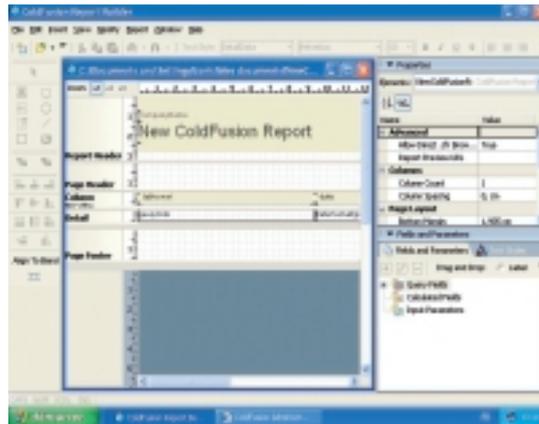


ColdFusion MX 7 : la fusion pour les applications web

Après 10 ans d'existence, ColdFusion tente tant bien que mal de résister à PHP, aux nouveaux JSF et à l'offensive ASP.NET. Présentée comme une évolution très importante, la version 7 apporte surtout des nouveautés de présentation des applications et d'interaction entre utilisateur et application.



Qu'est-ce que ColdFusion ? Il s'agit à la fois d'un serveur d'applications, d'un langage de script pour créer des pages dynamiques et d'un intégrateur HTML – SGBD. Dans une architecture ColdFusion optimale, on sépare l'ensemble des serveurs. Cette configuration permet d'obtenir de meilleures performances et une grande souplesse de montée en charge. Le langage ColdFusion reprend la conception par tag du HTML. Le langage CFML communique directement avec le serveur ColdFusion pour accéder aux données et se connecter aux autres applications. Comme dans un serveur d'application classique, quand une page CFM envoie des requêtes au serveur, celui-ci les réceptionne et traite les demandes en interrogeant tel serveur, telle base, telle application. Contrairement à d'autres serveurs d'applications " fermées ", ColdFusion joue l'ouverture : J2EE, .NET, XML, Web Service, Flash. La création d'un environnement ColdFusion est relativement simple. Tout d'abord, on crée le serveur de données et on crée la ou les bases de données que l'on connecte au serveur ColdFusion. Puis, on développe les pages CFM dans Dreamweaver (ou tout autre outil Web) que l'on poste sur le serveur FTP, qui lui-même les transfère au serveur d'application pour que le serveur web les publie.

ColdFusion 7 : les nouveautés

L'objectif est de passer moins de temps à développer et améliorer l'expérience utilisateur, comme les documents imprimables, les formulaires. La création de reporting en HTML est souvent difficile à créer et à gérer. ColdFusion 7 intègre un véritable outil de reporting totalement intégré, Report Builder. Il permet d'inclure des sous-rapports, ce qui est très pratique dans des données complexes. Pour incorporer les données dans un rapport, on peut passer par des requêtes ColdFusion. Le rapport se génère de manière dynamique (ou statique si cela est nécessaire). Le format de sortie est varié (FlashPaper, PDF, format Office, etc.). L'environnement de Report Builder est particulièrement agréable à l'usage. Il est possible de rapidement créer des rapports via les assistants. On peut soit réaliser des rapports stand alone (sans attacher immédiatement une source de données), soit partir d'un modèle de données existant.

L'autre grosse nouveauté concerne l'impression des pages Web de son application. Afin de faciliter l'impression, il suffit d'inclure dans son code le tag CFDocument. Cette commande permet de formater la page avec les bonnes marges, dans le format que l'on souhaite (PDF, FlashPaper, etc.). On évite ainsi de réécrire son code ou d'inclure du code XHTML. Un des nouveaux marchés en pleine évolution concerne les formulaires dynamiques. ColdFusion le proposait déjà, mais avec une conception

peu aisée. ColdFusion 7 se tourne vers le Rich Client en donnant la possibilité d'utiliser du Flash ou du XForms. On accède ainsi à des interfaces clientes plus fouillées (onglet, grilles, etc.).

Pour le fonctionnement des applications, on peut désormais lancer automatiquement du code à différents moments de l'exécution de son application : en début ou fin d'une session, au démarrage ou arrêt de l'application, en début ou fin d'une requête. On peut même introduire une gestion plus fine des erreurs.

Les autres nouveautés à noter sont : un nouveau moteur de graphiques, un nouveau moteur de recherche textuel (Verity). Le support du XML et des Web Services a été largement amélioré : support de WS-I, fonction de validation du code XML (via DTD ou un schéma). L'ouverture de la v7 permet de nouvelles possibilités à vos applications. On ne se contente plus du simple navigateur, on peut déployer vers des terminaux mobiles (via la passerelle d'événements). Macromedia permet aujourd'hui de créer des applications SMS ou de messageries instantanées.

Côté serveur

Cette v7 fournit une gestion des instances plus souples, tout en isolant chacune d'elles. Il est désormais plus facile de créer des clusters et des montées en charge selon le nombre d'utilisateurs et de requêtes. Comme ColdFusion fonctionne aussi en environnement J2EE (en plus de son fonctionnement indépendant), il supporte les paquetages d'exécution EAR et WAR. Une application ColdFusion publiée via un serveur J2EE se présente sous forme d'un package conforme J2EE et contenant l'ensemble des fichiers et ressources propres à une application ColdFusion (CFML, composants, code, fichier de configuration, etc.). On peut aussi déployer en J2EE en utilisant le bytecode Java.

■ François Tonic

Fiche technique

Produit : ColdFusion MX 7 - **Editeur :** Macromedia

Prix : version développeur gratuite, à partir de 1 299 euros HT.

Les + : éditeur de reporting, découplage des couches, fonctions formulaires, aide aux débutants, déploiement en serveur J2EE, intégration avec Dreamweaver MX.

Les - : pas de support MacOS X, support .NET encore limité, pas d'IDE propre à ColdFusion, pas de version française, mobilité trop restreinte.

Oracle veut séduire les développeurs de BI avec une suite intégrée

Ces dernières années, Oracle a pris du retard sur le marché de l'informatique décisionnelle mais l'éditeur compte bien revenir en force avec l'annonce d'une suite complète pour gérer les données, concevoir et diffuser les états destinés avant tout aux développeurs.

Baptisée Business Intelligence 10g, la nouvelle suite décisionnelle d'Oracle s'adresse avant tout aux développeurs comme le souligne Frédéric Demajeau, Business Development Manager Business Intelligence and Warehousing d'Oracle France " nous avons regroupé et intégré dans un atelier BI tous les outils dont le développeur a besoin afin de faciliter la mise en œuvre d'un projet décisionnel, qu'il s'agisse de simple reporting ou d'analyse multidimensionnelle ".

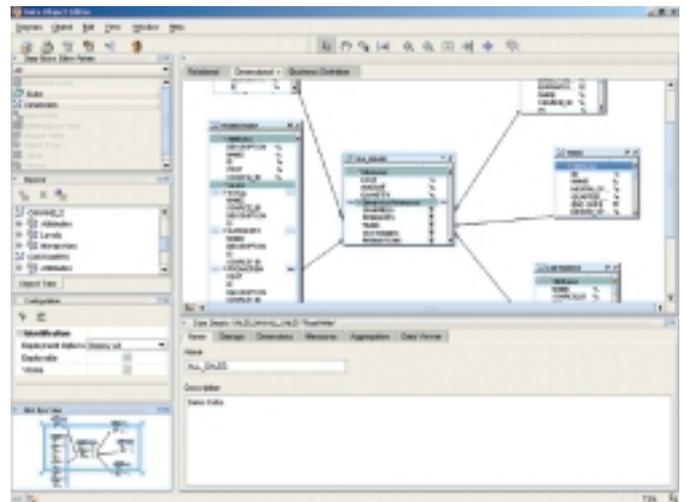
Intégration de toutes les couches de la Business Intelligence

Oracle Business Intelligence 10g comprend en effet la base de données Oracle 10g dotée d'un nouveau moteur OLAP et d'un ETL (extraction, transformation and load) totalement repensé dénommé "Oracle Warehouse Builder", une nouvelle version de Discoverer pour concevoir des tableaux de bord et des rapports d'analyse sur des données aussi bien relationnelles qu'OLAP, un ensemble d'API Java exposant les fonctionnalités OLAP (iBeans) et surtout un tout nouvel add-in OLAP pour Excel. Ce dernier permet en effet d'exploiter les cubes dans Excel de façon complètement dynamique, contrairement à la plupart des outils de ce type qui se contentent d'exporter une vue statique des données. L'utilisateur peut ainsi depuis Excel interroger, afficher et naviguer dans les données OLAP depuis son tableur. Parallèlement à ce lancement "BI Suite 10g", Oracle a récemment présenté la très attendue "Release 2" de Oracle 10g. Celle-ci comporte notamment une évolution majeure de l'outil ETL.

Accent mis sur la qualité des données

Avec Oracle Warehouse Builder 10gR2, Oracle se dote d'un ETL qui est à la fois un atelier de conception (la console Design Center facilite modélisation et développement) et de supervision (la console Control Center facilite le déploiement et l'exécution synchrone/asynchrone des codes). Il comprend également un générateur de code pour les jobs modélisés dans les consoles.

Toujours centré sur la capture des meta-données des diverses sources, OWB 10gR2 s'avère particulièrement pratique pour réaliser le profiling des données de façon à en augmenter la qualité. Il introduit en effet un nouveau module de "Data Profiling" tout à fait remarquable. Le principe même d'un ETL est d'extraire des données, de les qualifier en y appliquant des algorithmes de transformation, puis de les sauvegarder dans le datawarehouse sous leur forme transformée. Le module "Data Profiling" peut être perçu comme un super assistant capable d'automatiser et surtout de normaliser (en déduisant des règles générales) certains processus de transformation et de qualification des données.



Commercialisé à partir de 4 179 euros, l'ETL Oracle Warehouse Builder 10g est disponible pour les plates-formes Windows, Intel Linux, et Solaris/HP-UX/AIX/Tru-64

L'idée centrale de cette nouvelle fonctionnalité est de faciliter les 3 axes fondamentaux d'analyse d'une source pour sa mise en conformation : l'analyse des attributs (vérification des valeurs et des types), l'analyse du référentiel (vérification de la cohérence des données, identification des données orphelines, des conflits, des redondances) et l'analyse fonctionnelle (à quoi sert chaque donnée).

Avec Warehouse Builder 10gR2, en un clic vous obtenez des statistiques sur chaque colonne, et repérez/isolez d'un seul coup d'œil les données non-conformes aux spécifications. De cette analyse, le système est capable d'induire la règle qui qualifie une donnée et de générer une correction automatique des données déviantes basée sur cette règle. Cette correction devient alors un objet "OWB" que l'on incorpore ensuite dans les flux de transformation.

Dans un même ordre d'idée OWB 10gR2 propose un outil d'analyse d'impact des changements sur tout le système. Il permet ainsi d'analyser l'impact d'une modification structurelle dans une table donnée sur les tables et applications dépendantes.

Enfin, dernière autre grande nouveauté, OWB 10gR2 fournit des "Experts", sorte de macro-langage pour exposer des sous-ensembles d'OWB sous forme d'assistants linéaires et ainsi permettre à des services informatiques ou des VARS de produire des mécanismes d'automatisation de tâches répétitives.

■ Marie Varandat

Java 5

Le langage ultime

Connu sous le nom de code " Tiger ", puis sous la numérotation 1.5, la dernière version de J2SEE porte désormais le chiffre 5 et a vu le jour fin 2004. Impossible de s'y tromper : Java 5 constitue une avancée importante pour le langage et même pour la plate-forme Java dans son ensemble. Vous êtes 58% à utiliser Java, et 19% à envisager son utilisation (sondage en ligne sur Programmez.com), preuve de la popularité du langage.

