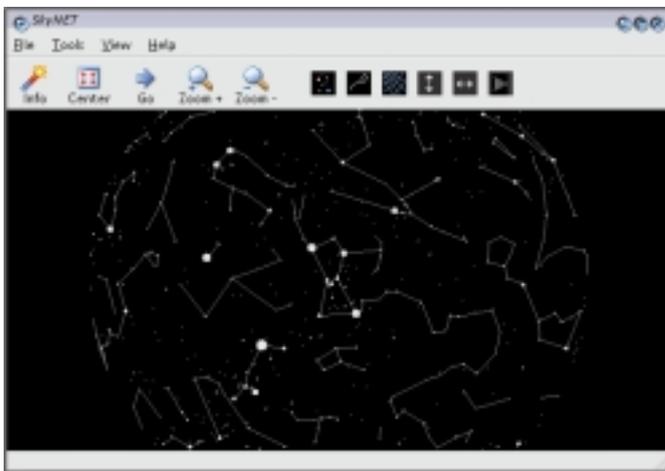


Migrer de Microsoft ASP.NET ou PHP, vers Mono XSP/ASP.NET, est ce possible ?

Est-il réalisable aujourd'hui de passer d'ASP.NET sous Windows, et/ou de PHP sous Linux, à un environnement homogène Mono ASP.NET ?

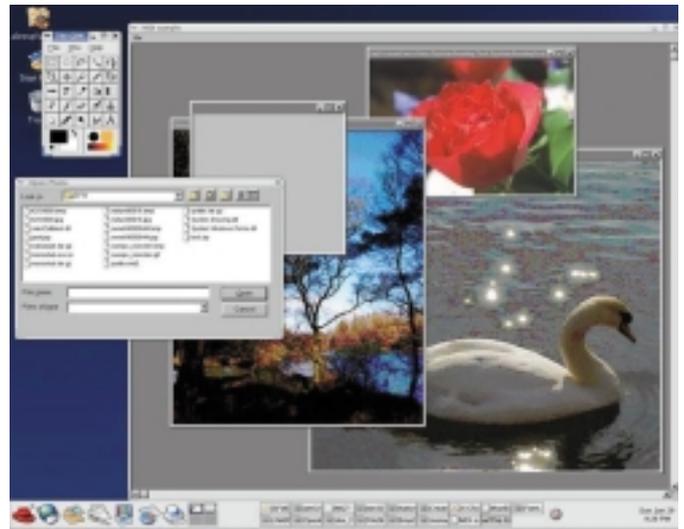
Ce mois-ci, nous avons rencontré un consultant (développeur) indépendant, lecteur de *Programmez de surcroît*, avec lequel nous avons discuté de longues heures des problèmes de migration qu'il rencontrait en ce moment (en raison de la non hétérogénéité des outils de développement qu'il utilise).

Les projets développés sont essentiellement des projets de gestion. La machine de développement est un serveur Windows 2000, comprenant Apache comme serveur http. Lorsque le client demande une application intégrée à un site Web, c'est PHP qui est choisi, sinon c'est VB.NET qui est employé. Du côté des bases de données, SQLSERVER ou MySQL représentent les deux options proposées. La base de données est modélisée avec un outil comme Open-Modeling (<http://open-modeling.sourceforge.net>), ce qui permet de la créer indépendamment du système d'exploitation cible et du langage. Cet outil est gratuit, ce qui n'est pas un luxe. Il arrive que PHP échoue en interrogeant des services Web distants, et dans ce cas, c'est java qui est utilisé comme colle (le code



PHP appelant directement java). Remarquez qu'il existe sans doute une solution technique en PHP, mais le développeur doit penser avant tout à respecter un délai de réalisation (une "deadline", un peu à l'image de la presse). Et par conséquent *"il n'a pas de temps à perdre avec ce genre de situation : il faut que cela tourne, vite et bien"*.

Une application .NET, VB.NET plus précisément, est facile à mettre en œuvre. Le code n'est pas compliqué à comprendre, et le déploiement des applications chez les clients est facile. Le code source est fourni (sans recours à un obfuscateur), ce qui représente un avantage non négligeable pour tout le monde (après livraison de l'application et paiement de celle-ci). L'administration est simple et le paramétrage aisé.



PHP et VB.NET possèdent d'ailleurs les mêmes qualités par rapport à un code non managé en C++ par exemple. Bref, il s'agit d'outils productifs. Ajoutons qu'il est apparu que la modélisation de la couche de présentation est très spécifique avec .NET ; et qu'enfin la formation d'un développeur de VB à VB.NET ou ASP.NET prend aussi du temps (+ 1 an selon les capacités de l'individu).

Comme nous l'avons dit, PHP souffre d'un problème d'intégration car les services Web et le middleware de développement en général sont très loin d'être normalisés. Ce qui porte aussi préjudice à .NET. Car ici, Microsoft paie comptant sa mauvaise campagne de communication du "tout ou rien, XML/Services Web". En effet, si l'appel interservices entre Linux et Windows ne fonctionne pas convenablement, alors pourquoi ne pas écrire toute la plomberie en Java ?

Mais n'existe-t-il pas une autre alternative ? Comme de développer en ASP.NET sous Windows, Linux et MacOS X par exemple ?

Mono à la rescousse

Un développeur indépendant peut aujourd'hui réfléchir sérieusement à Mono. Si vous devez migrer vers un outil, autant qu'il soit utilisable sur le plus grand nombre de systèmes d'exploitations possible. Java, à ce titre, est une solution mais lorsqu'il est question de migration, reste à déterminer le temps de formation nécessaire à celle-ci. Il sera plus facile pour un programmeur de migrer de VB vers VB.NET/ASP.NET, plutôt que de passer de VB à java. L'aspect d'une application java/Swing exécutée sur différents OS ne sera pas homogène non plus, et esthétiquement une interface portable écrite en QT ou GTK sera plus réussie (sans compter la réputation de lenteur, non justifiée sans doute de nos jours, que traîne java).

"Mono" signifie singe en espagnol. Cette initiative a débuté en 2001, du temps où la société Ximian (<http://www.go-mono.com>) n'appartenait pas encore à Novell (<http://linuxfr.org/2003/08/04/13518.html>). L'objectif

de ce projet est de créer un environnement d'exécution CLR (Common Language Runtime) et un canevas commun (Framework) à plusieurs langages. Par Framework, il faut comprendre un "ensemble de classes conçues pour aider le développeur dans un domaine précis". Le but du jeu est d'implémenter sous Linux l'environnement d'exécution de la plate-forme dot Net, ainsi que des compilateurs, et un jeu de classes compatible.

Le point de la situation

Actuellement, il est concevable de migrer presque sans obstacles une application ADO.NET ou ASP.NET. Mais attention, l'ensemble des classes du Framework .NET de Windows est loin d'être porté. C'est un projet gigantesque, car il y a près de 3 000 classes à réécrire.

Le cœur de Mono s'articule autour des logiciels du monde open source. Citons par exemple :

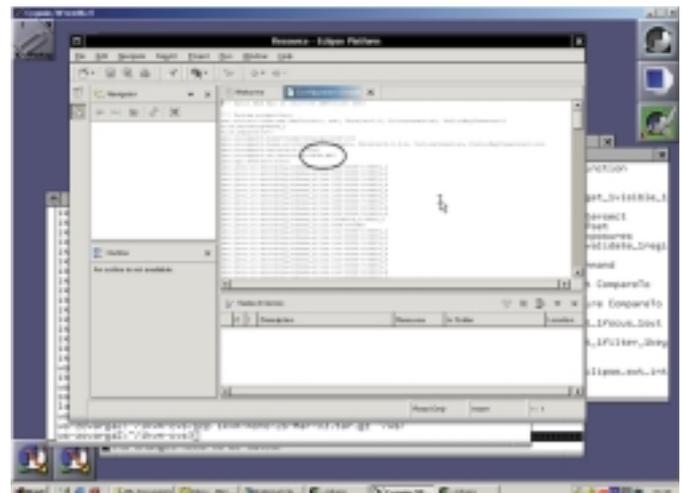
- l'algorithme du ramasse miettes provenant du modèle ORP (Open RunTime Platform) d'Intel (en open source : <http://www.intel.com/research/mrl/orp/>) ;
- Le runtime, qui comprend également un chargeur de classes (ClassLoader) et un système de gestion de threads indépendant du modèle API Win32. La création du code assembleur est inspirée du projet GNU LIGHTNING (<http://www.gnu.org/software/lightning/lightning.html>).
- Enfin un désassembleur (Monodis) qui permet de visualiser le code CIL. Mono tourne sous Linux, Mac OS X, BSD et Windows (dont Windows 95 !). Les langages supportés comprennent le VB.NET ainsi que le C#. D'excellents environnements de développement commun sous Windows et Linux sont Emacs (non recommandé si vous n'êtes pas initié), ou encore Eclipse (assez lourd). SharpDevelop est écrit en C# et ne tourne pour l'instant que sous Windows (parfait pour le C#). Toujours sous Windows, rien ne vous empêche de développer avec l'outil gratuit de Microsoft : WebMatrix.

L'installation

L'installation de Mono sous Windows passe par un "setup" classique. Aucune difficulté en vue. Sous Linux, vous pouvez installer les binaires au format RPM (Redhat), ou encore compiler les sources après les avoir rapatriées à l'aide d'un client CVS (notons aussi qu'une source apt-get est disponible pour Debian : lisez à ce propos <http://www.atoker.com/mono/>). Un "daily snapshot" est disponible sur le site de Mono (une mise à jour du code source journalière).

Installation de Mono sous Debian Woody :

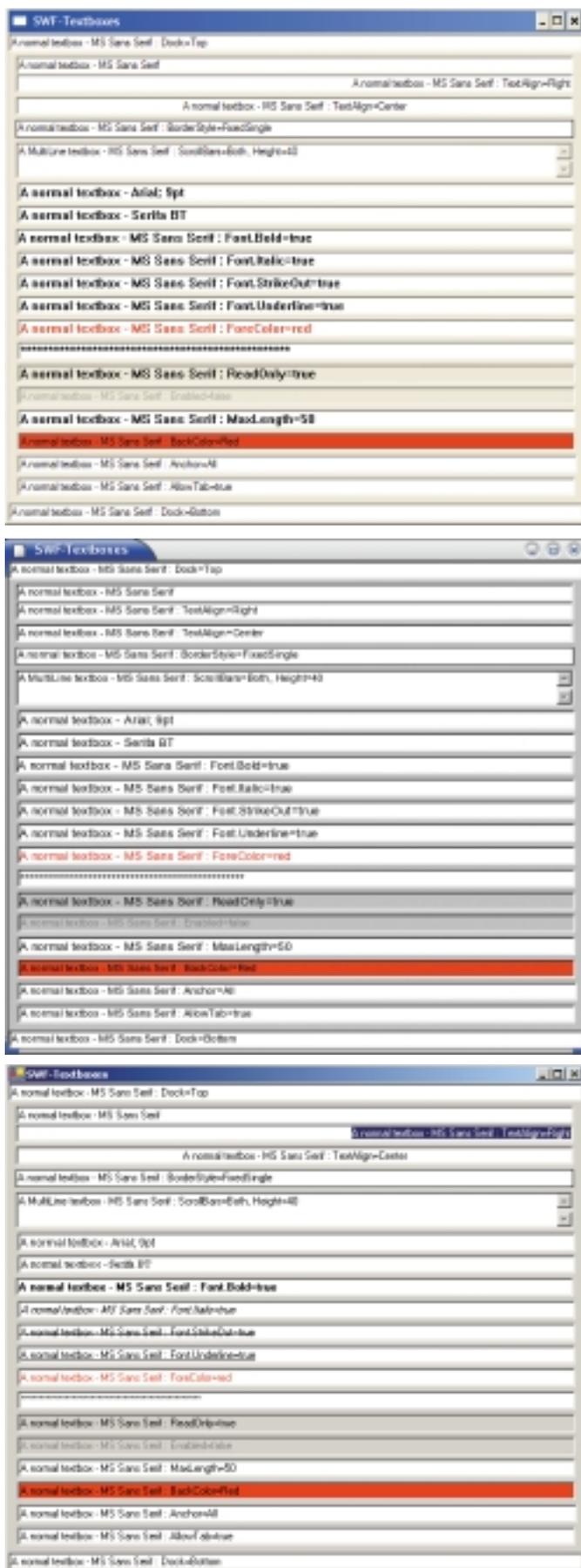
```
#nano /etc/apt/sources.list
...
deb http://www.debianplanet.org/mono stable main
deb-src http://www.debianplanet.org/mono stable main
...
# apt-get install mono
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  binfmt-detector-cli binfmt-support libgc6 libmono0 libxml2-dev libxslt1-
  dev mono-assemblies mono-common mono-jit mono-mcs mono-utils
```



The following NEW packages will be installed:

```
binfmt-detector-cli binfmt-support libgc6 libmono0 libxml2-dev libxslt1-
dev mono mono-assemblies mono-common mono-jit mono-mcs mono-utils
0 packages upgraded, 12 newly installed, 0 to remove and 13 not upgraded.
Need to get 9269kB of archives. After unpacking 31.8MB will be used.
Do you want to continue? [Y/n]
```

A cela s'ajoute l'installation de XSP dont nous parlons plus loin. Ceci dit, le compilateur de Mono n'est pas intégré à GCC. Les composants majeurs de Mono ont été programmés en C#, et le moteur JIT runtime de Mono peut interpréter du code CIL (Common Intermediate Language) en provenance directe de Windows. Aucune recompilation n'est nécessaire. Développer des applications graphiques sous Linux en partant de l'espace de nom des SWF (System.Windows.Forms) n'est pas un pari gagné d'avance, car ce "toolkit" est propriétaire, et non portable. A l'heure où ces lignes sont rédigées, 55% seulement de l'espace de nom a été porté. Vous pouvez toujours passer par WinLIB, mais il s'agit d'émulation, ce qui n'est pas toujours d'une fiabilité, ni d'une rapidité à toute épreuve. Alors un conseil : ne passez par Winlib que si vous partez d'un existant sous Windows sans avoir pu réfléchir dès le départ à la porta-



bilité des interfaces. GTK# (GIMP Toolkit) est un "sous-ensemble" compatible SWF avec lequel vous préférerez sans doute travailler. Une autre proposition est de solliciter QT# (<http://qtsharp.sourceforge.net>) sous licence GNU GPL. Ces systèmes (GTK# et QT#) ne sont pas encore parfaitement finalisés mais sont déjà utilisables par MONO sous Windows ou Linux. Les couples C#/QT# ou C#/GTK# sont plus rapides que java/Swing, et offrent parfois de meilleures fonctionnalités que le SWF de Windows (c'est le cas des applications multi langues).