Aujourd'hui, développer en Java n'est pas simple. Il existe toutes sortes d'API, des multitudes de fonctions. Java 5 promet de simplifier quelques approches afin de rendre Java plus simple à coder, même si l'effort au départ n'est pas négligeable.

Java5 n'est qu'une étape, son adoption conditionne l'avenir de votre application. Java doit régler trois problèmes : faire converger le modèle de développement J2SE, J2EE, J2ME, éviter l'apparition de Java "incompatibles", assurer une parfaite interopérabilité et portabilité du code entre les outils et sur les systèmes.

Sur la convergence, il faudra attendre Java 6 et le prochain J2EE pour en voir les premières briques concrètes. Elle permettra de faciliter le codage sur les trois " plates-formes " et d'avoir sans doute un core code commun plus important.

Le risque de l'incompatibilité

L'inquiétant n'est pas là. Il est dans l'apparition de Java " incompatible ", ou avec une interopérabilité discutable : multiplication des frameworks d'éditeurs (souvent incompatibles entre eux), et surtout non stan-



dardisés par le JCP. Et souvent, leur usage se fait pour un outillage spécifique. Chaque éditeur cherche à mettre en avant son framework, en évitant celui de la concurrence. Ainsi la technologie d'interface de type SWT, non standard à Java, mais utilisée dans Eclipse a-t-elle sa place dans le monde Java ? IBM ne s'en soucie guère car son Workplace Client Technology Rich Edition s'appuie dessus (via la plate-forme Rich Client d'Eclipse).

Ne voyons pas tout en noir. Sur l'interopérabilité de Java entre les outils, on peut espérer quelques progrès. Ainsi, le fait qu'Eclipse soit de plus en plus utilisé par les éditeurs permettra de créer un socle technique unique. Ensuite, faut-il aussi que l'ensemble des éditeurs du monde Java fasse un effort. Peut-on accepter, encore aujourd'hui, que migrer d'outils ou de serveur d'application soit parfois aussi difficile ? Le travail d'homogénéité et de cohérence du modèle de développement demeure important. .NET montre un exemple intéressant que Sun et le JCP peuvent suivre.

■ François Tonic

Le futur de Java dévoilé

Au fil des versions, Java s'alourdit, se complexifie. À force d'ajouter des API et fonctions, le développement Java est devenu parfois un exercice de style. Trois modèles de développements, trois versions de Java, une par plate-forme visée, des API à n'en plus finir, des frameworks dans tous les sens, voilà à quoi ressemble aujourd'hui Java et sa faune.

Le slogan, comme une maxime, " écrivez une fois, déployez partout " n'a jamais été aussi peu vrai. Difficile en effet, aujourd'hui, d'avoir un Java d'une portabilité exemplaire. Devant cette situation, Java se devait de réagir rapidement. Java 5 (anciennement Java 1.5, connu aussi par son nom de code, Tiger) constitue une étape cruciale pour le développement Java et son avenir. Grâce à une profonde modification de son fonctionnement et du modèle de développement, il gagne en souplesse et en simplification... à long terme. La balle est désormais chez les éditeurs et les développeurs. À eux, à vous, de s'approprier les nouveautés de Tiger.

Avec l'accroissement rapide de .NET, Java devait absolument offrir un autre modèle. Il lui manquait une approche plus homogène. La plate-forme Microsoft a cet énorme avantage de posséder un modèle unique de développement. Les promoteurs de Java ne s'en cachent d'ailleurs pas. .NET sert de modèle. Après des années passées à ajouter sans cesse des couches, il fallait bien donner de la cohérence et simplifier tout cela. Malgré les attentes, la convergence J2SE, J2EE et J2ME n'est pas pour demain, mais les premiers rapprochements, au moins entre J2SE et J2EE et J2SE et J2ME, se font en grande partie grâce à Java 5 (et aux outils).

Java 5 prépare-t-il réellement l'avenir ?

OUI ! Tiger constitue la charnière entre le Java d'hier et de demain. Il préfigure les évolutions de J2EE 5 et du futur J2SE (Mustang et Dolphin). Plusieurs points militent en faveur de Tiger. Tout d'abord, pour les performances et la qualité de l'application, Tiger apporte une expertise supérieure : runtime et JVM plus performants, code existant compatible (à part exception), API et outils de monitoring et de management en standard, nouveau thème d'interface (Ocean) et meilleure prise en compte du look & feel natif.

Sur la performance, le temps de démarrage d'une application a été amélioré. Pour prépa-



Intégration native du futur Java 6.

rer le futur (très proche) des processeurs 64, Tiger tourne nativement sur ce type de processeurs. Sur le gain de temps de développement, l'argument est à nuancer. A court terme, on perdra en productivité, car il faut apprendre de nouveaux réflexes, et adapter un code existant ne sera pas nécessairement simple. L'apparition des metadata doit, selon Sun, réduire le temps de développement, car une partie de la génération est prise en charge par le compilateur et les outils (cf. EJB 3). L'introduction des types generics dans le compilateur doit permettre d'éviter les erreurs et de détecter plus rapidement les problèmes et les incohérences de code. Java 5 renforce donc son côté typage fort. A terme, on doit pouvoir réduire les erreurs inexplicables du runtime, dues aux ClassCastException.

Tiger est aussi censé apporter une meilleure garantie dans les applications critiques, dans la montée en charge, la qualité et le déploiement. Pour la montée en charge, il y a l'apparition de la librairie " concurrency " afin de mieux développer Java en mode multi-thread, ce qui demeurait une véritable plaie pour les

Avec Jconsole, fini la boîte noire JVM !

Une des avancées appréciables de Java 5 est de faire sauter l'aspect boîte noire de la JVM. C'est pour cela que des fonctions de monitoring et de management sont incluses en standard, grâce à l'apparition de JMX. En complément, on dispose de la suite d'outils Jconsole. Ainsi, il devient aisé d'observer le comportement global à l'intérieur de la JVM. Jconsole est orienté système et non développeur. Cette suite est livrée en standard, alors que précédemment, Sun proposait l'outil Java Stats, uniquement pour sa JVM. Une des particularités de Jconsole est de pouvoir être branché et débranché dynamiquement, fini la nécessité de relancer la JVM. Pour l'instrumentation, on dispose des Mbeans. Par défaut, Java 5 en propose un certain nombre. Il est possible d'en rajouter et d'en créer.

développeurs. De plus, le support du 64 bits est important pour le futur, même si pour le moment, cela demeure axé serveurs (excepté pour les G5 d'Apple). Cela permet d'aller au-delà des 4 Go de mémoire sur un espace heap. De plus, on utilisera plus facilement les processeurs hyper threadés et le multi core. Cependant, si vous portez tel quel un code Java 1.4.x sous Tiger, peu ou pas de soucis, mais le code ne tirera pas profit des nouvelles fonctions de Tiger, hormis sur les performances de la JVM.

Java 5 et J2ME

Tiger ne va pas imposer son modèle stricto sensu à J2ME. De plus en plus de fonctionnalités de la JDK rentrent dans J2ME, même si cela se fait avec un décalage de calendrier. Il est à noter qu'il y aura de plus en plus d'API communes. Ainsi, Swing fonctionne sous J2ME. À l'avenir, on gardera la distinction actuelle : J2SE, J2ME et J2EE. Sun mise avant tout sur les outils de développement pour masquer le plus possible la complexité. NetBeans 4.1 intègre un RAD pour le développement J2ME (que nous trouvons plutôt performant). J2ME est encore en phase de consolidation.

Compatibilité Java 5

Développer avec / en Tiger demande de bonnes compétences Java. Dans un projet de migration, cette technicité s'avérera indispensable. Le compilateur Java 5 signale les principaux problèmes ou obsolescences d'un code Java 1.x passé en Java 5. Par exemple, dans Java 1.4, on disposait des Assert, dans Java 5, il s'agit des enum. Dans ce cas, il faut impérativement adapter le code. Il existe cependant peu de soucis, donc peu de changement pour un binaire Java 1.4.2 tournant en JDK 5. Pour réaliser une migration au sens strict du mot, il faudra réécrire une partie du code pour tirer parti des nouvelles API et méthodes de développement. Une telle migration doit être réfléchie et pertinente.

L'ajout de nouvelles API peut parfois poser problème. Ainsi, l'apparition de la classe java.net.Proxy peut causer des soucis de compatibilité de code. Si vous utilisez du DOM niveau 2, dans la plupart des cas, l'application fonctionnera en niveau 3 (le niveau 3 étant implémenté dans Tiger). Le JVMDI, bien qu'encore présent, sera définitivement retiré de la JDK dans Java 6. Il est vivement conseillé d'utiliser à la place JVMTI (idem pour JVMPDI).

Java 6 alias Mustang (1er trimestre 2006)

• Un nouveau cycle de développement

Java 5 a été conçu sur un cycle long, régulièrement critiqué, de 24 mois. À la suite de cela, Sun a modifié en profondeur le processus de développement. Désormais, Java aura une mise à jour majeure tous les 18 mois. Sans version intermédiaire entre les deux sorties. Il y aura juste des mises à jour de maintenance et de bug fix. Pour preuve, la première update est disponible depuis fin 2004. Pour améliorer la qualité de Java et être plus réactif, Sun a décidé de poster la liste des bugs sur le Web. Et les développeurs peuvent donner leur propre bug et suivre l'évolution de leur prise en compte ou non par Sun. Un système de notification sera disponible dans les prochains mois.

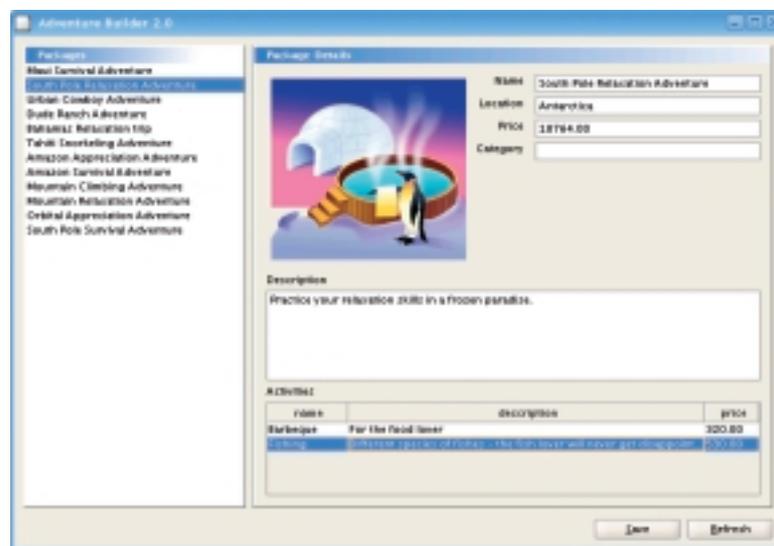
• Continuité de Java 5

Mustang est la continuité de Tiger. Les objectifs principaux sont identiques : compatibilité, stabilité. La compatibilité entre Java 1.x et Java 5 est d'ailleurs un point important que

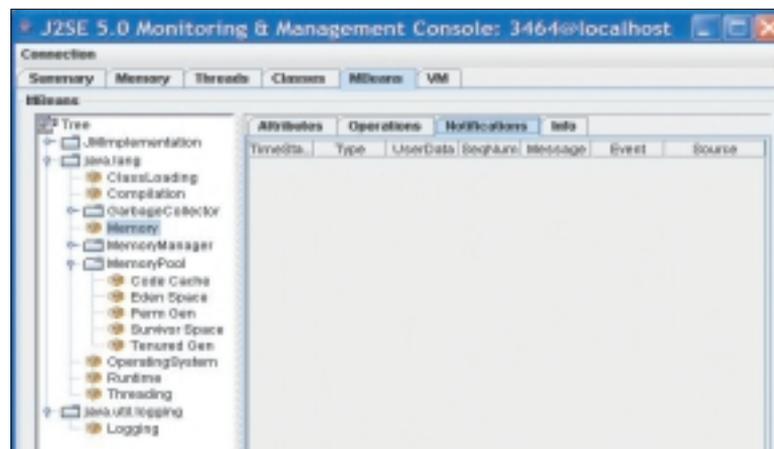
Sun a voulu expertiser au mieux. Ainsi dans Java 5, on dispose d'une API Proxy. On peut attribuer et configurer un Proxy par connexion, ce qui n'était pas possible précédemment. Donc, par défaut, si un code ancien tourne sous une JVM Java 5 qui utilise l'ancien Proxy, la JVM ne prend pas la nouvelle API Proxy de Tiger. Il ne faut pas casser la compatibilité. Si Tiger introduit JMX et la suite d'outils Jconsole pour monitorer l'activité de la JVM, Mustang reprend naturellement ces ensembles et continuera de l'affiner, même si pour le moment peu de détails sont connus.