XSP

Les Webforms de Mono sont fonctionnelles. Le nom de code de cette implémentation ASP.NET est XSP dont l'archive est téléchargeable sur le site de Mono. Il embarque son propre serveur http (tournant sur le port 8080). Un module apache est également disponible (mod_mono). Toute une panoplie de pages ASPX d'exemples est livrée avec XSP et presque tous les contrôles ASP.NET ont été migrés ! Cerise sur le gâteau, si vous utilisez Mono comme environnement de compilation et de déploiement, un outil vous permettra de "monter" et de "déployer à chaud" des services Web !

Pour les connexions aux bases de données vous pouvez lancer l'outil graphique SQL#. L'accès aux bases de données suivantes est fonctionnel : IBM DB2 Universal Database, MySQL, ODBC, Oracle, OLE DB, PostgreSQL, Microsoft SQL Server, SQL Lite, Sybase et TDS Generic.

Mono ne se contente pas ici de viser l'objectif de la portabilité stricte mais améliore .NET en ajoutant, par exemple, des extensions au framework (comme "Mono.Data.MySQL"). Votre code comme par miracle, s'exécute sous Linux mais aussi Windows 9x ou Mac OS X, ce qui n'est pas le cas de dot NET. Enfonçons le clou : certaines fonctionnalités de GTK# et de QT# sont plus efficaces que le SWF de Microsoft et sans exagérer le "Mono.AppServer" est un outil de déploiement extraordinaire.

Pour conclure

Un développeur indépendant peut envisager sérieusement de migrer d'ASP.NET/PHP vers Mono ASP.NET avec accès à des bases de données. Mono est un environnement gratuit et portable (en ce qui concerne le monde Unix, l'interpréteur tourne sous Linux/x86, Linux/PPC, S390, StrongARM, SPARC, HPPA, SPARC v9).

Et si vous modélisez vos objets et bases de données avec un outil indépendant, c'est une raison supplémentaire de faire le pas.

Par contre du côté des applications textuelles ou graphiques non Web (GTK# ou QT#), l'état de développement des classes du framework .NET n'est pas (encore) assez avancé pour envisager sereinement une migration.

Voici un extrait d'un code source présentant une série de contrôles de saisies de type "TextBox". Ce code a été exécuté sous Linux Mono, Windows mono, et Windows Microsoft dot net.

La comparaison des trois captures d'écrans est intéressante, car on y constate finalement peu de différences.

```
using System;
using System.Windows.Forms;
```

```

namespace MyFormProject
{
    class MainForm : System.Windows.Forms.Form
    {
        //
        //Define the controls
        //
        private System.Windows.Forms.TextBox textBox;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.TextBox textBox4;
        private System.Windows.Forms.TextBox textBox5;
        private System.Windows.Forms.TextBox textBox6;
        private System.Windows.Forms.TextBox textBox7;
        private System.Windows.Forms.TextBox textBox8;
        private System.Windows.Forms.TextBox textBox9;
        private System.Windows.Forms.TextBox textBox10;
        private System.Windows.Forms.TextBox textBox14;
        private System.Windows.Forms.TextBox textBox11;
        private System.Windows.Forms.TextBox textBox12;
        private System.Windows.Forms.TextBox textBox13;
        private System.Windows.Forms.TextBox textBox15;
        private System.Windows.Forms.TextBox textBox16;
        private System.Windows.Forms.TextBox textBox17;
        private System.Windows.Forms.TextBox textBox18;
        private System.Windows.Forms.TextBox textBox19;
        private System.Windows.Forms.TextBox textBox20;

        public MainForm()
        {
            InitializeComponent();
        }

        void InitializeComponent()
        {
            //
            //Initialize the controls and set their properties
            //
            this.textBox19 = new System.Windows.Forms.TextBox();
            this.textBox3 = new System.Windows.Forms.TextBox();
            this.textBox20 = new System.Windows.Forms.TextBox();
            this.textBox18 = new System.Windows.Forms.TextBox();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.textBox6 = new System.Windows.Forms.TextBox();
            this.textBox7 = new System.Windows.Forms.TextBox();
            this.textBox4 = new System.Windows.Forms.TextBox();
            this.textBox5 = new System.Windows.Forms.TextBox();
            this.textBox10 = new System.Windows.Forms.TextBox();
            this.textBox8 = new System.Windows.Forms.TextBox();
            this.textBox9 = new System.Windows.Forms.TextBox();
            this.textBox13 = new System.Windows.Forms.TextBox();
            this.textBox12 = new System.Windows.Forms.TextBox();
            this.textBox11 = new System.Windows.Forms.TextBox();
            this.textBox = new System.Windows.Forms.TextBox();
            this.textBox17 = new System.Windows.Forms.TextBox();

```

```

            this.textBox16 = new System.Windows.Forms.TextBox();
            this.textBox15 = new System.Windows.Forms.TextBox();
            this.textBox14 = new System.Windows.Forms.TextBox();
            this.SuspendLayout();

            //
            // textBox19
            //
            this.textBox19.Dock = System.Windows.Forms.DockStyle.Top;
            this.textBox19.Location = new System.Drawing.Point(0, 0);
            this.textBox19.Name = "textBox19";
            this.textBox19.Size = new System.Drawing.Size(627, 20);
            this.textBox19.TabIndex = 19;
            this.textBox19.Text = "A normal textbox - MS Sans Serif : Dock=Top";

            ...

            //
            // MainForm (Add the controls to the form and set its properties)
            //
            this.ClientSize = new System.Drawing.Size(627, 538);
            this.Controls.AddRange(new System.Windows.Forms.Control[] {
                this.textBox20,
                this.textBox19,
                this.textBox18,
                this.textBox17,
                this.textBox16,
                this.textBox15,
                this.textBox14,
                this.textBox13,
                this.textBox12,
                this.textBox11,
                this.textBox10,
                this.textBox9,
                this.textBox8,
                this.textBox7,
                this.textBox6,
                this.textBox5,
                this.textBox4,
                this.textBox3,
                this.textBox2,
                this.textBox,
                this.textBox1});
            this.Text = "SWF-Textboxes";
            this.ResumeLayout(false);
        }

        [STAThread]
        public static void Main(string[] args)
        {
            Application.Run(new MainForm());
        }
    }
}

```

■ **Xavier.Leclercq** - Xavier.Leclercq@programmez.com

Développer un framework PHP

(suite)

Smarty, comme Smart.

Dans notre précédent article sur les méthodes de développement en PHP, nous avons abordé l'aspect organisation. Nous avons laissé de côté tout l'aspect présentation, pourtant pierre angulaire d'un site web réussi. Si nous n'affichons rien, je doute que notre positionnement au hit parade soit des plus brillants (même si, réflexion faite, un site comme perdu.com a gagné sa célébrité de par son absence de contenu). Templates et moteurs de templates en PHP, ou "comment procéder pour afficher correctement des pages HTML".

L'art de l'affichage

La question n'est pas là, nous voulons (si, nous le voulons), afficher des informations à l'écran.

Nous pouvons le faire, via deux méthodes:

- echo, printf & autres variantes.
- Contenu hors des balises php.

Notre démonstration en 5 lignes de code :

```
<?php
echo "avec un echo", "toujours avec un echo";
printf ("avec un printf c'est pareil");
?>
<!-- du html -->
Et voilà comment faire en dehors de php sans echo ni printf.
<!-- encore du html -->
```

(Note aux hackers chevronnés : oui, nous pouvons aussi ouvrir un socket sur un serveur distant pour rediriger ce que nous lisons sur le poste client, nous pouvons utiliser la gdlib, générer du pdf, des images, lire des fichiers, mais soyez raisonnables, nous cherchons à démontrer un point important)

Maintenant que nous sommes rassurés sur notre capacité à afficher des éléments sur le navigateur de nos internautes, interrogeons-nous sur la façon de procéder. La première considération : "pourquoi ne pas dire ce que j'ai à dire au moment où j'ai envie de le dire ?". La deuxième, plus réfléchie : "pensons à tout ce que nous avons à dire et réfléchissons sur la forme et sur la manière de l'annoncer". Malgré mes efforts pour ne pas entamer le suspens, vous vous doutez que nous allons opter pour la deuxième solution. Tentons de démontrer pourquoi, avec un exemple volontairement dédié, mais proche de la réalité.

Le scénario d'un problème

Vous développez le site personnel d'un ami, non informaticien, qui s'étonne tous les jours de l'étendue de vos talents (profitons-en, nous sommes entre nous). Cet ami a tenu, devant la somme de travail vous étant assignée, à prendre en charge la partie graphique. Ce n'est pas plus mal me direz-vous, vous êtes informaticien, pas graphiste... quoique, à la vue de la charte produite par votre ami, vous nourrissez des doutes (mais, après tout, c'est son site).

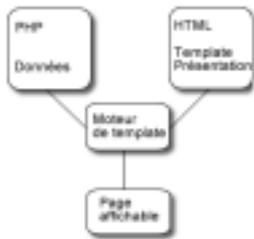
Vous voici muni d'un ensemble de code HTML pour le transformer en un système de news.

Courageusement, vous commencez à insérer au milieu des balises le code PHP pour vous connecter à la base, lancer vos requêtes, transformer les multiples tables imbriquées, pour finir par obtenir la première page de news du site. Excellent résultat, félicitations ! Votre ami, par souci de pouvoir satisfaire tous les anciens élèves de son école à qui le site est destiné, souhaite disposer d'une fonctionnalité d'export pour que ces nouveaux travailleurs (les anciens élèves... suivez un peu) puissent télécharger les nouvelles dans leur PDA. Idée comme une autre, vous vous exécutez (dans le sens "vous procédez au développement"). Vous récupérez les bribes de code PHP de connexion à la base, en extrayez les parties graphiques et obtenez un nouveau fichier *news_export.php*.

Maintenant que tout commence à fonctionner, votre ami souhaite renommer le champ "content" en "contenu". Même si cela ne représente rien de bien compliqué, vous en convenez, il vous faut changer vos deux fichiers (celui affichant les news et celui les exportant) pour une seule modification qui n'a rien à voir avec la façon d'afficher les éléments.

Votre ami, toujours dans un souci de ne pas vous accabler, essaie de lui-même de changer la charte graphique, et le site tombe en panne. Après avoir passé la moitié de la soirée qui suit à lui expliquer que echo n'est pas un attribut de style, vous réintégrez tous les éléments supprimés.

L'histoire peut continuer indéfiniment, mais il est évident qu'une telle éthique de programmation n'est pas satisfaisante car vous passez votre temps à refaire ou à corriger ce que vous avez déjà fait, et si vous maintenez plusieurs sites de front, vous n'aurez bientôt plus de temps à vous, ni d'amis d'ailleurs. Vous noterez également que dans des conditions d'entreprise, ce problème est d'autant plus prononcé, dans le sens où les acteurs (graphistes, clients, développeurs et chefs de projets) sont tous distants et que les avis comme les besoins changent (ou sont affinés) tout au long de la phase de développement. Nous ne pouvons pas nous permettre de remettre en cause nos procédés à chaque changement d'interface (un peu comme si nous devions ressaisir tout le contenu d'un document texte pour pouvoir changer la couleur d'un titre). De même, dans le milieu professionnel, vous serez souvent amené à vouloir réutiliser du travail que vous avez déjà fait, à la condition que cette tâche de reprise ne représente pas plus de temps que la refonte du système.