• Swing simplifié et les sous-projets JDNC et JDIC

Un effort important sera réalisé sur le développement Desktop. L'objectif premier est de simplifier (enfin) le développement Swing et le déploiement. " Swing est très puissant, mais assez compliqué " explique Alexis Moussine-Pouchkine (Sun France). Deux sous-projets Java.net doivent aider à cette simplification : JDNC (Jdesktop Network Components) et JDIC



Un déploiement plus simple et natif avec Java 6.



JConsole en action.

(Jdesktop Integration Components). Tout ne sera pas repris dans Java 6, seuls les éléments matures feront partie de la future JDK.

• JDNC : vers une GUI dynamique

JDNC a pour " mission " de simplifier la construction d'applications Swing. Aujourd'hui, le processus demeure trop lourd, trop complexe. Ainsi, Jtable ne possède pas moins de 300 méthodes. Certes, on peut tout faire avec, mais avec une souplesse inexistante. Il s'agit donc de créer des composants Swing simplifiés avec les méthodes les plus utilisées. JDNC est donc une couche d'extension Swing et ne le remplace pas. De plus, le sous-projet introduit un langage de description d'interface XML, un peu comme XUL ou XAML. Il faudra donc apprendre un nouveau langage. Mais contrairement à XUL ou XAML (et les autres) nécessitant un runtime spécifique, ce dialecte XML nécessite juste une JVM. Bref, on découple fortement le code de la présentation (= GUI). On peut avoir en présentation l'interface de son choix selon le contexte ! Il est d'ores et déjà prévu que NetBeans intègre ce projet open source. L'application s'appuyant dessus utilisera le runtime JDNC. L'application crée une JFrame. Une application JDNC est facile à déployer : Java Web Start, ou par un Jar avec un installateur... JDNC peut répondre à de nombreux cas. Dans le cadre d'applications à interface statique, JDNC fournit un markup localisable, un code plus concis, extensible et personnalisable. Pour le serveur, JDNC permet de générer dynamiquement une interface. Il est capable de fonctionner en mode asynchrone et de travailler avec des données réseaux, propose du Data Binding, et enfin une véritable infrastructure.

Le planning est clairement donné. La version 0.8 doit arriver fin mai 2005 avec comme focus les outils et le développement. La version 1.0 doit arriver lors de la JavaOne en juin prochain. Pour une bonne pratique de développement Swing, Sun prévoit d'inclure dans JDNC et dans Java 6 le Swing Worker.

• JDIC : encore plus de natif !

Aujourd'hui, 90 % des éléments d'interface sont natifs au système. Il demeure donc 10 % à adapter via des kits. Le projet JDIC doit viser à réduire ces 10 %. Il s'agit d'améliorer le look & feel de l'interface de son application Java. J2SE propose déjà beaucoup de choses natives mais sans être 100 % natif. JDIC propose

Témoignage

Java5 et Eclipse

A lors que Java5 est déjà implémenté NetBeans, Eclipse y travaille encore. Le support de Tiger sera effectif avec la version 3.1 sortant en juin prochain. Pour Philippe Mulet (IBM France, travaillant sur le JDT Core d'Eclipse) Java5 est encore assez immature au point de vue du support dans les outils du marché. " On est dans la phase de migration " dit-il. " Dans Eclipse, on peut choisir le parfum de Java que l'on souhaite utiliser (Java 1.x ou Java5). Cela doit être quasi transparent. Cette intégration demande beaucoup de travail. Avec Java5, Java est plus compliqué qu'avant ". Sur la migration du code vers Java5, Eclipse intégrera des services de refactoring pour faciliter ce travail. " Personnellement, je pense que Java5 met l'accent sur le typage fort. Le compilateur se débrouille, la sémantique est plus complexe. Je pense qu'au final, les applications seront plus sûres. L'utilisation des generics rend le code plus complexe avec une nouvelle dimension, la syntaxe alourdit le code. " conclut Philippe Mulet.

Témoignage

Java 5 et Ilog

P our l'éditeur français Ilog, Java 5 constitue une étape importante pour l'univers Java. Nitsan Seniak (Directeur composants techniques) parle même de " signe de vitalité pour Java et pour la pérennité de nos investissements. Il y a un cercle vertueux avec la concurrence de .NET ". Ilog utilise déjà les avancées de Java 5 dans plusieurs solutions. " Les generics et les metadata ont des impacts sur nos logiciels. Les generics nous permettent d'améliorer les services offerts par nos outils. Cela permet à nos développeurs d'avoir une meilleure productivité et un code moins buggé. Les metadata sont encore plus importantes. Elles sont encore sous-estimées ". Sur l'utilisation de Java 5 par les développeurs, le commentaire est sans détour : " encore un peu tôt pour le dire. Java 5 introduit des changements importants. Je dirais que l'on ne touche pas un code qui marche. La question se pose pour les nouveaux projets ". Bref, pour que les développeurs passent à Java 5, les outils devront suivre !

d'ajouter de nouveaux services d'interfaces. Ainsi, il est possible d'intégrer directement le navigateur du système dans son application (via une JFrame), idem pour la messagerie. On se fonde donc mieux avec le système d'accueil. Quand on enverra un mail à partir de son application Java, avec JDIC, on passera par le mailer système. Pour le déploiement, même chose, il s'agit d'être encore plus proche du système. On pourra donc utiliser les packages d'installations RPM, MSI, PKG. On pourrait même avoir un icône tray et créer des économiseurs d'écrans en Java. Pour importer le navigateur dans son application, le code de base est très simple :

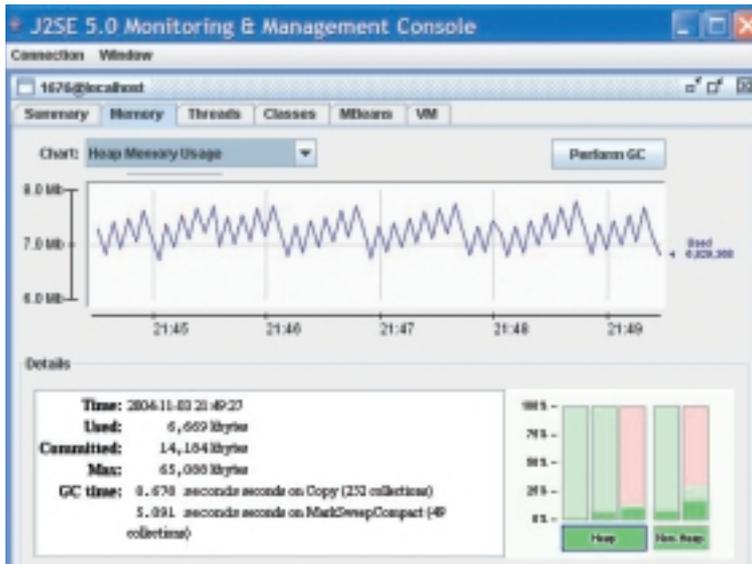
```
import org.jdesktop.jdic.desktop.Desktop;  
  
public class DesktopTest {  
    public static void main(String[] args) throws  
        Exception {  
        Desktop.browse(new URL ("http://www.sun.  
com/"));  
    }  
}
```

Tout comme pour JDNC, Mustang intégrera les éléments matures de JDIC.

J2EE 5 (début 2006)

Durant les premières semaines de 2006 doit sortir J2EE 5 (JSE 244). Certains éléments prennent appui sur Java 5. La généralisation des annotations (" là où cela a un sens d'en faire ", dixit Alexis Moussine-Pouchkine) est un axe fort de ce nouveau J2EE. Plusieurs éléments vitaux ont été modifiés ou ajoutés : EJB 3.0, Web Services Meta Data, JDBC 4.0, JSF 1.2 (maintenant en standard), JAXB / JaxRPC 2.0. La grosse nouveauté de J2EE 5.0 est l'apparition des EJB 3. Le concept a été largement dépoussiéré, au moins au niveau développement. Deux focus à retenir dès maintenant : simplification de l'API et amélioration de la persistance. La " facilité " de développement se résume ainsi :

- modèle d'API réduit et simplifié
 - réduction des classes nécessaires
 - suppression du " deployment descriptor " pour le développeur
- Jusqu'à présent, un EJB ne pouvait pas hériter d'un autre EJB. Et sa structure était éclatée



Gestion mémoire avec JConsole

(code Java, fichiers XML pour la description...). Désormais, EJB 3 et le concept POJO simplifient tout cela. L'annotation précise en début de l'EJB son type. Elle est alors analysée, le code se générant à partir de l'annotation indiquée. De plus, au lieu d'avoir 3 classes distinctes, on n'en a plus qu'une. Autre avantage, le développeur n'a plus à s'occuper de créer le descripteur de déploiement. Celui-ci est automatiquement généré. Cela est possible en utilisant (par EJB 3) la configuration par défaut. Bien entendu, le fichier descripteur est personnalisable. L'autre grand morceau est l'EJB QL pour le mapping O/R. L'ambition est grande. Outre un support natif des requêtes SQL, des requêtes dynamiques, il s'agit d'offrir une bien meilleure persistance des données et de sa gestion. Hibernate a largement inspiré l'outil. Le but est de proposer un standard au monde Java et d'éviter les querelles EJB, Hibernate, JDO. De plus, ce modèle

EntityManager fonctionne aussi avec Tiger, donc en dehors de contextes EJB et serveurs d'applications. Son intégration en standard dans la JDK est prévue dans Java 6. Il est similaire à Hibernate session et JDO Persistance Manager. Bref, on dispose désormais d'un unique format de persistance pour J2SE et J2EE. Cependant, les autres solutions demeurent et il n'y a pas de convergence des API. Une des grandes nouveautés est de pouvoir travailler en mode déconnecté, puis de procéder à la synchronisation (différentes formes de synchronisations disponibles) lorsque la connexion se réactive.