La solution

Arrive une idée de génie, née de ce constat : “séparons la logique de présentation de la logique de récupération des informations.”

Nous décidons donc de procéder à des changements majeurs dans notre façon de faire, et transformons notre code précédent en code séparé. Nous allons, pour ce faire, utiliser un moteur de templates appelé Smarty (<http://smarty.php.net>) qui a la particularité de compiler pour nous ses templates en PHP (nous reviendrons sur ce que cela implique, et nous reviendrons aussi sur l'utilité d'un moteur de template).

```
<?php
include ('Smarty.class.php');
$noMoreHTMLInPhp = & new Smarty ();

//récupération de nos éléments graphiques.
mysql_connect ();
mysql_query ('select id_news, title_news, content_news from news');
while ($r = mysql_fetch_object ()) {
    $noMoreHTMLInPhp->append ('NewsList', $r);
}
mysql_close ();

//nous en avons terminé avec la récupération des news, procédons à l'affichage
$noMoreHTMLInPhp->display ('news.list.tpl');
?>
```

Ce que nous avons fait est simple. Dans cette page, il n'existe que du code PHP, bien identifiable, pour récupérer des données depuis une base. Au tout début, nous réalisons l'inclusion de la classe Smarty, le moteur de template. Nous nous connectons à notre base, lançons la requête, puis stockons au fur et à mesure les résultats dans l'objet Smarty. Une fois que nous en avons terminé avec l'accumulation des news, nous demandons à Smarty de les afficher dans un template nommé *news.list.tpl*.

Fichier news.list.tpl

```
<table>
<tr><th>Titre</th><th>Contenu</th></tr>
{foreach from=$NewsList item=$new}
<tr><td>{$new->title_news}</td><td>{$new->content_new}</td></tr>
{/foreach}
</table>
```

Voilà ce que l'on appelle un template, un fichier qui ne contient que de la logique de présentation. Si nous nous souvenons bien de notre problème d'export, il n'y a rien de bien compliqué pour répondre au besoin : “Il nous suffit de créer un nouveau template”.

news.export.tpl

```
{foreach from=$NewsList item=$new}
{$new->title_news};{$new->content_new}
```

```
{/foreach}
```

Dans notre fichier PHP il nous suffit alors de rajouter un traitement particulier qui choisit comment afficher nos informations, par exemple.

```
<?php
//[...]
//Voilà la seule ligne de news.php que nous devons modifier
$noMoreHTMLInPhp->display (isset ($_GET['export']) ? 'news.export
.tpl' : 'news.list.tpl');
?>
```

Dorénavant, lorsque notre système de news rencontrera “export” dans l'URL, il affichera ses données dans un template dédié à cet export, plutôt que de les afficher dans un tableau HTML. Si vous voulez rajouter le champ “résumé”, seul le fichier de récupération des news est à modifier. Pour parfaire notre fonctionnalité de news, nous décidons de proposer une fonctionnalité d'export RSS, pour permettre aux autres sites d'afficher et d'utiliser nos news. Encore une fois, rien de plus simple, il suffit de créer un troisième template pour répondre au besoin. Par la même occasion, nous pouvons même systématiser le choix du template par une variable dédiée dans nos URLs (dans notre exemple kind).

```
<?php
//Voilà le code de notre récupération de news.
$noMoreHTMLInPhp->display (Template::get ('myTemplate'));
?>
```

et pour être exhaustif, voici le code de notre méthode `Template::get ($fileID)`;

```
<?php
class Template {
function get ($fileID){
    $kind = isset ($_GET['kind']) ? $_GET['kind'] : '';
    if (is_readable ($fileID.'.'.$kind.'.tpl')){
        return $fileID.'.'.$kind.'.tpl';
    }else{
        return $fileID.'.tpl';
    }
}
}
?>
```

En construisant nos liens correctement, nous sommes maintenant capables de choisir tout type de template pour un contenu donné, sans nous soucier de ces choix dans les appels de nos méthodes.

Mais pourquoi un “moteur” de templates ?

Les esprits critiques se sont réveillés et s'interrogent de fait sur la nécessité d'un “moteur” de templates. Attention, nous ne nous interrogeons pas sur la nécessité d'utiliser des templates (les fichiers), mais sur la nécessité d'utiliser un moteur de template (l'objet qui effectue le rapprochement donnée / fichier).

En effet, en PHP, nous disposons d'un superbe moteur de template : PHP lui-même. Qu'est-ce qui nous empêche de préférer la syntaxe `<?php`

echo \$monInformation; ?> à {\$monInformation}. Après tout, nous connaissons déjà PHP, inutile d'apprendre un nouveau langage, et niveau performances, inutile d'inclure une bibliothèque volumineuse pour afficher nos informations. De plus, le temps d'analyse du fichier est sûrement long, et si j'utilise PHP, je suis sûr que PHP fonctionnera là où PHP fonctionne ??? (ce qui n'est pas le cas avec une bibliothèque : je ne suis pas sûr que ma bibliothèque fonctionne sur cette "plate-forme PHP et cette configuration" particulière).

Nous aurions donc un système PHP du style:

```
<?php
include ('Smarty.class.php');
$noMoreHTMLInPhp = array ();

//récupération de nos éléments graphiques.
mysql_connect ();
mysql_query ('select id_news, title_news, content_news from news');
while ($r = mysql_fetch_object ()) {
    $noMoreHTMLInPhp['NewsList'][] = $r;
}
mysql_close ();

//nous en avons terminé avec la récupération des news, procédons à l'affichage
include ('templates/news.list.php');
?>
```

Maintenant, notre template "PHP"

```
<table>
<tr><th>Titre</th><th>Contenu</th></tr>
<?php foreach ($noMoreHTMLInPhp['NewsList'] as $new) { ?>
<tr>
<td><?php echo $new->title_news; ?></td>
<td><?php echo $new->content_new; ?></td>
</tr>
<?php } ?>
</table>
```

Bien, oui, vous avez raison. Toutefois, il existe de nombreux avantages à utiliser un moteur de templates, dans le sens où ce dernier est paramétrable selon vos souhaits. Vous voulez que tous les caractères spéciaux de vos variables soient affichés avec leurs équivalents HTML, sans

que vous ne soyez obligé de vous en soucier ? (é => é, â => à, ...). Avec un moteur de template, et notamment avec Smarty, c'est possible, il vous suffit d'utiliser un "modificateur de variable" par défaut, qui se chargera de procéder à ce remplacement au bon moment. Autre exemple: Vous ne souhaitez afficher que les 30 premiers caractères du contenu de la news dans votre liste ? Avec Smarty, utilisez encore un modificateur de variable: {\$contenu|truncate:30}. En PHP, cela devient tout de suite plus long à écrire: <?php echo strlen (\$contenu) > 30 ? substr (\$contenu, 0, 30).'...' : \$contenu; ?>

Vous souhaitez afficher directement un tableau associatif dans un tableau HTML ? Avec Smarty: {html_table loop=\$table} en PHP : vous voilà obligé de prendre quelques instants pour y réfléchir.

Bien sûr, vous pouvez développer vos bibliothèques de fonctions pour prendre en charge ces balises particulières, mais alors vous vous retrouverez vous aussi à utiliser de lourdes bibliothèques de fonctions, d'autant plus que ces dernières seront sûrement déjà développées dans un moteur de template classique.

Bref, en un mot, Smarty intègre nombre d'éléments qui vous permettent d'accélérer vos développements en vous facilitant la vie.

De plus, bien que nous ne l'ayons pas vu, il est tout à fait possible avec Smarty (et avec la plupart des moteurs de templates), de développer vos propres balises. Si vous souhaitez avoir un calendrier html en réponse à la balise {monCalendrier}, c'est possible. Vous pouvez aussi définir vos propres modificateurs, vos propres fonctions de bloc et même vos propres fonctions de compilation !

Ce qu'il faut retenir de l'utilisation des templates

L'utilisation des templates va garantir à votre application son indépendance vis-à-vis de son interface. Grâce à cette indépendance, il vous sera possible de réutiliser systématiquement les parties "métiers" des fonctionnalités que vous allez développer. Si vous réalisez un système d'agenda, vous pourrez réutiliser toutes les briques métiers de ce dernier, que l'affichage en découlant se fasse en liste, en calendrier ou par export, en des formats ICalendar.

Ce que nous attendons d'un tel système, c'est de ne pas être lié à cette interface, de pouvoir la modifier quand bon nous semble sans impacts, et de pouvoir réutiliser le "métier" développé.

Maintenant, que vous optiez ou non pour un "moteur" de templates, et si c'est le cas, que vous choisissiez Smarty ou un autre, c'est une question "technique", qui n'entame en rien l'objectif à atteindre.

A suivre. (Configurer et étendre les fonctionnalités de Smarty, Développer avec des objets métier, Compléter notre framework d'après les trois notions abordées, Exemple du framework Copix).

■ Gérald Croes

www.programmez.com

**La source
de vos sources**

Créée en 1990, Aston propose des prestations de conseil, assistance à maîtrise d'ouvrage, réalisation d'applications au forfait, en assistance technique, tierce maintenance applicative et formation. Ses domaines de prédilection sont : le développement, l'intégration, le workflow, le décisionnel, les portails et Internet. Les résultats de sa cellule de veille technologique font l'objet de publications régulières dans la presse. Par ailleurs, Aston organise périodiquement des séminaires sur des sujets techniques d'actualité.

Renseignements et inscriptions : www.aston.fr



Java et les sockets

Rien de plus amusant que la programmation réseau, domaine sur lequel les sockets règnent en maîtres. Nous allons apprendre à maîtriser ceux-ci en Java, avec un petit programme de chat à la clé.

Aujourd'hui, où l'Internet est omniprésent, nous oublions facilement que faire communiquer des processus n'est pas une mince affaire. Si les deux processus ne résident pas sur une même machine, les choses se compliquent notablement. Pour solutionner ce problème, l'université de Berkeley, au début des années 80, a doté son système UNIX d'un outil de communication appelé socket, ou prise, en bon français. Prise qui permet de brancher l'un avec l'autre des processus, localement ou à distance. Le succès des sockets a été fulgurant et ceux-ci sont maintenant omniprésents. Ainsi votre navigateur Web fait un usage intensif des sockets, RMI, RPC, CORBA ou DCOM sont construits sur les sockets, le système X-Windows de votre Linux les utilise aussi. Implémentés par tous les systèmes, même Windows ;-), les sockets sont devenus de facto un standard dans le monde de l'informatique. Rappeler l'origine UNIX des sockets a son importance car, sous UNIX, tout est fichier et les sockets n'échappent pas à la règle comme nous le verrons. Au fil de cet article nous allons nous amuser avec les sockets mais sans nous embarrasser de considérations techniques de bas niveau. Nous choisissons pour cela le langage Java qui masque les détails fastidieux. Python aurait également constitué un choix excellent.

Etablir une connexion

Pour le programmeur un socket c'est trois éléments :

- une adresse IP
- un port
- un fichier ouvert en lecture/écriture

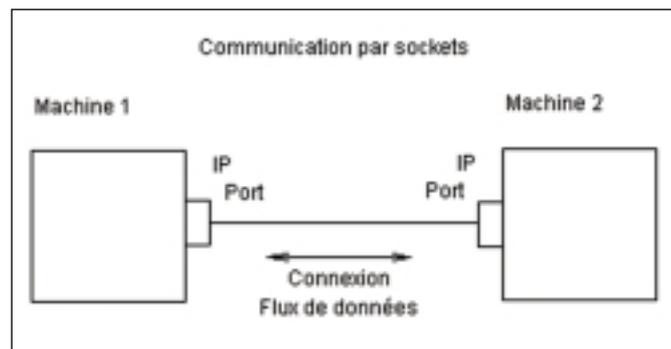


Figure 1: Anatomie d'une connexion par sockets.

Les deux premiers éléments servent à établir la connexion (figure 1), le troisième élément est automatiquement obtenu lorsque la connexion est établie.

L'adresse IP est attribuée à une machine, soit statiquement par l'administrateur d'un réseau Intranet, soit dynamiquement par un fournisseur d'accès à Internet. Il s'agit d'un nombre 32 bits représenté par une chaîne à points. Par exemple sur mon réseau Intranet une de mes machines s'est vue attribuée par mes soins l'adresse IP 192.168.10.2. En outre,

cette machine comme toutes les autres machines du monde, y compris la vôtre, se voit attribuer l'adresse 127.0.0.1, adresse qui est utilisée lorsqu'il s'agit de brancher deux processus en local, c'est à dire tournant sur la même machine. Les ports, quant à eux, n'ont rien de physique, mais constituent une abstraction qui permet d'avoir plusieurs connexions sur une même adresse. Les valeurs possibles vont de 0 à 65535. 0 est une valeur très particulière, les valeurs inférieures à 1024 sont en principe réservées à des services système et les valeurs à partir de 1024 sont à notre disposition. Supposons qu'une seule machine offre à la fois un service http et un service ftp. Les sockets pour http travailleront sur le port 80 et les sockets pour ftp travailleront sur le port 21. Ainsi pour une seule adresse IP nous pouvons établir deux connexions différentes et indépendantes avec une même machine.