Java 7 alias Dolphing (fin 2007)

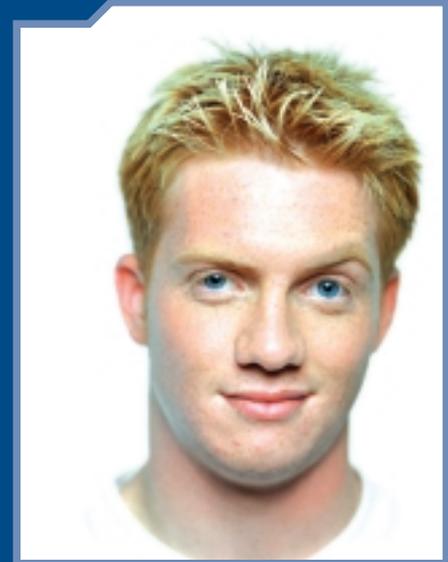
Eh oui déjà ! À peine Java 5 disponible que Sun travaille déjà sur la Java 7. Cette version doit contenir les fonctionnalités non incluses dans Mustang (mais prévue initialement). Pas encore d'information sur le sujet. ■ F. Tonic

Mini Lexique

- J2SE :** Java 2 Standard Edition, orienté desktop, projet non entreprise
- J2EE :** Java 2 Enterprise Edition, orienté entreprise, business, serveur
- J2ME :** Java 2 Micro Edition, pour les applications mobiles
- JMX :** Java Management Extensions, API de gestion et de monitoring
- JVMTI :** JVM Tool Interface, nouvelle API native utilisée par les outils.
- JVMPI :** JVM Profiler Interface, utilisée par ceux qui développent des outils de profiling.
- MBeans :** classes Java correspondant à la définition des JavaBeans.
- JaxB :** Java Architecture XML Binding, architecture servant à faire une liaison avec XML. Utilise des metadonnées.
- JaxRPC :** Api Java pour effectuer des appels RPC utilisant XML
- POJO :** Plain Old Java Objects, utiliser un modèle objet le plus simple possible orienté développeur et ne dépendant pas d'API ou de framework (ex. JDO, EJB 3).

JDO*

l'indépendance totale



J'ai choisi JDO* pour le mapping O/R et l'accès aux données

Versant Open Access JDO* est l'outil surperformant de mapping O/R. Il permet de gérer facilement la persistance objet dans le respect du standard JDO*.

Versant Open Access supporte les principales bases de données telles que : Microsoft SQL Server, MySQL, Hypersonic, DB2, PointBase, Sybase, Informix, SAP DB, InterBase...

Inclus dans Versant Open Access :

- Éditeur Graphique de Diagramme ER, Constructeur de Requêtes et Moniteur de Performance, pour livrer plus vite vos applications.
- Technologie Hyperdrive Versant, Cache Multi Niveaux et PreFetch configurable pour une accélération radicale de vos accès aux données.

Versant France

Direct +33 (0)1 4419 1003
 Fax +33 (0)1 4533 8963
 Courriel voa@versant.fr

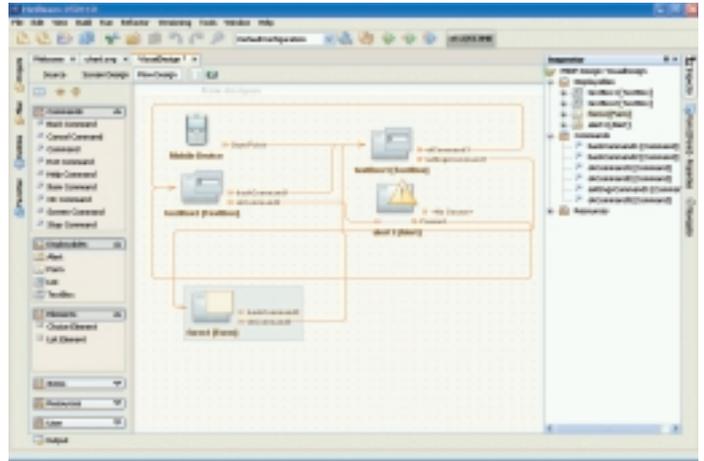
Versant Open Access JDO

*le standard garantissant la liberté de choisir (ou de changer) la base de données cible

➔ www.versant.com

Développer : Quoi de neuf avec Java 5 ?

La version 5.0 de la J2SE a vu le jour fin 2004. La J2SE correspond à un environnement complet de développement d'application java, cette plate-forme inclut en particulier deux entités : à savoir un JRE (J2SE Runtime Environment) et un JDK (J2SE Development Kit). Le JRE fournit principalement un ensemble d'APIs et une machine virtuelle. Le JDK comprend le JRE et un ensemble d'outils pour le développement (compilers, debuggers, etc.)



Tiger est ainsi la dernière proposition de Sun en terme de plate-forme Java, elle intègre plusieurs évolutions du langage Java issues de 15 JSR (Java Specifications Request) différentes. Les JSR sont des propositions d'évolutions de la plates-forme Java émises par le JCP (Java Community Process), groupe de travail sous l'égide de Sun, regroupant plus de 700 spécialistes indépendants ou issus du secteur public et privé (Apache Software Foundation, IBM, Oracle, Bouygues Telecom ...).

Cette version apporte, entre autres, des améliorations spécifiques en terme de facilité de développement, de performance et de supervision d'applications. Nous présenterons dans cet article les principales évolutions du langage Java, ainsi que les changements significatifs des APIs de la nouvelle version de la J2SE.

Un développement facilité par plusieurs évolutions du langage

Les métadonnées

Un des changements les plus intéressants de Tiger, issu de la JSR 175, est l'utilisation des métadonnées. Ces dernières permettent de disposer d'un équivalent des " attributs " de .NET, à savoir le fait de pouvoir "décorer", suivant le principe de programmation déclarative, des éléments spécifiques d'un code source comme des classes, packages ou méthodes. Les développeurs utilisent aujourd'hui fréquemment des tags Javadoc dans leurs commentaires de code pour générer la documentation de leurs programmes. Parfois, l'utilisa-

tion de tels tags peut être détournée de leur rôle initial. Par exemple, XDoclet se base sur ce type de tags pour générer du code et donc augmenter la productivité du développement (génération automatique des interfaces pour les EJB, des fichiers de mapping objet/relationnel, etc.).

Tiger introduit aujourd'hui le concept d'annotation, le principe sous-jacent est d'associer des informations déclaratives (les métadonnées), indiquées par des tags spécifiques, à des éléments spécifiques d'un code source (une méthode, une variable, etc.). Ces métadonnées peuvent être stockées dans les classes (fichiers .class) par le compilateur et ainsi enrichir les éléments auxquels elles sont liées pour générer du code spécifique (concept de XDoclet). Selon le principe d'introspection, elles peuvent être lues également par la JVM ou d'autres applications au runtime et engendrer dynamiquement un traitement particulier. L'exemple suivant illustre une définition d'un type d'annotation. Les annotations sont déclarées comme des interfaces.

```
public @interface MonCalcul
{
    String pourcentage();
}
```

Ce type d'annotation nouveau étant créé, il est alors possible de l'utiliser :

```
public class EssaiAnnotation
{
```

```
@MonCalcul(pourcentage="2")
public void calculer()
{
    ...
}
```

L'annotation de l'exemple est une annotation dite "membre unique" puisqu'elle ne contient qu'un seul membre (pourcentage), elle aurait été de type "marqueur" si elle n'avait pas contenu de membre ou "normale" si elle en avait contenu plus d'un.

La déclaration d'un type d'annotation peut être elle-même annotée. De telles annotations sont appelées meta-annotations. Par exemple :

- La méta-annotation @Retention indique comment l'annotation sera conservée. Ainsi @Retention(RetentionPolicy.CLASS) indique que ce type d'annotation est conservé dans le bytecode
- La méta-annotation @Target indique l'élément sur lequel l'annotation porte. Ainsi @Target(ElementType.METHOD) indique que ce type d'annotation porte sur une méthode de classe.

Dans un contexte de Programmation Orientée Aspect (AOP), les annotations peuvent se révéler très adaptées pour la définition même, sur tous types de composants, de différents aspects (services de logs, sécurité, cache applicatif).

A noter qu'un nouveau framework Open Source de test, TestNG, utilise fortement ce principe des annotations. Ce framework

reprend les grands principes de JUnit en y apportant certaines évolutions : pas de nécessité d'hériter de classes spécifiques (TestCase avec Junit) ou d'utiliser des conventions de nommage particulières pour l'écriture des méthodes de tests. De plus, son modèle d'exécution ne demande pas de ré-instancier la classe à chaque exécution de méthode de test (voir <http://beust.com/testng>).

Types énumérés

Un nouveau type est introduit : enum, semblable au type du même nom en langage C. Ce type, issu de la JSR 201, permet de disposer d'une énumération de constantes. Avant la J2SE 5.0, il était possible de simuler ce type en dédiant une classe à l'encapsulation des constantes. L'exemple de code suivant illustre cette technique utilisant une classe spécifique, il présente une classe OldSemaine composée de 7 constantes.

```
public class OldSemaine
{
    public static final int LUNDI = 1;
    public static final int MARDI = 2;
    public static final int MERCREDI = 3;
    public static final int JEUDI = 4;
    public static final int VENDREDI = 5;
    public static final int SAMEDI = 6;
    public static final int DIMANCHE = 7;

    // jour par défaut
    private int jour = LUNDI;

    // recuperer le jour de la semaine
    public int getJour()
    {
        return this.jour;
    }

    // mettre a jour le jour de la semaine
    // danger : pas de verification de la donnée
    // de mise a jour
    public void setJour(int jour)
    {
        this.jour = jour;
    }
}
```

Le problème de cette méthode est que la classe OldSemaine ne dispose pas de contrôle sur l'affectation de l'attribut jour, il est possible de lui donner une valeur non cohérente, l'erreur ne sera levée qu'à l'exécution. Le nouveau

type énuméré de la J2SE 5.0 est dit " type-safe ", c'est-à-dire qu'une erreur de compilation a lieu s'il est affecté par des valeurs non déclarées. L'exemple de code suivant illustre l'utilisation du type enum.

```
enum Semaine {LUNDI,MARDI,MERCREDI,JEUDI,
VENDREDI,SAMEDI,DIMANCHE};
```

Dans l'exemple de code ci-dessous, la classe EnumTestSemaine affiche le jour de la semaine en utilisant soit la classe OldSemaine (méthode oldQuelJour() : n'importe quelle valeur peut être passée au setter de l'attribut jour), soit la classe Semaine (méthode quelJour()). Notons qu'avec le type enum, un espace de nommage est fourni pour chaque énumération, ce qui évite d'utiliser en préfixe de constante, par exemple dans le switch, la classe à laquelle cette constante est rattachée.

```
public class EnumTestSemaine
{
    public static void main (String args[])
    {
        oldQuelJour();
        quelJour();
    }

    public static void oldQuelJour()
    {
        System.out.println("Quel jour sommes-nous?");
        OldSemaine oldSemaine = new OldSemaine();

        // danger : il est possible d'affecter ici un état
        // non cohérent
        oldSemaine.setJour(OldSemaine.DIMANCHE);

        int jourSemaine = oldSemaine.getJour();
        switch(jourSemaine)
        {
            case OldSemaine.LUNDI:
                System.out.println("Nous sommes
                lundi");
                break;
            ...
            case OldSemaine.DIMANCHE:
                System.out.println("Nous sommes
                dimanche");
                break;
        }
    }

    public static void quelJour()
    {
```

```
        System.out.println("Quel jour sommes-nous?");
        Semaine semaine = Semaine.LUNDI;
        // changer d'état est facile
        semaine = Semaine.DIMANCHE;

        switch(semaine)
        {
            case LUNDI:
                System.out.println("Nous sommes
                lundi");
                break;
            ...
            case DIMANCHE:
                System.out.println("Nous sommes
                dimanche");
                break;
        }
    }
}
```

Autoboxing/ auto-unboxing des types primitifs

Pour effectuer une conversion en types primitifs (int, boolean, char, float, long, byte, double, short), l'écriture de code était jusqu'ici fastidieuse : il s'agissait d'utiliser une classe Wrapper pour chaque conversion (respectivement Integer, Boolean, Character, Float, Long, Byte, Double, Short). L'exemple de code suivant illustre une conversion classique (appelée Boxing) avant Tiger.

```
List list = new ArrayList();
int i = 100;
list.add(new Integer(i));
```

A noter que l'opération de conversion inverse (UnBoxing) nécessitait également un recours à un Wrapper.