Résolution d'adresses IP

En pratique, personne n'utilise les adresses IP parce qu'elles sont humainement difficiles à mémoriser d'une part, et qu'elles sont d'autre part susceptibles de changer, surtout sur Internet où les adresses sont souvent attribuées dynamiquement. Au lieu de cela on utilise des noms. Le système a alors à charge de faire correspondre les noms avec les adresses IP, soit en utilisant le contenu du fichier système hosts, soit en utilisant un service à cet effet, un DNS par exemple. Voici un extrait du fichier hosts d'un de mes Linux :

```
192.168.10.1  soleil.local  soleil
192.168.10.2  lune.local   lune
# etc, etc.
```

J'ai baptisé ma machine à l'adresse 192.168.10.2 du nom de lune. Et j'utiliserai de préférence ce nom pour établir une connexion. Dans ce cas, le système trouvera la correspondance dans le fichier hosts. Par contre, si avec mon navigateur, je veux me rendre à www.programmez.com, le système interrogera un DNS d'Internet pour connaître l'adresse IP de www.programmez.com afin d'ouvrir un socket à cette adresse sur le port 80.

Les protocoles Internet

Quand une connexion par socket est établie, la lecture de données se fait comme dans un vulgaire fichier, (en fait un tube pour être plus précis). Pour que les deux parties s'y retrouvent il convient que les données soient écrites et lues selon une procédure que l'on appelle un protocole. Lorsqu'on travaille sur son Intranet, on est libre de définir le protocole que l'on veut. Lorsqu'on qu'on travaille avec un service Internet ou même s'y l'on interroge son propre serveur Web depuis son Intranet, il convient d'observer un protocole précis. Vous pouvez trouver les protocoles Internet à <http://www.faqs.org>. Les protocoles sont décrits dans des documents dits RFC. Le protocole Daytime est décrit dans le rfc 867.

Pour nous faire la main avec les sockets, nous allons travailler avec ce protocole. Nous choisissons Daytime en raison de sa simplicité. Un service Daytime est un service rudimentaire qui, lorsqu'on établit une connexion avec lui sur le port 13, retourne brutalement une chaîne de caractères contenant le jour et l'heure, puis ferme la connexion. Daytime met à la poubelle tout octet que nous pourrions lui envoyer. A ma connaissance, le seul site qui implémente encore Daytime est le site du NIST (National Institute of Standards and Technology) à l'adresse time.nist.gov. Demandons donc l'heure qu'il est à ces braves gens !

Etablir une connexion

Nous avons le nom du site, mais pas son adresse IP. En Java nous utilisons la classe `InetAddress` et sa méthode `getByName` pour résoudre le nom, comme ceci :

```
InetAddress date_server = InetAddress.getByName("time.nist.gov");
```

Notez que lorsque cette ligne de programme est exécutée, vous devez être connecté à Internet. En effet, `time.nist.gov` ne figurant pas dans votre fichier `hosts` (le contraire serait assez étonnant...), le système va automatiquement chercher à contacter un DNS pour résoudre le nom. Lorsque le nom est résolu, nous obtenons une classe encapsulant l'adresse IP, classe que nous pouvons passer en paramètre au constructeur de la classe `Socket`, comme cela :

```
Socket s = new Socket(date_server, 13);
```

et nous pourrions commencer à travailler. Toutefois, il est mieux de gérer les choses plus finement. En effet, si personne ne "décoche" à l'autre bout du fil, le constructeur du socket va bloquer l'exécution. Il est préférable de définir un temps limite, à l'échéance duquel une exception sera levée si la connexion n'a pu être établie. Nous procédons ainsi :

```
Socket s = new Socket();
s.setSoTimeout(5000);
InetSocketAddress adresse = new InetSocketAddress(date_server, 13);
s.connect(adresse);
```

Notons au passage l'apparition d'une nouvelle classe, `InetSocketAddress`, qui encapsule à la fois l'adresse IP et le port. Nous pourrions encore employer un constructeur de `Socket` qui reçoit un timeout en argument.

Exploiter et fermer une connexion

La connexion établie, on obtient un flux via une méthode de la classe `Socket`, on lit le flux tant qu'il contient quelque chose, puis on ferme le socket. Certes nous avons dit que le socket était fermé par le serveur. Ceci n'est pas suffisant. Une connexion socket doit être fermée des deux côtés. L'encadré ci-contre (encadré 1) contient le listing complet qui a été écrit avec `JBuilder` et une `JDK 1.4.1`. Pour lancer le programme, positionnez vous à la racine de l'arborescence des classes et saisissez :

```
java -cp . programmez.fred.sockets.DateClient
```

Vous obtenez alors la date et l'heure GMT comme le montre la [figure 2](#)

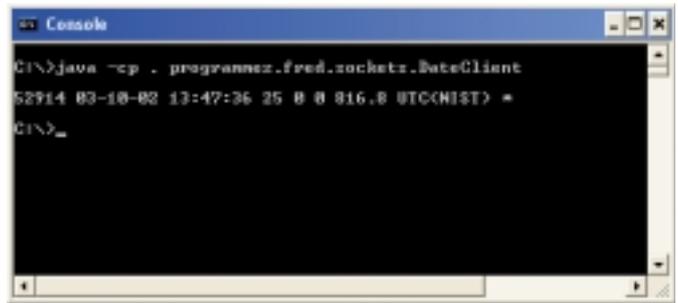


Figure 2: Interrogation d'un service Daytime.

Les exceptions

Comme on le voit dans le listing d'exemple, bon nombre d'exceptions sont capturées. Ceci est incontournable, sinon le compilateur Java vous bouterait hors du royaume de la programmation réseau. Dans un contexte réseau, quantités d'erreurs sont susceptibles de se produire. Défaillances matérielles, Hôte inconnu, etc. Le programme d'exemple capture `UnknownHostException`, qui est levée si le nom n'a pu être résolu. Ensuite nous capturons `IOException` qui sera levée si une erreur se produit lors de la lecture/écriture dans le flux. Enfin nous capturons tout le reste, sans discernement. Mais nous aurions pu, par exemple, capturer explicitement `SocketTimeoutException` qui est levée lorsqu'une connexion n'a pu être établie à l'issue du temps limite. Gérer toutes les exceptions est assez fastidieux. Alors le mieux est de prendre les choses du bon côté en utilisant les exceptions pour piloter le programme plutôt que de se limiter à émettre un message d'erreur. Vous trouverez un exemple (très simple) de ceci dans notre programme de chat.

Communication dans les deux sens

Cette fois nous allons travailler avec le protocole `http` (rfc 2616). Qui dit protocole `http` dit port 80. Il s'agit d'obtenir la page Web de www.programmez.com. Lorsqu'on ouvre une connexion avec un serveur Web, celui-ci attend patiemment, dans la limite d'un temps limite, par lui défini, que nous émettions une requête `GET` définissant quelle page nous voulons consulter. Dès qu'il a reçu la requête et qu'il l'a interprétée, il retourne la page sous la forme d'un fichier texte. Nous n'avons plus qu'à en lire toutes les lignes et afficher le résultat. Nous affichons le texte dans une console, car nous n'avons pas aujourd'hui le temps de développer un navigateur Web complet ;-). En outre, pour ce faire, nous utiliserions sans doute des classes de plus haut niveau. Mais au final, la JVM ferait la même chose que ce que nous faisons dans notre exemple. La différence par rapport à notre premier programme est que nous envoyons la requête `GET` sous la forme d'une chaîne de caractères dans le flux avant de lire le résultat, comme ceci :

```
DataOutputStream out =
new DataOutputStream(s.getOutputStream());
out.writeBytes("GET / HTTP/1.0\r\n\r\n");
```

Pour le reste, le code, que vous trouverez sur le Cd-Rom accompagnant la revue, est identique au premier exemple.

Un petit programme de chat

Jusqu'ici nous n'avons établi de connexions que du côté client. Voyons

maintenant comment les choses se passent à l'autre bout du fil, côté serveur. Pour cela, nous allons bricoler un serveur de chat rudimentaire. Un 'vrai' serveur de chat accepte une grande quantité de connexions et se charge de dispatcher les messages à tous les connectés, voire d'établir des discussions privées. Pour ne pas entrer dans des considérations qui nous éloigneraient des sockets, nous allons nous limiter. Ainsi notre serveur acceptera une seule connexion d'un seul client. Autrement dit nous établissons une seule discussion privée entre le serveur et le client, l'un et l'autre disposant d'une interface utilisateur dotée de deux zones de texte pour émettre et recevoir les messages. La connexion est à l'initiative du client qui utilise le nom d'hôte du serveur. Vous pouvez parfaitement utiliser ce chat révolutionnaire avec un ami via Internet. Tout ce que vous avez à faire est alors de modifier le programme pour spécifier une adresse IP au lieu d'un nom d'hôte et de demander à la personne du côté serveur de vous envoyer son IP dynamique, dans un e-mail par exemple. Comment peut-on connaître une IP dynamiquement attribuée par le fournisseur d'accès à Internet en Java ? Comme ceci :

```
import java.net.*;

public class monIP
{
    public static void main(String[] args)
    {
        try {
            InetAddress adr = InetAddress.getLocalHost();
            System.out.println(adr);
        }
        catch(Exception e) {
            System.out.println(e);
        }
    }
}
```

Vous pouvez encore utiliser le Serveur et le client sur la même machine, c'est à dire localement. Dans ce cas, spécifiez simplement localhost comme nom d'hôte. La [figure 4](#) montre le programme en action. Le serveur tourne sur la machine lune sous Linux et ce qui se passe sur cette machine est visualisé à distance, via le serveur X de Cygwin dans la station Windows qui fait en outre tourner le client. Vive Linux, ses sockets et son serveur X!

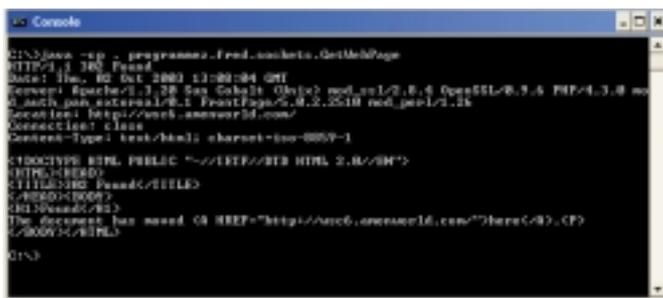


Figure 3: Et voilà. Il ne vous reste plus qu'à écrire en Java le navigateur qui permettra de visualiser cette page ;)

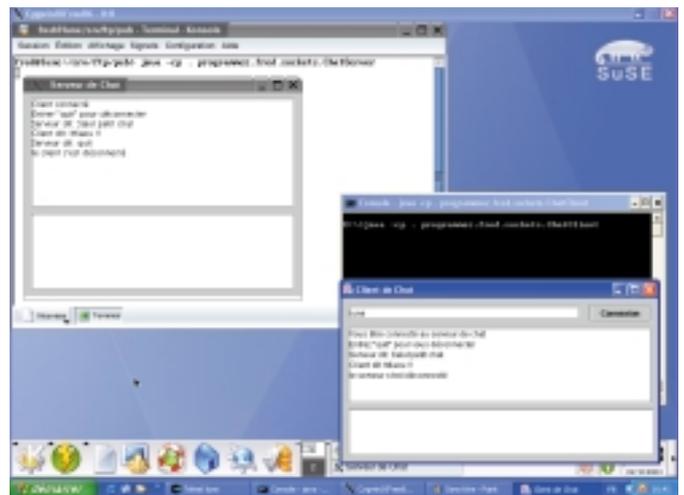


Figure 4: Chat entre deux machines sur mon Intranet, le tout visualisé sur un seul poste grâce à la magie de Cygwin, de Linux et de ses sockets :-)

Côté serveur

Nous avons implicitement vu dans les exemples précédents qu'un serveur attend pour établir une connexion. Pour cela nous utilisons la classe Java ServerSocket et sa méthode accept. Attention l'invocation de cette méthode place le thread courant en attente. Pour ne pas bloquer l'application, il est indispensable d'effectuer cette opération dans un thread séparé. Nous n'entrons pas dans le détail des threads Java aujourd'hui, car l'utilisation que nous en faisons ne présente aucune difficulté. La méthode accept rend la main, lorsqu'elle est sollicitée du côté client. En rendant la main, elle livre côté serveur un socket prêt à l'emploi. Du côté client la méthode connect a réussi et le socket est actif de ce côté aussi : la communication est établie. En principe un "écouteur" de socket boucle indéfiniment sur lui-même, ce qui est une raison de plus pour le placer dans un thread séparé. Veuillez vous reporter au programme d'exemple sur le Cd-Rom. Le reste de l'application est des plus trivial. Côté Serveur et Client, pour ne pas bloquer les interfaces, la lecture/écriture du flux est elle aussi faite dans un thread séparé. En cas de déconnexion de l'une ou l'autre partie, en cas de problème réseau ou autre, les applications sont réinitialisées lors de la capture d'exceptions. Voir ci-contre ([encadré 2](#)) la méthode readSocket du client et du serveur qui est invoquée par le thread de lecture. C'est tout simple. Et même si la gestion de l'interface utilisateur Swing mériterait d'être un chouïa améliorée, nous pouvons apprécier la magie de la programmation réseau. Soyez à l'aise dans vos sockets ! A bientôt.