Avec Tiger, et l'introduction de l'autoboxing et de l'auto-unboxing, la conversion des types primitifs est automatique (recours à un Wrapper non nécessaire), comme l'illustre l'exemple de code suivant :

```
List list = new ArrayList();
int i = 100;
list.add(i);
```

L'import statique

Tiger propose une variante à l'import classique de classes ou d'interfaces, l'objectif est de pouvoir importer dans un programme des constantes ou des méthodes statiques et d'y faire référence en utilisant un espace de nom-

mage dédié. Cette simplification de code apportée par Tiger permet ainsi de ne plus préfixer les composants statiques importés par les classes auxquelles ils appartiennent, au risque de conflits sur les espaces de nommage (un même nom de méthode peut être utilisé dans différentes classes).

Le code suivant illustre cette nouvelle fonctionnalité, basée sur l'exemple classique de la classe Math de la J2SE, classe ne comportant nativement que des méthodes statiques.

```
import static java.lang.Math.* ;
...
sin(x) ; // équivalent de Math.sin() avec java 1.4
cos(x) ; // équivalent de Math.cos() avec java 1.4
```

Les arguments variables

Tiger propose de disposer d'un moyen de passer plusieurs arguments en paramètres de méthodes. Le nombre n'est pas connu à priori, il n'est pas fixé à l'écriture de la méthode. Il n'est donc plus nécessaire d'utiliser un tableau d'objets par exemple, lorsque le nombre de paramètres d'une méthode n'est pas connu. L'exemple suivant illustre cette nouvelle fonctionnalité (l'opérateur ellipse noté sous la forme " ..." permet d'indiquer que le nombre d'arguments dans la signature de méthode est variable).

```
String ecrire(Object...mots)
{
    // bloc d'instructions
}

// la methode calculer est invoquée avec 2 arguments
calculer("mot1", "mot2");

// la methode calculer est invoquée avec 4 arguments
calculer("mot1", "mot2", "mot3", "mot4");
```

Les types génériques

Le polymorphisme caractérise un principe selon lequel un nom d'objet peut désigner des instances de classes différentes, issues d'une même arborescence. Le polymorphisme est soit de type universel, agissant sur un nombre indéterminé de types ayant un comportement commun, soit de type ad-hoc, agissant sur un nombre fini de types sans comportement commun (principe par exemple du casting). Tiger introduit avec les types génériques la possibi-

lité en Java de réaliser un type de polymorphisme universel, le polymorphisme dit paramétrable, selon lequel une seule abstraction s'applique à un ensemble de types. Les types génériques donnent la possibilité de ne pas avoir à contrôler le type d'une valeur lors de l'exécution, ils permettent de définir des comportements communs sur des objets sans avoir à les typer.

Classiquement, un contrôle du typage est fait à chaque compilation, ce qui peut être cause d'exceptions si le type de casting n'est pas adapté (la fameuse ClassCastException).

Les types génériques introduits par Tiger permettent ainsi de détecter ce type de problème dès la compilation. L'exemple suivant permet de comparer deux listes : listOld est définie classiquement sans utiliser les types génériques, list est définie en utilisant la notation <> permettant de préciser le type des objets.

```
import java.util.*;
import java.lang.*;
public class EssaiGenerics
{
    public static void main (String args[]){

        ArrayList<Integer> list = new ArrayList<Integer>();
        ArrayList listOld = new ArrayList();
        list.add(0, new Integer(9));
        listOld.add(0, new Integer(9));
        Integer instance = list.get(0); // Pas besoin de " cast "
        Integer instanceOld = (Integer) listOld.get(0); // besoin de " cast "
        list.add(1, new Long(4)); // Plantage a la compilation
        listOld.add(1, new Long(4)); // Pas de plantage a la compilation
        Integer instanceOld2 = (Integer) listOld.get(1); // le cast est faux : pas de plantage a la compilation
        // compilation

        // mais seulement a l'execution
    }
}
```

Simplification des itérations

Tiger propose un nouveau formalisme pour opérer une itération. Jusqu'à présent l'API Collection et sa classe Iterator était utilisée. Aujourd'hui Tiger permet de simplifier le code en apportant une alternative à l'utilisation de l'Iterator dans le cas d'une itération sans modi-

fication de la collection. L'exemple suivant illustre l'utilisation de cette nouvelle notation.

```
ArrayList<Integer> list = new ArrayList<Integer>();
for (Integer i:list)
{
    // blocs d'instructions
}
```

Dans l'exemple, list est obligatoirement un tableau ou doit implémenter la nouvelle interface java.lang.Iterable (en l'occurrence, avec J2SE 5, ArrayList étend bien java.lang.Iterable)

Entrées et sorties formatées

Une nouvelle API, Scanner, est disponible avec Tiger donnant la possibilité de lire des données à partir de tout flux d'entrée. Tout objet implémentant l'interface Readable peut être manipulé via cette API : elle permet, entre autres, de travailler à base d'expressions régulières et d'opérer des conversions de textes. L'exemple suivant permet de lire l'entrée standard et d'accéder aux éléments du flux par la méthode next().

```
Scanner scan = new Scanner(System.in);
String element = s.next();
```

Quant à la sortie standard, Tiger permet dorénavant aux développeurs de disposer de messages formatés, via une heuristique proche de la fonction printf() du langage C. L'exemple de code suivant illustre l'utilisation de la méthode printf() :

```
String nom= "geronimo";
int age= 27;
System.out.printf("je suis %s et mon age est %d ans \n", nom, age);
```

De nouvelles APIs pour une plateforme plus performante

Au-delà du langage, Tiger propose des évolutions sur les APIs du JDK. Nous nous focalisons sur quelques une des APIs importantes de la J2SE 5.

Jusqu'ici la machine virtuelle permettait déjà de gérer l'ordonnancement de plusieurs threads (processus légers) simultanés. La programmation multi-thread est très courante en Java (contrairement à l'utilisation de threads en C par exemple) : les fonctionnalités multi-tâches intégrées à Java le rendent par exemple

parfaitement adapté à l'écriture d'applications côté serveur (traitement simultané de requêtes de plusieurs utilisateurs pour un serveur web par exemple). Une nouvelle API de concurrence (package `java.util.concurrent`), issue de la JSR 166, a ainsi été intégrée à Tiger. Cette dernière comprend en particulier un framework d'exécution de tâches et un pool de thread. L'interface `Executor` permet de spécifier le cycle de vie de chaque thread.

L'implémentation `ScheduledThreadPoolExecutor` et l'interface `ExecutorService` permettent de gérer des queues de tâches et de spécifier leur ordonnancement. L'utilisation de cette nouvelle API va permettre une gestion avancée de l'exclusion mutuelle entre threads (sections critiques), et une réduction des risques d'interblocage (deadlocks).

JMX (Java Management eXtension) était, avant Tiger, une API externe. Elle a été intégrée à la J2SE 5 (JSR 3 pour JMX et JSR 160 pour JMX Remote) et permet de définir une supervision de la machine virtuelle. Il est ainsi possible d'accéder à des informations intrinsèques de

la JVM telles que l'état de la mémoire ou de ses processus. L'outil de monitoring JMX-compliant `Jconsole` permet en l'occurrence de fournir des informations, via une console graphique, sur les performances et la consommation de ressources des applications s'exécutant sur la JVM.

Au-delà des évolutions impactant le monitoring de la JVM, il est également possible à partir de la J2SE 5 de disposer d'une API dédiée au profiling d'applications, via l'outil `JVMTI` (issu de la JSR 163). Cet outil permet ainsi de créer des agents pour analyser le code exécuté ou disposer de fonctionnalités de debugging.

Le développement des applications client riche a été fortement amélioré avec Tiger : il est désormais possible d'utiliser le concept des skins pour paramétrer de manière déclarative (via des fichiers XML) ou par programmation l'apparence des interfaces graphiques. De plus, Tiger apporte de meilleures performances sur ce type d'applications client riche en terme d'optimisation d'occupation des res-

sources système et de rapidité de chargement. Parallèlement à ces changements au niveau des APIs, nous pouvons également noter de fortes améliorations concernant le support XML et des évolutions sur l'API JDBC (cinq nouvelles implémentations de la classe `JDBC RowSet` permettant, à l'inverse des objets `ResultSet`, de travailler sans être connecté en permanence à une source de données). Tiger a également intégré la JSR 28, spécification décrivant une API client et serveur SASL (Simple Authentication and Security Layer) et la JSR 204 pour le support de caractères Unicode supplémentaires.

<http://jcp.org/en/home/index>

<http://java.sun.com/developer/technicalArticles/releases/j2se15/>

<http://java.sun.com/j2se/1.5.0/docs/rel-notes/features.html>

<http://java.sun.com/j2se/1.5.0/docs/guide/language/>



■ Agnès CREPET - Architecte technique, SQLI



LiDO XD 3.1

Solution de persistance multi-standard JDO, EJB3... Multi-sources de données, SGBDR, SGBDO, XML, fichier, et mainframe. LiDO XD permet désormais d'optimiser dynamiquement la performance de vos applications Java.

Cet outil de développement gratuit est en téléchargement libre sur xdn.xcalia.com

Venez découvrir comment nos clients et notre partenaire SQLI utilisent LiDO XD 3.1 :

Séminaire le 14 Avril, 9h30, porte de Versailles, inscrivez-vous ; seminaire@xcalia.com ou 01-56-56-12-50

Les marchés Java : L'Eldorado du mobile

Dans un marché aussi volatile que le marché informatique, on peut tout même distinguer plusieurs tendances dans les secteurs Java. Selon Gartner, il y aura en 2007 près de 3 millions de développeurs Java. Quels sont les marchés prometteurs et ceux qui sont ou seront matures ?

Les applications entreprises desktop

Le marché continuera sans doute à être important. Si le côté serveur avec J2EE 5 et EJB3 se renforcera en 2006, les disponibilités d'outils plus ergonomiques et plus productifs simplifieront le développement d'applications J2EE. On peut aussi prévoir une belle croissance des JSF, en lieu et place de Struts, un travail de migration en perspective. Ceux-ci sont désormais graduellement intégrés dans les IDE du marché. Cependant, avec l'émergence de .NET en entreprise, cette plate-forme devant commencer à croître cette année, on peut s'attendre à des besoins soit de migrations, soit d'interopérabilité entre les deux environnements.

Du point de vue desktop, la visibilité est un peu plus incertaine. Java 6 va-t-il chercher à mieux s'imposer sur les applications desktop ? Java 5 ne le permet pas. Le client riche Java peut cependant être un marché en devenir très important pour la plate-forme Java, et donc les développeurs. Le fait qu'Eclipse ait un dérivé dédié aux Rich Client va dans ce sens. La disponibilité du framework Workplace d'IBM (basé sur la plate-forme Rich Client d'Eclipse) conforte cette idée, même si, sur ses solutions, Java n'est pas l'unique prétendant. Un marché à suivre de très près.

Si sur le portage et migration du code existant vers Java 5, il est trop tôt pour le dire, par contre, pour les nouveaux projets, le choix de Java 5 s'impose de facto. Une remise à niveau des connaissances sera nécessaire pour réellement connaître les subtilités du modèle Java 5.