Encadré 1

```
// Exemple de connexion à un service Daytime
```

```
package programmez.fred.sockets;
```

```
import java.io.*;
import java.net.*;
```

```
public class DateClient {
    public static void main(String[] args) {
```

```

try {

    InetAddress date_server =
InetAddress.getByNome("time.nist.gov");

    /*
Socket s = new Socket(date_server, 13);
*/

    Socket s = new Socket();
s.setSoTimeout(5000);
InetSocketAddress adresse =
new InetSocketAddress(date_server, 13);
s.connect(adresse);

    BufferedReader in = new BufferedReader(
        new InputStreamReader(s.getInputStream()));

    String responseLine;
while ((responseLine = in.readLine()) != null) {
    System.out.println(responseLine);
}

    s.close();
}
catch(UnknownHostException uex) {
    uex.printStackTrace();
}
}

```

```

catch (IOException e) {
    System.out.println("Erreur" + e);
}
catch(Exception ex) {
    ex.printStackTrace();
}
} // fin main
}

```

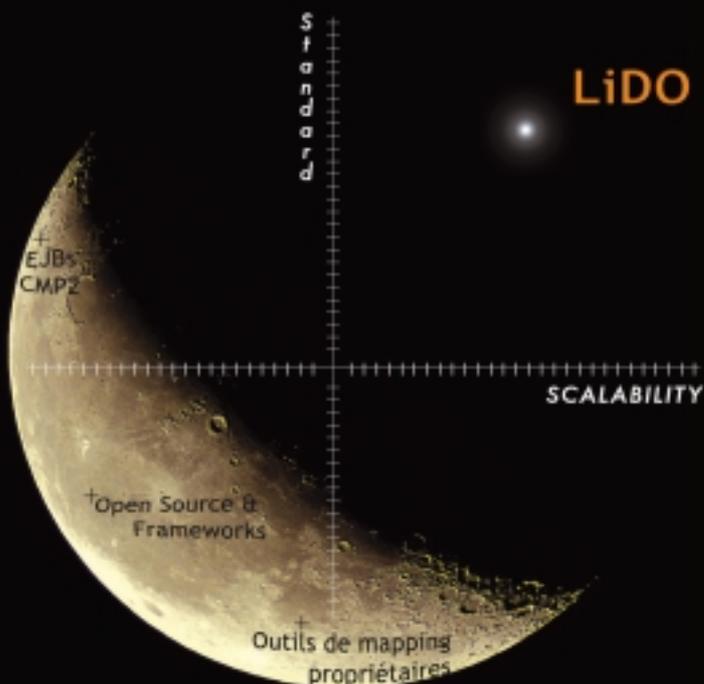
Encadré 2

```

public void readSocket() {
    try {
        String str = in.readLine();
        if (str == null) {
            reset();
        }
        else {
            if(str.equals("Client dit: quit")) {
                reset();
                return;
            }
            JTextArea1.append(str + "\n");
        }
    }
    catch(Exception e) {
        reset();
    }
}

```

■ Frédéric Mazué - fmazue@programmez.com



Réveillez-vous !
 Boostez vos applications Java
 en utilisant JDO,
 le standard pour
 l'Accès aux données d'Entreprise





Découvrir les attributs de C#

C# s'inspire de nombreux langages. Il introduit aussi quelques nouveautés originales. Parmi elles, les attributs méritent d'être découverts. Petite visite guidée.

Les détracteurs de C# lui reprochent d'être une copie de Java. La réalité est différente. L'ingénieur en chef du projet C# est Anders Hejlsberg. Anders n'en est pas à son premier langage, puisqu'il est l'auteur du très fameux Turbo Pascal qui a fait la réputation de Borland dans le domaine du développement rapide. Il a été ensuite le chef de l'équipe qui a développé Delphi, autre produit phare de Borland. De fait C# emprunte à Delphi. Les propriétés notamment. Anders déclare s'être inspiré de nombreux langages lors de la conception de C#: Modula 2, Smalltalk, C, C++ et aussi ... Java. Il n'est pas pour nous question de nier que Microsoft entend bien concurrencer Sun Microsystems sur son terrain et c'est un fait certain que C# et Java ont des points communs. C'est également un fait que C# apporte des innovations et nous allons nous intéresser aujourd'hui à l'une d'elle.

Les attributs

Parmi d'autres fléaux de la programmation, la documentation, la maintenance et les méta-données sont une source d'ennuis permanents. Le problème de la documentation est qu'il faut la maintenir à jour, au rythme des modifications dans le code. Le problème des méta-données est qu'elles sont généralement séparées du code, ce qui complique la maintenance et la construction de distributions. L'idée des attributs de C# est de fournir un moyen pour que les classes soient auto-documentées. Ainsi, les attributs permettent d'intégrer au code des informations de documentation, des informations de type méta-données, par exemple une clé de registre et sa valeur, ou encore des informations sur la capacité à faire quelque chose. La maintenance du code est simplifiée puisque tout se trouve dans un seul fichier et surtout ces informations peuvent être lues à l'exécution, au moyen du mécanisme de réflexion. C# propose un grand nombre d'attributs par défaut (figure 1) mais permet aussi et surtout de créer des attributs personnalisés, ce que nous allons faire.

Un attribut personnalisé

Il y a peu de temps, nous avons écrit ensemble un petit éditeur en C#. Supposons que notre éditeur stocke le nom du dernier document ouvert ou encore les préférences utilisateurs dans la base de registres. Un développeur peut souhaiter intégrer notre composant dans une application plus grande. Il a alors besoin de savoir quels sont les noms des clés de registre utilisées. Grâce aux attributs, notre classe elle-même peut livrer les informations requises. Nous avons codé un petit attribut comme vous pouvez le voir.

Exemple de déclaration d'attribut personnalisé

```
using System;

namespace MonEditeur
{
    public enum Hive
```

```
{
    HKEY_CLASSES_ROOT,
    HEY_CURRENT_USER,
    HKEY_LOCAL_MACHINE,
    HKEY_USERS,
    HKEY_CURRENT_CONFIG
}

public class CleRegistreAttribute : Attribute
{
    // Constructeur de l'attribut
    public CleRegistreAttribute(Hive h, String cle)
    {
        Hive = h;
        Cle = cle;
    }

    protected Hive hive;
    public Hive Hive
    {
        get { return hive; }
        set { hive = value; }
    }

    protected String cle;
    public String Cle
    {
        get { return cle; }
        set { cle = value; }
    }
}
}
```

Par souci de concision, l'attribut ne s'occupe de déclarer qu'une clé. Nous ne nous préoccupons pas de la valeur. Regardons ce code. Nous travaillons dans un espace de nom. Dans cet espace de nom nous déclarons une énumération des ruches de la base de registres. (Ceci pour l'exemple uniquement. Dans une application réelle cette énumération serait sans doute mieux placée ailleurs). Ensuite nous déclarons une classe qui dérive de System.Attribute. Nous voyons que cette classe ne diffère en rien d'une classe C# 'normale' et c'est ce qui fait toute la puissance des attributs. Ceux-ci sont certes une déclaration, mais pas une déclaration inerte. Un attribut peut exécuter du code, ce qui ouvre des possibilités à peu près infinies ou permet de résoudre des problèmes épineux. Ainsi c'est en s'appuyant sur les attributs que l'on peut appeler des APIs Windows depuis C# ou s'interfacer avec COM. Nous pouvons encore remarquer que si notre classe dérive de System.Attribute, ceci n'est en rien obligatoire. Enfin notre classe comporte deux propriétés qui sont initialisées depuis un constructeur.

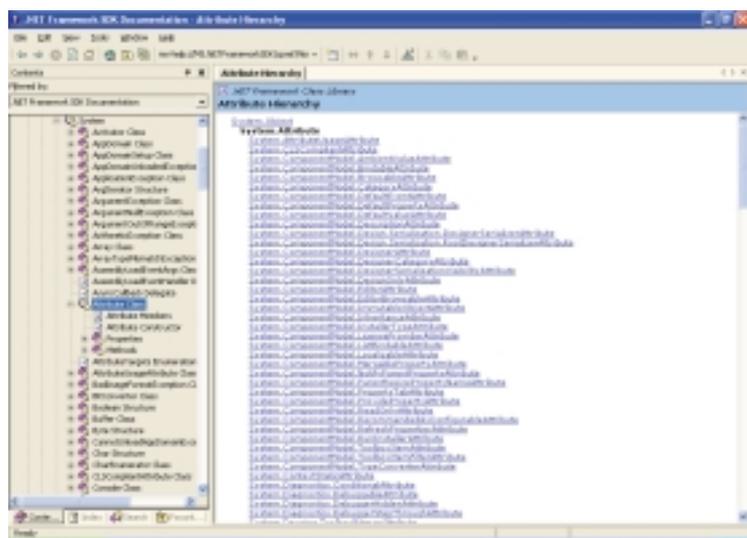


Figure 1: C# dispose d'une ribambelle d'attributs documentés dans le Framework SDK de .NET.

Utiliser un attribut

C'est tout simple. En se plaçant dans le code immédiatement au dessus de l'élément que l'on veut doter d'un attribut, on invoque le constructeur de celui-ci entre des crochets (voir ci-contre) (encadré 2). Du moins, à un détail près dans notre exemple. En effet le nom de notre classe attribut 'CleRegistreAttribute' se termine par le suffixe 'Attribute', alors que le suffixe en question n'apparaît pas dans l'invocation du constructeur. Ceci est un sucre syntaxique de C# qui est permis, si et seulement si, le nom de la classe attribut est doté du suffixe 'Attribute'. Sinon le constructeur sera énoncé entier. Bien entendu il est toujours possible d'énoncer en entier le constructeur lorsque le suffixe est présent. Ainsi :

```
[CleRegistre(Hive.HKEY_LOCAL_MACHINE, "MonEditeur\\Dernier Document")]
```

eût été parfaitement correct dans notre exemple.

Lire un attribut

Grâce à notre attribut, il est dit que notre composant travaille avec la clé " MonEditeur\\Dernier Document " sous HKEY_LOCAL_MACHINE pour sauvegarder le nom du dernier document édité. Ceci posé, comment accéder à l'information ? Nous ne voulons plus lire une documentation erronée. Au contraire, nous allons interroger à l'exécution notre composant afin de connaître la clé. Ainsi, même si le code de ComposantEdition a été modifié derrière notre dos, sans que nous en soyons informé, notre code à nous continuera de se comporter impeccablement. Pour réaliser cela nous utilisons le mécanisme de réflexion de C#.

Un peu de réflexion

Le code ci-contre (encadré 3) se comprend aisément, même si l'on a pas l'habitude de travailler avec des langages réflexifs. Tout ce qui concerne la réflexion est encapsulé dans une classe Type. Classe dont le rôle est, on s'en doute, de décrire tout Type C#. Pour obtenir une telle classe à partir de notre classe ComposantEdition, tout ce que nous avons à faire est d'employer l'opérateur typeof. Une instance de Type obtenue,

il ne nous reste plus qu'à parcourir le tableau d'attributs retourné par GetCustomAttributes. Dans ce parcours nous devons identifier l'attribut qui nous intéresse. Pour cela nous employons l'opérateur as qui se comporte comme un opérateur de transtypage, à cette différence près qu'il retourne null s'il n'y a pas concordance de type, alors qu'un vrai transtypage lèverait une exception. Dès que as retourne une valeur non null, nous sommes sur notre attribut dont nous affichons les propriétés sur une console (figure 2). Une dernière remarque : dans le code d'exemple que vous trouverez sur le Cd-Rom accompagnant la revue, la classe UtilisationMonEditeur se trouve dans le même espace de noms que ComposantEdition. Il est certain que ceci n'a pas beaucoup de sens en situation réelle. Fort heureusement, le mécanisme de réflexion continue de fonctionner si la classe qui interroge se trouve dans un autre espace de nom.