Les applications mobiles

Avec l'explosion du téléphone portable, la demande de services et d'applications ne fait que croître. Et la demande va continuer à exploser, même si la 3G demeure encore timi-

de. Sur ce marché, Java constitue une plate-forme de développement de référence. Avec plus de 100 millions de téléphones équipés de J2ME en 2003 (la tendance semble être identique en 2004), Java est incontournable. Il s'agit de créer des services, des applications, et des jeux. Le rapprochement progressif de J2ME sur les API et le modèle de J2SE, des IDE de meilleure qualité avec RAD, la disponibilité de Swing, offrent de nouvelles perspectives intéressantes.

Durant le dernier 3GSM World Congress, Sun n'a pas manqué d'être largement présent. Il a donné des chiffres assez évocateurs : 750 millions de smartcards Java, 400 modèles de téléphones supportant Java, provenant de 40 constructeurs ! Un des objectifs de Sun pour 2005 et les prochaines années, est de proposer des solutions spécifiques aux Telco. C'est à dire de proposer des solutions et des technologies leur permettant de pérenniser l'investis-

sement, d'être plus réactifs et rentables. Cela passe par plusieurs initiatives :

- proposer une API Java au-dessus de J2EE et répondant au standard OSS. Cela doit permettre de faciliter les développements d'applications mobiles professionnelles et de services.
 - Préparer dès maintenant le passage à la 3G en proposant des outils de tests et de profiling des applications Java pour le portable en 3G.
 - Proposer une nouvelle solution de messageries pour les Telco (via Mobeon), basée sur les services IP et supportant les données, les mails, la voix, la vidéo. Fonctionne sur Solaris.
 - Renforcer le positionnement Rich Media de J2ME.
- Les constructeurs de téléphones proposent souvent des kits de développement Java répondant aux spécifications de leurs modèles. Pour le moment, Microsoft demeure marginalisé sur ce secteur.

■ F.T.

Les tendances

	Marché	Tendance
PDA	Les PDA non Windows CE résistent assez bien. Concurrence de VB, .NET et C/C++. Forte poussée de Microsoft.	Incertaine. Marché en baisse.
Smartphone	Le marché continue à exploser (services, jeux...). • J2ME est largement présent en standard sur les téléphones. • Sun prépare l'arrivée en masse de la 3G. • Symbian continue à s'imposer, • Microsoft peine à percer sur ce marché. • Nouveaux outils plus visuels pour la conception. • Arrivée de Swing.	Excellente, notamment par les jeux et les applications.
TabletPC	Peu favorable à Java, a priori. Domination de .NET / Windows.	Pour le moment pas grand chose à espérer, sauf portages spécifiques
Applications entreprises	Marché mature. Si pour les anciens projets en Java, le langage restera sans doute le même, pour les nouveaux projets, la concurrence devient intéressante notamment avec .NET.	Bonne. L'arrivée de .NET sur les nouveaux projets peut provoquer un premier renversement de marché.
Applications desktop	Sur le Rich Client, Java bien que bien placé n'est pas le seul à vouloir occuper le terrain	Mitigée. Même si Java constitue un langage de choix (notamment avec le Rich Client d'Eclipse).

Migrer du code Java 1.x vers Tiger

Il n'y a que de bonnes raisons de passer à Java 5, mais migrer de l'ancien code Java, surtout venant de Java 1.3, n'est pas forcément immédiat. Ce n'est cependant pas une tâche trop fastidieuse.

Les améliorations apportées par Tiger sont telles qu'il semble évident de se mettre à coder avec lui. Il est également tentant de migrer de l'existant vers Tiger, car la nouvelle JVM est globalement plus performante et assure, selon Sun, une meilleure portabilité. Enfin, la migration peut être forcée, par exemple si on incorpore des améliorations écrites en Java 5 à une application existante. Selon Sun, migrer du byte-code 1.4.2 ne doit pas poser de problème, et cela semble bien le cas, mais cela ne dispense pas de tests méthodiques. Est on jamais trop prudent en informatique ? Par contre, si l'on vient de 1.4 et surtout de 1.3, les choses ne coulent pas de source, si l'on peut dire. Sun Microsystems fournit un guide de migration téléchargeable à <http://java.sun.com/j2se/>. Ce document est la bible à laquelle se référer. On y découvre que les problèmes de migration sont présents à l'exécution, au déploiement, à la compilation et que les développeurs d'outils et les implémenteurs sont particulièrement concernés par quelques finesses.

Les problèmes à l'exécution

Evidemment, le plus simple est de prendre les paquets de byte-code et de les déposer sous la nouvelle JVM. Une phase rigoureuse de tests viendra valider l'opération, et le cas échéant, on pourra en rester là. Toutefois, il est probable que quelques grains de sables perturbent la mécanique. Notamment, en ce qui concerne AWT et Swing. Par exemple, le membre `static java.awt.event.MouseEvent.MOUSE_LAST` n'a pas la même valeur numérique à partir de 1.4. Comme cette valeur a été codée en dur par le compilateur, une recompilation est nécessaire. En outre il n'est pas exclu qu'un autre cas de cet acabit ne soit pas documenté par le guide. On aurait alors à faire face à l'exécution, à un bug difficilement identifiable. Pour cela, il semble qu'il soit sage de recompiler le code sous Tiger pour toutes migrations. Ce sera aussi l'occasion de profiter

de corrections de bugs Swing ou d'améliorations, comme la gestion du Glisser-Déposer (cf. la bible). Les interfaces utilisateurs ne sont pas seules concernées.

Par exemple, le support de CORBA est bien meilleur à partir de 1.4. Raison supplémentaire d'envisager une retouche et une recompilation du vieux code. Sous Tiger, les packages `org.apache` ont été déplacés. Là aussi, il est probablement mieux, pour simplifier les déploiements à long terme, de recompiler plutôt que d'agir sur le classpath. Enfin, si le code doit être retouché, c'est sans doute une très bonne occasion d'en améliorer la lisibilité en profitant des génériques, outils de refactoring aidant éventuellement.

Les problèmes de recompilation

Du byte-code 1.3 qui tourne sous Tiger n'est pas forcément directement recompilable pour autant, cela serait trop beau. Ainsi en est-il des pilotes JDBC. Un pilote 1.3 tournera sans doute très bien, mais ne compilera plus à partir de 1.4, ou autrement dit, à partir de JDBC 3.0, dont quelques interfaces comportent des méthodes supplémentaires.

Avant de procéder à la compilation, on vérifiera encore à l'aide d'un outil quelconque que des mots clés ne sont pas employés comme identifiants dans le vieux code. Il s'agit de :

```
assert (depuis 1.4)
enum (depuis 1.5)
```

Si l'on ne souhaite pas retoucher le code il est toujours possible de passer les options `-source 1.3` ou `-source 1.4` au compilateur. Mais ce n'est qu'un pis aller. Un reculer pour mieux sauter. Le compilateur peut aussi rencontrer des ambiguïtés. Sous Tiger il existe deux classes Proxy :

```
java.lang.reflect.Proxy
java.net.Proxy
```



Tout code qui travaille avec le mécanisme de réflexion rendra dubitatif un compilateur qui rencontrera :

```
import java.lang.reflect.*;
import java.net.*;
```

Le remède est bien entendu d'ajouter un `import` explicite :

```
import java.lang.reflect.Proxy;
```

Pour conclure

Nous n'avons fait que signaler des problèmes. Le tableau est il noir pour autant ? Non. Guide de migration en main, et ainsi qu'en attestent les exemples que nous avons montrés, nous estimons que les retouches à effectuer ou les précautions à prendre sont relativement légères en regard des bénéfices obtenus. Le point le plus épineux concerne l'initialisation des classes évaluées en tant que littéraux, mais ceci ne concerne à priori que les implémenteurs. Alors autant faire maintenant un travail qui sera peut être plus difficile lorsqu'il s'agira de passer à Java 6 :-)

■ Frédéric Mazué - fmazue@programmez.com

JDO 2.0 : controversé, mais tellement utile

Malgré l'opposition de JBoss principalement, JDO 2.0 a reçu l'approbation du JCP fin février 2005. C'est donc le bon moment de jeter un œil à ce framework controversé, mais qui constitue l'une des solutions de persistance transparente en Java les plus universelles.

A bien regarder nos codes, et ceci depuis des années, une partie non négligeable est dédiée à l'interrogation et l'insertion de données en base de données, ou à l'écriture sur fichiers. A y regarder d'encore plus près, l'essentiel de ce code est composé d'instructions destinées à charger (ou vider sur disque) les classes, collections et propriétés. Cela s'explique en grande partie par le fait que la programmation objet (OOP) est postérieure à l'usage répandu des SGBDR.

Pourtant, et typiquement dans le cadre de langages tels que Java ou C#, l'idée de pouvoir manipuler, stocker et récupérer directement les objets depuis la base de stockage a fait son chemin et convaincu bien des développeurs.

Reste à savoir ce qu'on entend réellement par objets ? Il y a ceux d'ordre purement applicatif qui modélisent la logique applicative et n'ont aucune persistance. Il y a ceux qui véhiculent des données, des états, des comportements (des Domain Objects en Java) et qui sont eux de nature persistante. Toute la difficulté réside dans la gestion de cette persistance de sorte qu'elle soit la plus efficace et la plus transparente possible pour le développeur.

Face à cette problématique de persistance "transparente" des objets, le monde Java a abouti simultanément à deux API concurrentes : EJB-CMP et JDO. Ces deux API viennent compléter une offre déjà riche d'autres solutions basées sur des bases de données objets (telles que Gemstone, VDS, FastObjects, ObjectStore) ou des outils de "mappings

O/R" (tels que Toplink ou Hibernate). Sans compter bien sûr les solutions basées sur JDBC ou encore les "serialization". Mais un "objet" ce n'est pas qu'un ensemble d'états, c'est aussi des héritages, des encapsulations, des comportements que ces solutions ne gèrent que partiellement, voire pas du tout. Les EJB-CMP adressent une problématique d'un niveau d'abstraction plus élevé. Mais leurs jours sont comptés avec leur remise en question au sein d'EJB 3.0. Même si EJB 3.0 se veut plus consensuel, cette approche reste relativement peu naturelle pour des développeurs JDBC. Pour ces derniers, qui constituent la très vaste majorité des développeurs Java, JDO demeure une approche à la fois plus logique et plus simple.

Historique JDO

Les solutions comme Toplink (racheté par Oracle) sont simples à mettre en œuvre, mais elles restent hautement propriétaires avec des API bien spécifiques qui lient définitivement votre application à ces vendeurs. Une solution Open Source comme Hibernate (dont la destinée est désormais supervisée par JBoss) n'est d'ailleurs pas plus standard en terme d'API que celle d'Oracle et vous lie à la base de données.

JDO (Java Data Objects) a été conçu en 1999 mais n'a été approuvé qu'en Mars 2002. C'est une spécification et une référence d'implémentation destinée à prendre en charge et automatiser les problématiques de persistance. Elle permet aux développeurs d'utiliser leur modèle objet comme modèle de données et ceci de façon standardisée, quel que soit ensuite le support des données (SGBDO, SGBDR, etc). Mais il est cependant essentiel de comprendre que JDO n'était pas jusqu'à présent un standard de mapping O/R (Objet-Relationnel).

JDO par l'exemple

Pour ceux qui ne savent pas à quoi ressemble JDO en matière de codage, revenons rapidement sur la simplicité de mise en œuvre de la chose :

Les polémiques JDO 2 / EJB3

Une sérieuse polémique a précédé l'approbation des spécifications JDO 2. En effet lors d'une première tentative d'adoption par le JCP en Janvier, la spécification fut rejetée sous la pression de JBoss mais également Oracle, Sun et IBM.