On modifierait alors simplement le code en :

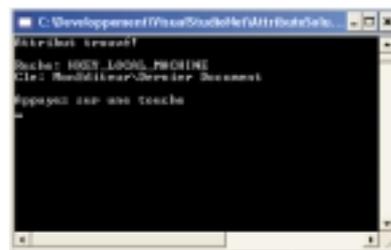


Figure 2: Et voilà, nous savons, retrouver l'attribut de notre classe à l'exécution.

```
using MonEditeur
```

```
public class UtilisationMonEditeur
{
    static void Main(string[] args)
    {
        Type type = typeof(ComposantEdition);
        // etc
    }
}
```

ou

```
public class UtilisationMonEditeur
{
    static void Main(string[] args)
    {
        Type type = typeof(MonEditeur.ComposantEdition);
        // etc
    }
}
```

Un attribut de méthode

Nous connaissons la clé utilisée par notre composant logiciel. Fort bien, mais le composant logiciel a-t-il les capacités d'écrire lui-même dans le registre ou bien compte-t-il sur un composant tiers ? Par convention nous dirons que la réponse est positive si la classe dispose d'une méthode d'attribut AutoEnregistrable et négative dans le cas contraire. Commençons par déclarer la classe d'attribut, qui cette fois, n'est rien d'autre qu'une coquille vide :

```
public class AutoEnregistrableAttribute : Attribute
{
    public AutoEnregistrableAttribute() {}
}
```

Nous ajoutons ensuite à notre classe une méthode dotée d'un tel attri-

but. Attention, la méthode doit être 'public' sinon elle ne sera pas vue lors de l'inspection.

```
public class ComposantEdition
{
    [AutoEnregistrable]
    public void ecrit_registre() {}
    // etc, etc....
}
```

Voici maintenant le code pour détecter la présence de l'attribut à l'exécution.

```
foreach(MethodInfo method in type.GetMethods())
{
    foreach(Attribute attr in method.GetCustomAttributes(true))
    {
        if(attr is AutoEnregistrableAttribute)
        {
            Console.WriteLine("\nComposantEditeur
sait écrire dans le registre");
            Console.WriteLine("Au moyen de la méthode
{0}", method);
        }
    }
}
```

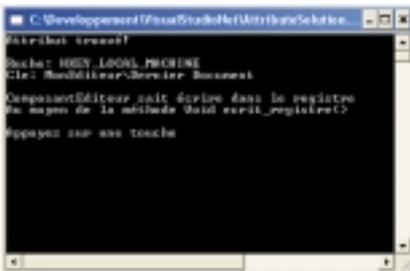


Figure 3: Lecture d'un attribut de méthode.

Nous avons cette fois deux boucles imbriquées, une pour parcourir les méthodes de la classe, une pour parcourir les attributs de chacune de ces méthodes. A l'issue de quoi nous obtenons notre information (figure 3). Nous ferions la même chose si

nous étudions les attributs des membres de la classe, mais en invoquant cette fois GetField.

Et plus encore.

Les attributs sont loin de nous avoir livré tous leurs secrets. C'est avec profit que vous étudierez la documentation de la SDK. Vous découvrirez l'utilisation des attributs par défaut, vous découvrirez encore qu'il est possible de limiter l'usage d'un attribut en dotant celui-ci d'un attribut. Si nous voulons que notre attribut AutoEnregistrable ne puisse être utilisé que pour annoter les méthodes et jamais les classes ni les membres, nous procéderons ainsi :

```
[AttributeUsage(AttributeTargets.Method)]
public class AutoEnregistrableAttribute : Attribute
{
    public AutoEnregistrableAttribute() {}
}
```

Il est également possible de donner des valeurs par défaut aux paramètres des constructeurs, d'hériter ou même de surcharger des attributs. Enfin, nous l'avons dit au début de cet article, les attributs ont un rôle à jouer pour les appels système ou l'interface avec COM, fonctionnalité que nous étudierons ensemble une prochaine fois.

Encadré 2

Utilisation d'un attribut

```
using System;

namespace MonEditeur
{
    [CleRegistre(Hive.HKEY_LOCAL_MACHINE, "MonEditeur\\Dernier
Document")]
    public class ComposantEdition
    {
        public static void Main ()
        {
            System.Console.WriteLine(
                "Ce composant est rudimentaire n'est ce pas ?");
        }
    }
}
```

Encadré 3

Lecture des attributs d'une classe au moyen du mécanisme de réflexion de C#.

```
public class UtilisationMonEditeur
{
    static void Main(string[] args)
    {
        Type type = typeof(ComposantEdition);
        foreach (Attribute attr in type.GetCustomAttributes(true))
        {
            CleRegistreAttribute cra =
                attr as CleRegistreAttribute;
            if(cra != null)
            {
                Console.WriteLine("Attribut trouvé!\n");
                Console.WriteLine("Ruche: {0}", cra.Hive);
                Console.WriteLine("Cle: {0}", cra.Cle);
            }
        }
        Console.WriteLine("\nAppuyez sur une touche");
        Console.ReadLine();
    }
}
```

■ Frédéric Mazué - fmazue@programmez.com

PHP : au-delà du Web

PHP a été créé pour le Web et est utilisé essentiellement dans ce domaine. Cependant, en bousculant un peu les idées reçues, un programmeur peut aussi l'employer comme langage généraliste.

Rasmus Lerdorf en créant en 1995 le 'Personnal Home Page', ne pensait sans doute pas qu'un jour PHP pourrait servir à autre chose que le web... En tant que programmeur, le but serait d'écrire un code en PHP qui s'exécute comme n'importe quel autre script. C'est maintenant une réalité avec PEAR et PHP-GTK.

Le code source que nous allons créer ensemble s'exécutera indifféremment sous Linux ou Windows. Sous Linux Debian, il suffit de lancer un apt-get pour installer PHP4 (voir plus loin pour l'installation sous Windows). Voici au final ce qui est installé :

```
#dpkg -l | grep php
ii libphp-adodb 1.51-1 The 'adodb' database abstraction layer for p
ii php-gtk 0.5.0-3 PHP extension for GTK+ client-side cross-pla
ii php4 4.1.2-6woody3 A server-side, HTML-embedded scripting langu
ii php4-cgi 4.1.2-6woody3 A server-side, HTML-embedded scripting langu
ii php4-pear 4.1.2-6woody3 PEAR - PHP Extension and Application Reposit
```

Transformons PHP en langage de script

Après l'installation de PHP4, nous avons créé un lien symbolique de /usr/bin/php4 vers /usr/bin/php (commande ln). Maintenant si nous exécutons le script bj.php suivant :

```
<?php
echo "Hello World\n";
?>
```

Par le biais de l'instruction " php bj.php", nous obtenons la réponse suivante :

```
X-Powered-By: PHP/4.1.2
Content-type: text/html

Hello World
```

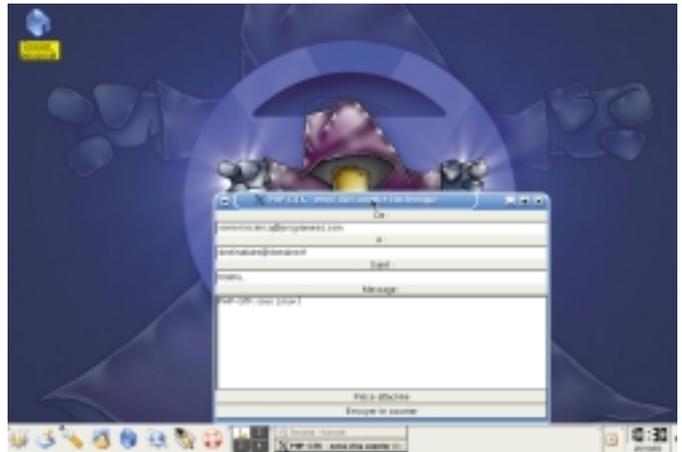
Ce qui est assez gênant... L'option "-q" placée en ligne de commande permet d'éviter d'afficher les en-têtes de réponse http. Transformons notre code source qui devient :

```
#!/usr/bin/php -q
<?php
echo "Hello World\n";
?>
```

Et rendons-le exécutable par la commande "chmod +x bj.php". Nous pouvons maintenant invoquer directement le script en tapant "./bj.php". Et l'interpréteur ne renvoie plus qu'une seule ligne : "Hello World".

L'extension Pear

PEAR signifie "PHP Extension And application Repository". Il s'agit d'une collection pour PHP de classes en Open Source.



Exécution sous Linux

PEAR s'installe en pointant son navigateur à l'adresse go-pear.org. Sous Linux cela donne :

```
#lynx -source http://go-pear.org | php -q
```

Après l'installation, vous vous retrouvez avec la commande "pear". Celle-ci permet d'installer une nouvelle extension, de lister les extensions déjà installées, de lister l'ensemble des packages existants que l'on peut installer, etc.

```
# pear list
Installed packages:
=====
Package      Version  State
Archive_Tar  1.1      stable
Console_Getopt 1.0      stable
DB            1.5.0RC2 stable
Mail          1.1.2    stable
Mail_Mime    1.2.1    stable
Net_SMTP     1.2.3    stable
Net_Socket   1.0.1    stable
PEAR         1.3b2    beta
PHPUnit      1.0.0-alpha2 alpha
XML_Parser   1.0.1    stable
XML_RPC      1.0.4    stable
```

Nous allons utiliser les extensions Mail et Mail_Mime avec notre application. Celle-ci consiste à afficher une boîte de dialogue graphique d'envoi d'un courrier électronique. En utilisant "Mail_Mime" nous pourrions y attacher une pièce jointe. Mais comment afficher graphiquement une boîte de saisie ou un autre objet ? C'est ici que GTK+ intervient.

Le monde graphique avec GTK+

Nous avons installé PHP-GTK sous Linux et Windows. Si vous êtes sous

Windows vous pouvez télécharger les binaires à l'adresse <http://gtk.php.net/download.php>. Il suffit de décompresser le fichier zip (de moins de 3 Mo) et de suivre les instructions du fichier README.TXT pour recopier les répertoires aux endroits adéquats selon votre version de Windows (<http://gtk.php.net/manual/fr/installwin32.php>). En fait tel quel, en ligne de commande, php-gtk est déjà fonctionnel.

Notre première ligne de code fait appel à l'interpréteur :

```
#!/usr/bin/php -q
```

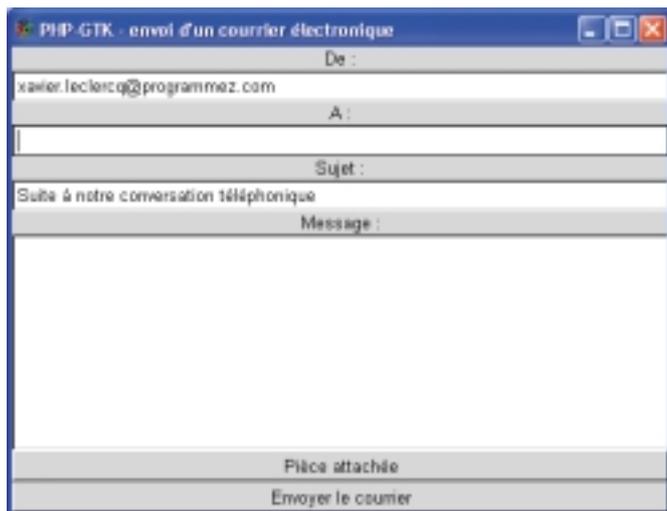
Sous Windows, vous devez indiquer le chemin à l'interpréteur php-win ou ne rien mettre si vous tapez en ligne de commande "php-win ..\test\mel.php". Pour nous, débutons notre code PHP :

```
<?php
```

Nous rendons une variable globale. Rappelons qu'en PHP, pour que celle-ci soit accessible dans une fonction, il faut indiquer à nouveau le fait qu'elle est globale.

```
global $nomfichier;
```

Le code suivant charge l'extension PHP_GTK. Le code est adapté à Windows ou à Linux :



Exécution sous Windows XP

```
dl('php_gtk.'. (strpos(PHP_OS, 'WIN') ? 'dll' : 'so'));
```

Le code source s'exécute séquentiellement. Nous ouvrons une nouvelle fenêtre :

```
//Nouvelle instance d'une fenêtre  
$w=&new GtkWindow();
```

Et nous réagissons aux événements utilisateurs en fixant des fonctions "callbacks". Concrètement, que ce soit avec "connect_object", ou plus loin avec "connect", nous relierons un événement comme le "click" sur un bouton de commande avec un appel d'une fonction gérant l'action utilisateur. Lorsque l'utilisateur cliquera (action 'clicked') sur le bouton "Envoyer le courrier", PHP-GTK exécutera la fonction 'envoi' en lui transmettant les variables \$entry(1 à 4).

```
$button2 = &new GtkButton("Envoyer le courrier");  
$button2->connect_object('clicked', 'envoi', $entry1, $entry2, $entry3, $entry4);
```

L'interface graphique est construite en élaborant un conteneur (\$box) qui "contiendra" les autres objets (étiquettes, entrées et boutons de commandes).

```
$box = &new GtkVBox();  
$w->add($box);
```

Envoi d'un courrier électronique

Pour envoyer des messages multipart, c'est-à-dire des messages combinant du texte et du html nous utiliserons l'extension mail_mime de PEAR. Les logiciels de courriers électroniques afficheront automatiquement la portion texte ou html qui leur convient. Nous créons un objet mail_mime et les méthodes SetTXTBody() et SetHTMLBody() ajouteront les portions de textes ou HTML. Pour ajouter un fichier attaché, nous appelons la fonction addAttachment. Ensuite, il faudra récupérer les entêtes et le corps du message MIME (avec headers() et get()).

Enfin Mail_mime::send() expédiera le message. D'autres méthodes sont envisageables comme le passage par un serveur SMTP ou l'appel direct sous Unix à SendMail. La méthode que nous préconisons présente l'avantage de fonctionner sous Windows et sous Linux sans comprendre "la machinerie qui tourne derrière".

Pour conclure

A l'adresse <http://www.zend.com/zend/tut/tutorial-silva.php> vous découvrirez un tutorial (en anglais) qui décortique chaque fonction PHP-GTK. PHP est très facile d'apprentissage, et il s'avère extrêmement productif. Si les applications temps réel ou de calculs intensifs devront toujours s'écrire en C/C++, rien n'empêche PHP de jouer sur les autres terrains de la programmation.

■ **Xavier.Leclercq** - Xavier.Leclercq@programmez.com

Récapitulons l'ensemble du code :

```
#!/usr/bin/php -q  
<?php  
global $nomfichier;  
dl('php_gtk.'. (strpos(PHP_OS, 'WIN') ? 'dll' : 'so'));  
  
function get_input($entry) {}  
  
function envoi($entry1, $entry2, $entry3, $entry4) {  
    global $nomfichier;  
  
    $De=$entry1->get_text();  
    echo "De: $De\n";  
    $en_têtes['From'] = $De;  
  
    $A=$entry2->get_text();  
    echo "A: $A\n";  
  
    $Sujet=$entry3->get_text();  
    echo "Sujet: $Sujet\n";  
    $en_têtes['Subject'] = $Sujet;  
  
    $corps=$entry4->get_chars(0,-1);  
    echo "Corps: $corps\n";  
  
    echo "$nomfichier\n";  
  
    require 'Mail.php';  
    require 'Mail/mime.php';
```

```

$text = $corps;
$html = '<html><body>' + $corps + '</body></html>';

$crf = "\r\n";
$headers = array(
    'From' => $De,
    'Subject' => $Sujet
);

$mail = new Mail_mime($crf);

$mail->setTXTBody($text);
$mail->setHTMLBody($html);

if (empty($nomfichier)) {
}
else {
    $mail->addAttachment($nomfichier, 'text/plain');
}

$body = $mail->get();
$headers = $mail->headers($headers);

$mail =& Mail::factory('mail');
$mail->send($A, $headers, $body);

Gtk::main_quit();
}

function fichier() {
    global $windows;
    if (!isset($windows['file_selection'])) {
        $window = &new GtkFileSelection('File selection dialog');
        $windows['file_selection'] = $window;
        $window->hide_fileop_buttons();
        $window->set_position(GTK_WIN_POS_MOUSE);
        $window->connect('delete_event', 'delete_event');

        $button_ok = $window->ok_button;
        $button_ok->connect('clicked', 'file_selection_ok', $window);

        $button_cancel = $window->cancel_button;
        $button_cancel->connect('clicked', 'close_window');
    }
    if ($windows['file_selection']->flags() & GTK_VISIBLE)
        $windows['file_selection']->hide();
    else
        $windows['file_selection']->show();
}

function file_selection_ok($button, $window) {
    global $nomfichier;
    $nomfichier = $window->get_filename();
    //echo "$nomfichier\n";
    close_window($window);
}

function close_window($widget) {
    $window = $widget->get_toplevel();
    Gtk::grab_remove($window);
    $window->hide();
}

//Nouvelle instance d'une fenêtre
$w=&new GtkWindow();

```

```

//Connection pour pouvoir quitter proprement
$w->connect_object('destroy', array("gtk", "main_quit"));

//Titre de notre fenêtre
$w->set_title("PHP-GTK - envoi d'un courrier électronique");
//Dimension
$w->set_default_size(500,150);
//Position centrée
$w->set_position(GTK_WIN_POS_CENTER);

$box = &new GtkVBox();
$w->add($box);

$lbl1 = &new GtkLabel();
$lbl1->set_text("De :");
$box->add($lbl1);
/* GtkEntry avec callback */
$entry1 = &new GtkEntry();
$entry1->grab_focus();
$entry1->set_text("de@domaine.com");
$entry1->connect('activate', 'get_input');
$box->add($entry1);

$lbl2 = &new GtkLabel();
$lbl2->set_text("A :");
$box->add($lbl2);
$entry2 = &new GtkEntry();
$entry2->set_text("destinataire@domaine.com");
$entry2->connect('activate', 'get_input');
$box->add($entry2);

$lbl3 = &new GtkLabel();
$lbl3->set_text("Sujet :");
$box->add($lbl3);
$entry3 = &new GtkEntry();
$entry3->set_text("blabla...");
$entry3->connect('activate', 'get_input');
$box->add($entry3);

$lbl4 = &new GtkLabel();
$lbl4->set_text("Message :");
$box->add($lbl4);
$entry4 = &new GtkText();
$entry4->set_editable(TRUE);
$entry4->connect('activate', 'get_input');
$box->add($entry4);

$button1 = &new GtkButton("Pièce attachée");
$button1->connect_object('clicked', 'fichier');

$box->add($button1);

$button2 = &new GtkButton("Envoyer le courrier");
$button2->connect_object('clicked', 'envoi', $entry1, $entry2, $entry3, $entry4);
$box->add($button2);

$box->show_all();

//Affiche tout
$w->show_all();

//gtk main
Gtk::main();
?>

```

PAR LAURENT JAYR - DEV.NET@SKEDDIO.COM - TOUS DROITS RÉSERVÉS

Programmation d'un jeu 3D

La sauvegarde des scores



5^{ème} partie - et fin

Voici la conclusion de ce long dossier sur la conception d'un jeu 3D pour CD-Rom et pour Internet. La dernière technique que nous vous présentons vous permettra de créer un module de sauvegarde des meilleurs scores en ligne. Nous utiliserons pour ceci des scripts CGI en langage Perl, langage bien représenté sur la majorité des serveurs.

Sauvegarder les meilleurs scores en ligne

Notre programme doit être capable de sauvegarder en ligne des dix meilleurs scores. Cette fonctionnalité est décrite dans le dossier de spécifications de l'étude de cas.

Il n'existe pas beaucoup de solutions techniques autres que les scripts CGI pour assurer la sauvegarde en ligne des scores. En effet, cette application peut tourner dans un navigateur Internet et il demeure donc impossible d'accéder au disque dur de l'ordinateur de l'utilisateur, pour des raisons de sécurité évidentes. C'est pourquoi nous préconisons l'utilisation de scripts CGI développés en langage Perl.

DEFINITION : les scripts CGI

L'interface qui définit la façon dont les informations sont transmises du navigateur au serveur s'appelle la Common Gateway Interface (interface de passerelle commune), ou CGI. Le programme qui accepte ce type d'informations, et que l'on appelle généralement un programme ou script CGI, les traite et fait appel au protocole HTTP pour renvoyer des commandes ou des documents dynamiques au navigateur.

Méthode utilisée pour écrire et lire des données sur le serveur

Notre objectif est donc de manipuler un fichier de type texte ASCII dans lequel seront enregistrés les noms et scores associés de tous les joueurs qui accéderont à l'application en ligne et qui saisiront leur nom.

En effet, à la fin de chaque course, notre application affichera le chronomètre final consécutif aux trois tours de circuit, ainsi qu'un champ de saisie du nom du joueur, et ce, quel que soit le temps effectué (fig 1).

En cliquant sur le bouton SAVE, le joueur pourra donc sauvegarder le nom saisi et son temps dans le fichier de données, prévu à cet effet sur le serveur.



Figure n°1 - Affichage des meilleurs scores et des noms.

Nous utiliserons trois fichiers par circuit pour cette opération. Par exemple, pour le premier circuit, nous utiliserons le fichier texte scores1.txt pour enregistrer les chronos et les noms des joueurs, le fichier script CGI read1.cgi pour lire les chronos et les noms écrits dans le fichier scores1.txt et le fichier script CGI write1.cgi pour écrire les noms et chronos des joueurs dans le fichier scores1.txt après la saisie du nom.

L'utilisation d'un logiciel FTP

Pour placer ces trois fichiers sur le serveur supportant l'étude de cas, vous devez utiliser un logiciel FTP, comme par exemple le logiciel SmartFTP téléchargeable sur le site www.smartftp.com.

Ensuite, vous devez avoir accès au répertoire \cgi-bin du serveur Web que vous utiliserez pour la sauvegarde des scores en ligne. Dans la plupart des cas, placez vos scripts CGI dans le répertoire \cgi-bin, car c'est généralement l'emplacement réservé à l'exécution des scripts CGI sur un serveur.

ATTENTION : les hébergements gratuits

Certains hébergeurs gratuits vous proposent un espace de plusieurs Mo pour votre site Internet. Cependant, il est fort probable que ces derniers ne vous permettent pas d'accéder au répertoire \cgi-bin pour y placer vos scripts personnels. Vous ne pourrez donc pas intégrer la procédure de sauvegarde en ligne des

meilleurs scores.

La figure f montre le répertoire cgi-bin ouvert avec SmartFTP. Tous les fichiers scripts relatifs aux trois circuits et les fichiers textes enregistrant les scores sont stockés dans ce répertoire.

Placer les scripts CGI dans le répertoire cgi-bin ne suffit pas en soi. Il faut aussi modifier les attributs de ces fichiers. Par exemple, il faut que le script read1.cgi puisse être lu, écrit et surtout exécuté, car c'est le principal rôle d'un script CGI. Cette opération peut être exécutée avec la fonction CHMOD de votre logiciel FTP. Avec SmartFTP, sélectionnez le fichier à "

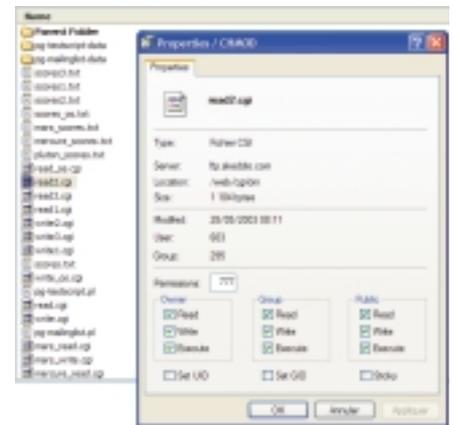


Figure f - Répertoire cgi-bin du serveur

CHMODER " et cliquez sur la touche F7 (figure f) pour paramétrer les propriétés du fichier en lecture, écriture et exécution.

Utilisez cette même commande pour paramétrer vos fichiers textes en lecture et écriture car ces derniers devront être lus et écrits sans restriction.

Analyse des scripts CGI en langage Perl

Vous connaissez le mode opératoire pour placer les fichiers scripts CGI, les fichiers textes de scores et comment les paramétrer sur un serveur avec un logiciel de transfert FTP. A présent, nous allons vous montrer comment nos

scripts CGI ont été conçus pour remplir leurs tâches définies précédemment.

Nous avons choisi d'utiliser le langage Perl pour l'écriture de ces CGI. Ils auraient pu être aussi bien écrit en PHP ou en C mais nous préférons utiliser dans ce cas le Perl, car ce langage standard pour les CGI nous assure qu'il sera interprété par tous les serveurs possibles.

Le langage Perl

Situé entre le shell et le langage C et influencé par nombre des outils standard UNIX, le langage Perl de Larry Wall convient parfaitement aux manipulations textuelles, à la création de scripts CGI et aux tâches d'administration système.

Notre objectif n'est pas de présenter ici un tutorial complet sur le langage Perl. Mais nous allons vous montrer quelques caractéristiques et fonctions de ce langage appliqué à notre étude de cas.

Les programmes Perl sont nos scripts CGI. Ces fichiers textes portent l'extension *.cgi* mais pourraient aussi porter l'extension *.perl*.

L'écriture des données dans un fichier texte

Commençons par analyser le code source du programme Perl *write1.cgi* (voir le script n°1). Ce programme permet d'écrire dans le fichier texte *scores1.txt* le score et le nom du joueur, informations envoyées par notre application lors d'une procédure que nous décrivons un peu plus loin.

Script n°1

```
#!/usr/bin/perl

# Récupérer la taille des données postées :
$taille = $ENV{'CONTENT_LENGTH'};

# Récupérer les données postées :
read (STDIN, $donnees, $taille);

# Conversion de caractères :
$donnees =~ s/%([\dA-Fa-f][\dA-Fa-f])/pack("C",hex($1))/eg;
$donnees =~ tr/\+/ /;

# Créer un tableau de données avec les données postées :
@split_data = split("&", $donnees);
foreach $index (@split_data)
{
    ($index1, $data1) = split("=", $index);
    $info{$index1} = $data1;
}

```

```
# Ouvrir le fichier texte :
open(SCORES1,">>scores1.txt") || exit;

# Copier les données "score" et "nom" dans le fichier :
print SCORES1 "$info{'score'}&$info{'nom'}&\n";

# Fermer le fichier :
close(SCORES1);

# Finir l'écriture du score et du nom :
exit;

```

On observe quatre étapes séquentielles pour ce script. La première étape récupère la taille des informations envoyées par l'application (autrement dit le score et le nom). La seconde récupère les données de ces informations. La troisième étape ouvre le fichier texte *scores1.txt* en mode écriture. La quatrième étape copie les informations récupérées à la fin de ce fichier et la dernière étape ferme le fichier. Le mode opératoire de copie de données est simple et terminé. Observons à présent ce qui se passe en amont au niveau de l'application, pendant le processus d'envoi des informations.