Tout part en réalité des travaux actuels sur EJB 3, qui n'adopte pas JDO 2 comme solution de persistance (en remplacement des principes actuels des EJB-CMP) mais opte pour un développement de leur propre POJO basé sur Hibernate, la solution de mapping O/R open source et contrôlée par JBoss. Cette décision est notamment motivée par la philosophie plus SQL du langage HQL comparée à l'approche plus programmatique de JDOQL. JBoss et les autres ont ainsi tenté d'étouffer JDO à leur propre profit.

Suite à la mobilisation de la communauté JDO (qui n'avait aucune envie d'attendre la finalisation d'EJB 3.0 en 2006 alors que JDO 2.0 est d'ores et déjà prête et résout l'essentiel des difficultés actuelles), la spécification a été finalement approuvée fin Février, sur le même texte (juste enrichi d'une volonté de rapprochement de JDO vers EJB dans une future spécification 2.1), JBoss, Oracle et IBM s'étant cette fois abstenus.

```
import javax.jdo.* ;

public class MaPersistence
{
    Public static void main(String[] args)
    {
        // initialisations principalement dédiées
        // à l'instanciation d'un PersistenceManagerFactory
        // createConnexionFactory est une routine qui réalise les initialisations de
        connexion au SGBDR
        // et qui contient du code directement dépendant de l'implémentation et
        de la base visée
        PersistenceManagerFactory pmf = JDOHelper.getPersistenceManager
```

```

Factory(System.getProperties());
    pmf.setConnectionFactory( createConnexionFactory() );

// Pour accéder à la persistance il faut obtenir un gestionnaire de persistance
PersistenceManager pm = pmf.getPersistenceManager();

// création d'un objet
Personne personne = new Personne("Marie", "Varandat", 75, 20)

// Pour rendre l'objet persistant il suffit d'utiliser makePersistentAll
pm.currentTransaction().begin();
pm.makePersistentAll( personne );
pm.currentTransaction().commit();

pm.close();
pmf.close();
}
}

```

Par principe, JDO utilise un fichier XML de description qui indique au système les informations sur les classes que l'on souhaite rendre persistantes. Celui-ci assure le lien entre le framework JDO et les spécificités de l'implémentation utilisée. Par exemple :

```

<?xml version="1.0" encoding="UTF-8"?>
<jdo>
  <package name="nomdemonpackage">
    <class name="Personne" identity-type="datastore">
      </class>
    </package>
  </jdo>

```

JDO 2.0

Approuvée début Mars 2005, la ratification de la spécification 2.0 aura été l'objet de bien des débats et controverses, principalement menés d'ailleurs par les fournisseurs de solution de "mapping O/R" qui voient d'un très mauvais œil que l'on vienne ainsi marcher sur leurs plates-bandes...

En effet, la principale innovation de JDO 2.0 réside dans la standardisation du "mapping O/R". Dans JDO 1.0, le "mapping O/R" était laissé à chaque vendeur et donc variait d'une implémentation à l'autre. Une

Point de vue éditeur

Jean-Claude Bellando

directeur général de Versant France



Face à la gestion manuelle, JDO est bien meilleur, il n'y a plus photo ! JDO 2 est une avancée importante pour java et pour le monde objet de manière générale. Il nous fallait un standard fort et si le débat était encore possible il y a deux ans, aujourd'hui ce n'est plus le cas : le développeur qui estime pouvoir faire plus performant à la main se trompe. Avec les outils qui automatisent la gestion de la persistance, et JDO 2 en particulier, on fait au moins aussi performant et on monte surtout mieux en charge grâce à la gestion automatisée du cache, opération particulièrement complexe manuellement. En plus, c'est zéro coût en développement. Et ce, quelque soit la solution retenue, car nous proposerons aussi bien JDO 2 que EJB3, mais également une solution équivalente pour .NET.

application JDO 2.0 est désormais directement portable d'un OS à l'autre, d'un JCA (serveurs d'application) à l'autre, et d'une implémentation JDO à l'autre. Voilà qui met définitivement fin à l'un des principaux griefs des adversaires de JDO. L'autre grande avancée de JDO 2.0 réside dans son "Query Language" JDOQL. JDO 1.0 était en effet limité par son incapacité à accéder à une sous classe sans passer par une navigation [de type contains()] partant de la classe principale mais également par l'absence de fonctions de groupage (équivalentes aux closes "group by" et "having" de SLQ) et par voie de conséquence de fonctions d'agrégation (équivalentes aux instructions Count(), Sum(), Avg() de SQL). Ces limitations sont désormais toutes levées. Dès lors, l'interrogation en JDOQL acquiert une souplesse très proche de celle d'un langage comme SQL.

Typiquement, une requête en JDOQL peut ressembler à l'exemple suivant :

```

String filter = "age > 20" ;
Query q = pm.newQuery (Personne.class, filter);
Collection pers = (Collection) q.execute() ;

```

Et on peut donc désormais réaliser des requêtes de cet ordre : la requête remonte la moyenne d'âge (=agrégation) des adultes (plus de 18 ans=filtrage) par département de plus de 100 personnes (=groupage) :



Tableau des implémentations JDO

Produit	Editeur	Licence	Version	Description
IntelliBO	SignSoft	commercial	3.7	cible les SGBDR à travers des connexions JDBC
JDO Reference Implementation	Sun	open source	2.0	utilise son propre datastore (en b-tree)
JPOX 1.1	jpox.org	open source	1.1	cible les SGBDR à travers des connexions JDBC
Kodo	SolarMetric	commercial	3.3.0	cible les SGBDR à travers des connexions JDBC
Lido	Xcalia	commercial	3.0	cible les SGBDR à travers des connexions JDBC
OJB (ObjectRelationalBridge)	Apache.org	open source	1.0.1	un outil de mapping O/R compatible JDO et ODMG 3.0
Open Access JDO (ex JDO GENIE)	Versant	commercial	pro	une implémentation simple et rapide s'appuyant sur JDBC
Speedo	ObjectWeb	open source	2.1	utilise les framework Jorm et Medor pour l'accès aux SGBDR
Triactive JDO	Tjdo	open source	2.1	cible les SGBDR à travers des connexions JDBC
XORM	Xorm	open source	beta6	n'utilise aucun "class file enhancer" avant déploiement.



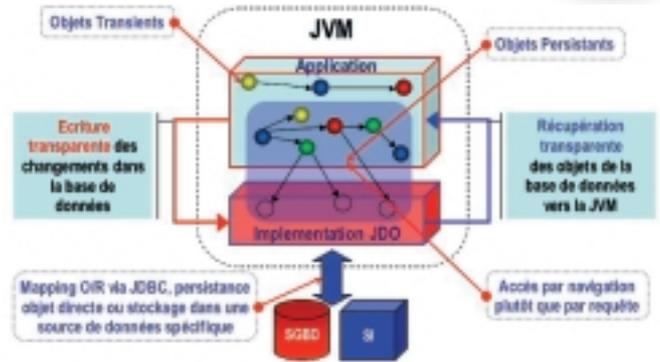
```
String filter = "age > 18" ;
Query q = pm.newQuery (Personne.class, filter);
q.setResult("departement, avg(age) ") ;
q.setGrouping("departement having count(this) > 100") ;
Collection pers = (Collection) q.execute() ;
```

Dans un même ordre d'idées, les méthodes toLowerCase(), toUpperCase(), indexOf(), et substring() n'ont pas été oubliées cette fois ci (l'exemple qui suit utilise une requête paramétrée) :

```
Query q = pm.newQuery (Personne.class, "firstname.toLowerCase() == name");
q.declareParameters("String name");
Collection pers = (Collection) q.execute("marie");
```

Autre nouveauté très appréciable, c'est l'enrichissement des métadonnées JDO par des requêtes nommées qui évitent d'embarquer les requêtes directement au sein du code Java. Désormais, on peut créer une instance " Query " à l'aide d'une nouvelle méthode :

```
Query newNamedQuery(Class cls, String nomRequete)
```



JDO 2.0 implémente également une fonction " Attach / Detach " pour détacher les instances d'un PM (PersistenceManager) donné afin de les attacher à un autre PM après les avoir par exemple sérialisées puis transférées à un autre tiers.

Enfin, notez également que JDO 2.0 supporte la persistance des interfaces comme pour n'importe quelle autre classe, ce qui n'était pas le cas avec JDO 1.0.

■ Marie Varandat

Découvrir la nouvelle API de synchronisation de Java 5

PRATIQUE



Java 5, alias Tiger est riche de nouveautés bienvenues. Parmi elles, l'API de synchronisation, dite encore API de concurrence vous permettra d'alléger votre travail. Découvrons la, ensemble.

La programmation générique qui apparaît avec Tiger fait beaucoup parler d'elle à juste titre, car elle apporte au langage un peu de cette expressivité qui lui faisait si cruellement défaut. Les génériques tenant la vedette, d'autres nouveautés ne retiennent sans doute pas autant l'attention qu'elles le mériteraient, et parmi elles, l'API de concurrence. Lorsque Java est apparu sur la scène informatique, ses concepteurs mettaient entre autres en avant qu'il s'agissait d'un langage facile à utiliser dans un contexte multi-thread. De fait, Java simplifiait la programmation concurrente, mais il n'empêche qu'une application comportant un nombre important de threads restait difficile à organiser. La nouvelle API apporte des solutions à ce problème. Si question concurrence, Java n'approche pas un langage comme Erlang, loin s'en faut, nous pensons que cette API sera néanmoins une auxiliaire très précieuse dans de nombreuses situations. L'API est constituée d'interfaces et de classes réparties en 3 paquetages. Le plus volumineux, java.util.concurrent contient des classes pour la synchronisation des threads à haut niveau et des conteneurs supportant les accès concurrents. Les paquetages java.util.concurrent.atomic et java.util.concurrent.locks contiennent quant à eux des objets de synchronisation de bas niveau. Dans cet article nous nous intéresserons essentiellement au premier paquetage. Comme d'habitude vous trouverez tous nos programmes d'exemple sur le Cd-Rom accompagnant le magazine.

1 L'interface Executor

Dotée d'un nom digne d'un mauvais film américain, cette interface à une seule méthode est au coeur de cette nouvelle approche de la programmation concurrente avec Java. Ses implémentations doivent procurer un découplage entre la soumission de nouvelles tâches et la création explicite de celles-ci. Ainsi grâce à elle nous n'écrirons plus :

```
new Thread(new RunnableTask()).start();
```

pour démarrer un thread, mais :

```
Executor executor = anExecutor;
executor.execute(new RunnableTask1 ());
```

Ceci ne signifiant pas forcément que RunnableTask1 va démarrer immédiatement, peut être parce que l'implémentation d'Executor opère des instantiations de threads planifiées dans le temps ou encore parce l'implémentation gère un pool de threads pour l'instant saturé. Immédiatement dérivée de Executor, vient l'interface ExecutorService. Cet Executor évolué comporte des méthodes pour renseigner sur son état ou pour cesser son activité, l'ordre d'arrêt se répercutant sur les tâches gérées selon une politique bien définie. Comme toujours en pareil cas, Java propose des implémentations par défaut des interfaces. Les objets sont alors instanciés par la fabrique d'objet Executors.