Script n°2 :

```
global gScore - chrono final
global gRace - planète sélectionné est
- Numéro d'identification de connexion réseau
global gNumID

on beginSprite me

-convert score in frames
gScore = HMStoFrames(gScore, 30, FALSE, FALSE)

- sélectionner l'URL
- appropriée pour lancer le script CGI adéquat :
case gRace of
    1 : urlCGI = "http://www.skeddio.com/cgi-bin/write1.cgi"
    2 : urlCGI = "http://www.skeddio.com/cgi-bin/write2.cgi"
    3 : urlCGI = "http://www.skeddio.com/cgi-bin/write3.cgi"
    otherwise : nothing
end case
- Récupérer le nom du joueur saisie dans l'acteur "NomJoueur"
INomJoueur = member("NomJoueur").text
- Déterminer le nombre de caractères dans le nom saisie
nb = member("NomJoueur").text.length
if nb < 10 then
    repeat with n = nb to 9
        - Compléter son nom avec des '.'
        - pour obtenir 10 caractères :
        INomJoueur = INomJoueur & "."
    end repeat
end if

```

```
end repeat
end if
if nb > 10 then
    - Si le nom fait plus de 10 caractères, sélectionner
    - les 10 premiers seulement :
    INomJoueur = chars (INomJoueur, 1, 10)
end if

- Enregistrer toutes les informations (score + nom)
- dans une liste :
liste = ["nom":INomJoueur, "score":string(gScore)]

- Puis envoyer cette liste vers le navigateur et le
- script CGI :
gNumID = postNetText(urlCGI,liste)
end

on exitFrame
- stay on this frame until the net operation is
- completed
if netDone(gNumID) = FALSE then
    go to the frame
else
    restart1()
    go to "s" & gRace
end if
end

```

Nous retrouvons dans ce comportement quatre étapes séquentielles. La première permet de définir l'URL du script CGI utilisé pour l'écriture des données sur le serveur, en fonction du circuit sélectionné (pour chaque circuit, on possède un script CGI différent).

Ensuite, le programme récupère le nom saisie et réajuste son nombre de caractères si besoin. Ce nom est stocké dans une liste temporaire qui enregistre aussi le score du joueur et nous utilisons pour finir la commande Lingo réseau *postNetText()* pour envoyer les informations contenues par cette liste vers le script CGI que nous avons défini plus tôt.

Commande réseau *postNetText (URL, data)*

Cette commande Lingo réseau envoie une requête POST à URL, qui est une adresse URL HTTP, avec data comme données.

La lecture des données à partir un fichier texte

A présent, nous allons étudier le programme Perl utilisé pour lire les informations stockées dans le fichier texte. Le fichier contenant ce programme porte le nom de "read1.cgi" pour les scores du circuit n°1. Voici l'intégralité de son script :

Script n°3

```
#!/usr/bin/perl

# Ouvrir le fichier "scores1.txt" qui intègre
# les 10 meilleurs noms+chronos du circuit 1 :
open(SCORES1,"scores1.txt") || exit;

# Enregistrer les données de ce fichier dans la
# variable @liste_scores :
@liste_scores = <SCORES1>;

# Fermer le fichier :
close(SCORES1);

# Trier les données :
@liste_scores = sort { $a <=> $b } @liste_scores;

# Définir le nombre de scores à lire :
$scores_max = 10;
$index = 0;

# Pour chaque ligne, retourner les scores et les
# noms associés du plus petit au plus grand
# chrono :
foreach $ligne_score (@liste_scores)
{
    print $ligne_score;
    $index = $index + 1;
    if ($index >= $scores_max)
    {
        last;
    }
}

# Finir lecture des scores
exit;
```

Nous pouvons définir quatre étapes importantes dans ce script. La première étape permet d'ouvrir le fichier texte. La seconde enregistre les données du fichier texte dans une variable. La troisième étape ferme le fichier, la quatrième et dernière étape trie les données pour ne retenir que les dix meilleurs scores (les noms et les chronos les plus faibles). Ces informations sont retournées au niveau de l'application, qui peut les interpréter et les afficher dans l'application grâce au comportement décrit dans le script n°4.

Script n°4

```
- Numéro d'identification de connexion réseau
global gNumID
global tmp
global gRace

on beginSprite me
gNumID = VOID
```

```
- Sélectionner l'URL appropriée pour lancer le
script CGI adéquat :
case gRace of
    1 : urlCGI = "http://www.skeddio.com/cgi-
bin/read1.cgi"
    2 : urlCGI = "http://www.skeddio.com/cgi-
bin/read2.cgi"
    3 : urlCGI = "http://www.skeddio.com/cgi-
bin/read3.cgi"
    otherwise : nothing
end case
```

```
- Puis réceptionner les informations:
gNumID = getNetText(urlCGI)
tmp = ""
texte = ""
text = ""
end
```

```
on exitFrame me
```

```
if gNumID <> VOID then
- Vérifier la connexion :
if netDone(gNumID) then
```

```
- Récupérer le texte dans une variable :
texte = netTextResult(gNumID)
text = ""
the itemDelimiter = "&"
```

```
gNumID = VOID
```

```
- Récupérer les composantes nom et score
- des données et les placer dans une variable
repeat with n = 1 to texte.line.count - 1
    texte1 = texte.line[n].item[3]
    texte2 = integer(texte.line[n].item[1])
    texte2 = framesToHMS(texte2, 30, FALSE,
FALSE)
    put texte1&" "&texte2&RETURN after text
end repeat
```

```
- Afficher les noms et scores :
member("scores_window").text = text
member("scores_window").bottomSpacing = 5
- member("scores_window").tabs = [[#
type: #left, #position: 90]]
member("scores_window").alignment =
#left
```

```
- Récupération et affichage des données
terminé,
- passer à la frame suivante :
go to the frame + 1
end if
end if
```

```
go to the frame
end
```

Dans ce script, nous retrouvons trois quatre fondamentales. La première permet de définir l'URL du fichier script CGI présent sur le serveur, la seconde initie la connexion en mode récupération de données, la troisième récupère les données dans une variable puis la quatrième et dernière étape affiche les scores dans un acteur texte.

LINGO : Commande réseau getNetText(URL)
Cette commande démarre la récupération de texte à partir d'un fichier placé sur un serveur HTTP ou FTP ou initie une requête CGI.

Vous trouverez l'ensemble de ces fichiers sur le CD-Rom et dans le répertoire CGI. Notre jeu peut tourner dans un navigateur Internet en exportant l'application au format Shockwave. Cependant, nous n'avons pas la place de vous montrer comment exporter ce jeu à ce format, aussi nous vous proposons les codes sources en application EXE conventionnel. Un exemple d'application Shockwave pour ce jeu est disponible ici : <http://www.skeddio.com/RR3D/RallyRace3D.htm>.

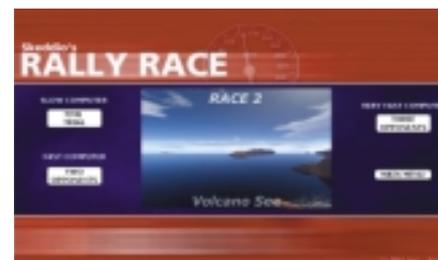
Récapitulatif du dossier

Cette dernière partie sera l'occasion pour nous de récapituler tous les sujets abordés dans ce dossier en cinq épisodes. Nous vous proposons donc une vue d'ensemble avec les thèmes techniques décrits et les références à retenir.

PARTIE 1

Programmation d'un jeu 3D Introduction

Programmez numéro 55
Juillet-Août 2003



Dans ce numéro, nous avons présenté un état des lieux sur la conception des jeux 3D en général et des jeux sur Internet en particulier.

Cette introduction nous a permis également de poser les fondations du cahier des charges pour l'étude de cas, en définissant les outils et le plan de développement du projet.

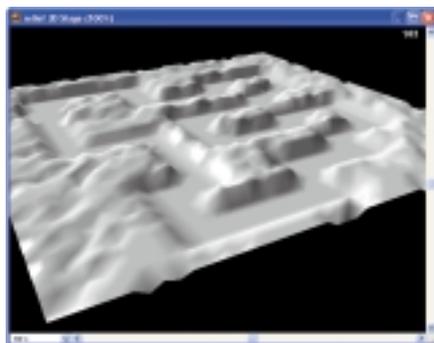
Thèmes abordés :

- La programmation des jeux 3D, vue d'ensemble
- Les spécificités des jeux 3D pour Internet
- Spécifications de l'étude de cas
- Les outils de développement
- Plan de développement du projet

PARTIE 2

Programmation d'un jeu 3D Technique de modélisation 3D

Programmez numéro 56
Septembre 2003



La seconde partie du dossier fut l'occasion de présenter une technique de modélisation 3D en temps réel pour la conception d'un relief tridimensionnel. La technique que nous utilisons ici est une technique classique mais astucieuse de l'utilisation d'image grayscale pour modéliser un relief sans outil de modélisation 3D complexe et onéreux.

Thèmes abordés :

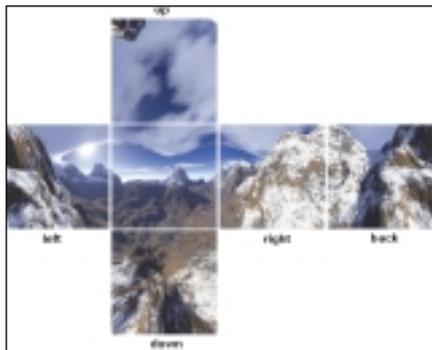
- Principe de la technique
- Analyse du code source exemple
- Utilisation d'un générateur de maille
- Concepts de représentation 3D
- La technique du mipmapping

PARTIE 3

Programmation d'un jeu 3D – Skybox et moteur de simulation physique

Programmez numéro 57
Octobre 2003

Un autre grand classique de la technique de modélisation 3D est abordé dans cette troisième partie : la conception de " Skybox ". Cette technique est essentielle pour la modélisation d'un environnement 3D fin et détaillé, comme



les paysages. Ensuite, nous avons présenté dans ce numéro le moteur de simulation physique Havok, colonne vertébrale de notre projet et qui nous permettra de simuler toute la dynamique physique du jeu.

Thèmes abordés :

- Principe d'un skybox
- Réalisation d'un skybox
- Le logiciel de génération de paysage Terragen
- Présentation du moteur de simulation physique Havok
- Utilisation du moteur Havok

PARTIE 4

Programmation d'un jeu 3D Dynamique physique et pseudo intelligence artificielle

Programmez numéro 58
Novembre 2003



Avec la partie 4, nous entrons dans le vif du sujet en démontrant les capacités du moteur Havok directement appliquées à l'étude de cas. Ensuite, nous présentons une technique simple mais astucieuse pour simuler un pseudo programme d'intelligence artificielle.

Thèmes abordés :

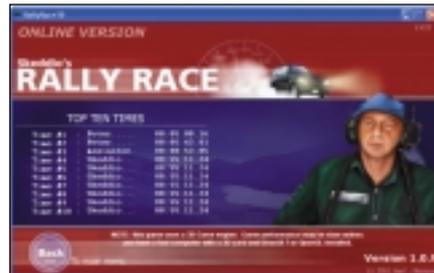
- La dynamique des véhicules avec le moteur Havok
- Pseudo IA pour les adversaires

PARTIE 5

Programmation d'un jeu 3D La sauvegarde des scores

Programmez numéro 59
Décembre 2003

La dernière partie du dossier est essentiellement tournée vers la sauvegarde en ligne des meilleurs scores via quelques scripts CGI. Nous



présentons également un récapitulatif global du projet.

Thèmes abordés :

- Sauvegarde des scores en ligne
- Récapitulatif du dossier.

Optimisations et conclusion

La réalisation d'un jeu en 3D est un développement complexe, qui fait intervenir de nombreuses compétences. Concevoir un jeu 3D même simple, qui intègre un moteur 3D, un moteur de simulation physique, une synthèse vocale en temps réel, des effets sonores, des musiques et qui fonctionne également dans un navigateur Internet peut sembler impossible au premier abord. Et pourtant, nous venons de démontrer que c'est possible, même pour une petite équipe limitée à une ou deux personnes !

Cependant, quelques améliorations sont à envisager. Notamment dans le domaine des modèles et textures 3D. Sans dépasser certaines limites dues aux caractéristiques des web games (limitation en nombre de polygones), il serait préférable d'augmenter le nombre de modèles 3D du jeu pour rendre ce dernier plus réaliste. Il serait aussi préférable d'améliorer la qualité de la texture de la piste. Pour finir avec les optimisations possibles, il serait bien d'affiner le comportement dynamique du véhicule dans les phases de collisions et dans son contrôle (avec plus de dérapages et de glissades par exemple). Nous espérons que cette étude de cas vous donnera les fondamentaux pour la conception de vos propres jeux 3D. Le marché est relativement fermé dans ce domaine pour une petite équipe débutante. Cependant, il reste très ouvert dans le domaine des web games 3D dédiés à Internet. A vous de jouer...

■ Laurent Jayr - dev.net@skeddio.com