2 Un pool de thread

Pour nous familiariser avec tout cela, nous allons commencer par travailler avec une implémentation de l'interface `ExecutorService` qui gère un pool de threads :

```
import java.util.concurrent.*;

class ThreadForExecutor implements Runnable {
    private int compteur = 10;
    private int id;

    ThreadForExecutor(int id) {
        this.id = id;
    }

    public void run() {
        while(compteur != 0) {
            System.out.println(
                "Je suis le thread " + id + " : " + compteur);
            try {
                Thread.sleep(500);
            }
            catch (InterruptedException ex) {
            }
            compteur--;
        }
        System.out.println("Thread " + id + " termine");
    }
}

public class DemoExecutor {
    public static void main(String[] args) {
        boolean result = false;
        ExecutorService es =
            Executors.newFixedThreadPool(2);
        es.execute(new ThreadForExecutor(1));
        es.execute(new ThreadForExecutor(2));
        es.shutdown();

        try {
            result = es.awaitTermination(20000,
                TimeUnit.MILLISECONDS);
        }
        catch (InterruptedException ex) {
        }
        if(result)
            System.out.println("Fin normale des thread");
        else
            System.out.println("Timeout");
    }
}
```

Examinons ce code. Dans la méthode `main`, nous obtenons notre `ExecutorService` via la fabrique d'objet. La taille du pool est de 2. Nous lançons ensuite 2 threads qui démarrent immédiatement. Que se passe

t-il si nous lançons un troisième thread en ajoutant :

```
es.execute(new ThreadForExecutor(3));
```

et en dépassant ainsi la capacité du pool ? Tout simplement, le thread sera mis en queue et sera lancé dès qu'un emplacement se libérera dans le pool, c'est à dire dès qu'un des deux premiers threads sera terminé. Ensuite nous bloquons le thread principal (ici le corps de main) avec l'invocation de `awaitTermination` qui attendra, soit que tous les threads du pool se terminent, y compris un éventuel thread mis en queue, soit que le délai d'attente spécifié soit dépassé. L'invocation de `awaitTermination` peut être précédée de l'invocation de `shutdown`. Dans ce cas il ne serait plus possible de mettre un thread en queue derrière cette invocation, ni même de lancer un thread "normalement" c'est à dire quand le pool a un emplacement libre. Une exception serait levée dans ce cas :

```
es.shutdown();
try {
    es.execute(new ThreadForExecutor(3));
}
catch (RejectedExecutionException ree) {
    System.out.println("le pool n'accepte plus de thread");
}
```

Autrement dit, l'invocation de `shutdown` annonce la mort du pool qui surviendra effectivement dès que toutes les tâches seront terminées. Il est possible de hâter la fin du pool avec `shutdownNow`. Dans ce cas, tout thread qui n'aurait pas encore physiquement démarré, un thread dans la queue autrement dit, ne sera jamais exécuté. Vous trouverez sur le Cd-Rom un autre programme de démonstration, `DemoExecutorPool` qui simule le comportement d'un serveur. Ce programme est inspiré du serveur de socket de la Javadoc. Vous saisissez une touche (suivie de [return]) pour démarrer un thread, autant de fois que vous voulez. Si le pool sature, les threads sont mis en queue. Le programme s'arrête avec la saisie de la touche 'q' et la terminaison de tous les threads.

3 L'interface `ScheduledExecutorService`

Cette interface permet de planifier le lancement de threads à partir d'un délai initial. En outre, le lancement peut être renouvelé sur la base d'un intervalle de temps fixe ou d'un intervalle de temps commençant à la fin de l'exécution du thread. L'exemple ci-dessous crée un pool de deux threads. Le premier démarre une seconde après sa création et le second, trois secondes après.

```
import java.util.concurrent.*;

class ThreadForScheduler implements Runnable {
    // code semblable à l'exemple précédent
}

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

```
ScheduledExecutorService sched =
    Executors.newScheduledThreadPool(2);
sched.scheduleAtFixedRate(new ThreadForScheduler(1, 100),
    1000, 1000, TimeUnit.MILLISECONDS);
sched.scheduleAtFixedRate(new ThreadForScheduler(2, 200),
    3000, 3000, TimeUnit.MILLISECONDS);
try {
    Thread.sleep(30000);
}
catch (InterruptedException ex) {
    ex.printStackTrace();
}
System.out.println("Arrêt de la demo demande");
sched.shutdown();
while(!sched.isTerminated()) {
    try {
        Thread.sleep(100);
    }
    catch (InterruptedException ex2) {
    }
}
System.out.println("Demo terminée");
}
```

Le programme tourne pendant 30 secondes pendant lesquelles les threads sont relancés toutes les secondes pour le premier et toutes les trois secondes pour le deuxième.

L'arrêt est demandé par notre méthode shutdown, comme expliqué plus haut. Aucun thread ne sera relancé après cela. Nous nous assurons du moment exact de cet arrêt, via l'invocation de isTerminated.

Nous avons parlé de temps et délais, mais il est clair qu'il s'agit ici d'ordre de grandeur.

En pratique, les résultats peuvent fluctuer, notamment en fonction de la charge du système d'exploitation hôte. On ne fera pas de programmation en temps réel avec ces classes.

4 Loquets de service

Les classes étudiées jusqu'ici, malgré leur intérêt évident, ne permettent pas de s'assurer du moment exact de la fin d'un thread ou d'un pool de thread. Pour répondre à ce besoin, Tiger met à notre disposition des classes de synchronisation. La première que nous étudions est CountdownLatch. Ce "loquet compte à rebours" permet de mettre en attente tous les threads qui ont invoqué sa méthode await. En attente de quoi ? Que d'autres threads aient fini de décrémenter son compteur via sa méthode countDown. Quand ce compteur arrive à zéro, les threads bloqués par await reprennent leur cours. Exemple :

```
import java.util.concurrent.*;

class ThreadForCountDownLatch implements Runnable {
    private int id;
    private int tempo;
    private CountdownLatch signal;
}
```

```
ThreadForCountDownLatch(int id,
    int tempo, CountdownLatch signal) {
    this.id = id;
    this.tempo = tempo;
    this.signal = signal;
}

public void run() {
    int compteur = 5;
    while(compteur != 0) {
        System.out.println("Thread " + id + " : au travail");
        try {
            Thread.sleep(tempo);
        }
        catch (InterruptedException ex) {
        }
        compteur--;
    }
    signal.countDown();
    System.out.println("Thread " + id + " : loquet libéré");
}
```

```
public class DemoCountDownLatch {
    public static void main(String[] args) {
        CountdownLatch loquet = new CountdownLatch(2);
        ExecutorService es = Executors.newFixedThreadPool(2);
        es.execute(new ThreadForCountDownLatch(1, 200, loquet));
        es.execute(new ThreadForCountDownLatch(2, 700, loquet));
        System.out.println("Le thread principal attend la libération du loquet");
        try {
            loquet.await();
            System.out.println("Fin du travail des threads");
            System.out.println("Reprise du thread principal");
            System.exit(0);
        }
        catch (InterruptedException ex) {
        }
    }
}
```

Cet exemple crée un loquet avec une valeur initiale de compteur de 2. Chacun des threads le décrémente une fois, après quoi le thread en attente des autres, ici la fonction main, reprend son cours. Deux remarques. D'abord un CountdownLatch ne peut être réutilisé une fois que son compteur est tombé à zéro. Ensuite, vous remarquerez un appel à System.exit(0) dans notre exemple. Sans lui, le programme ne rend pas la main. Il semblerait que nous ayons là un petit bug. Notre JDK est une 1.5.0-b64 et nous avons observé le même comportement sous Windows et Linux. Rien de très gênant car le thread de main reprend bien son cours comme prévu. Le phénomène n'a d'incidence que sur la terminaison de l'application. Le problème se produit exactement de la même façon si await est invoqué en dehors de la méthode main (cf. DemoCountDownLatch1 sur le Cd-Rom).



5 Une barrière cyclique

La classe `CyclicBarrier` est une variation de la précédente. On l'utilisera lorsque des threads devront s'attendre les uns les autres. Le terme 'Cyclic' signifie que contrairement à `CountDownLatch`, l'objet peut être réutilisé après que tous les threads aient atteint la barrière. En outre, lorsque l'on crée la barrière, on peut donner au constructeur un objet `Runnable` qui sera lancé une fois que tous les threads ont atteint leur but et avant que `await` ne rende la main. L'exemple `DemoCyclicBarrier` sur le Cd-Rom illustre ceci. Trois threads s'attendent mutuellement pour incrémenter une valeur. Quand le travail est terminé, l'application peut récolter le résultat final. Il est possible de savoir quel thread parvient en premier à la barrière. Pour notre exemple, `DemoCyclicBarrier.java` (sur le Cd-Rom), trois threads sont manipulés. Le premier est classé au rang 2 par l'objet `CyclicBarrier`, le suivant au rang 1, et le dernier au rang 0.

6 L'échangeur

Cette classe permet à deux threads de se passer une valeur comme deux joueurs de ping-pong se renvoient la balle. Dans cet exemple (`DemoExchanger.java` sur le Cd-Rom) on note l'apparition de la programmation générique. Un échangeur n'est rien d'autre, vu de l'extérieur, qu'un conteneur. On note aussi dans cet exemple que l'auto-boxing (par exemple `Integer compteur = 0;`) autre nouveauté bienvenue de Java 1.5, allège l'écriture du code.

Dans cet exemple un thread incrémente un compteur et l'autre le décrément, alternativement. Lorsqu'une certaine valeur de compteur est atteinte, le thread en cours échange cette valeur avec son alter ego. Cet échange a pour effet de le mettre en attente, tandis que l'autre thread démarre. L'autre thread procède de même.

Lorsqu'il échangera la valeur requise, il se mettra automatiquement en attente et réveillera le premier thread en lui passant cette valeur. Et ainsi de suite selon un mécanisme de bascule. Cet exemple boucle à l'infini, ce n'est pas un bug ;-)

7 Les valeurs du futur

L'interface `Futur`, ou encore l'interface `ScheduledFutur`, est un bienfait rendu possible par la généricité de Java. `Futur` encapsule le résultat, de type quelconque, d'un calcul asynchrone. Son implémentation par défaut, la classe `FuturTask`, encapsule un thread qui démarrera immédiatement ou de façon planifiée, selon l'`ExecutorService` et l'interface employés. Dans tous les cas, la méthode `get` de la tâche permet d'attendre de façon bloquante, éventuellement pendant un laps de temps défini, le résultat du calcul. Exemple :

```
import java.util.concurrent.*;

class CompteurDuFutur implements Callable<Integer> {
    Integer compteur;

    CompteurDuFutur(Integer compteur) {
        this.compteur = compteur;
    }
}
```

```
public Integer call() {
    while(compteur < 10) {
        try {
            Thread.sleep(1000);
        }
        catch (InterruptedException ex) {
        }
        System.out.println(
            "Compteur du futur presentement au boulot");
        compteur++;
    }
    return compteur;
}

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

        Integer resultat = 0;
        ExecutorService es = Executors.newFixedThreadPool(1);
        FutureTask<Integer> future =
            new FutureTask<Integer>(new CompteurDuFutur(0));

        System.out.println("Je lance le compteur du futur");
        es.execute(future);
        System.out.println("J'attends la valeur du compteur du futur");
        try {
            resultat = future.get();
        }
        catch (ExecutionException ex) {
        }
        catch (InterruptedException ex) {
        }
        System.out.println(
            "Le compteur du futur vaut " + resultat);
        System.exit(0);
    }
}
```

Cette fois la programmation générique est très présente. Un thread encapsulé par `FuturTask` n'implémente plus `Runnable`, mais la nouvelle interface générique `Callable`. On notera encore la présence de `System.exit(0)`, car nous avons rencontré ici exactement le même problème que celui signalé pour `CountDownLatch`.

La nouvelle API de Java 1.5 ne nous a pas encore livré tous ses secrets. Ses locks et sémaphores ainsi que ses conteneurs synchronisés sont également dignes d'intérêt. Toutefois, nous avons constaté, par ces quelques exemples, que la programmation asynchrone en Java sera grandement facilitée par cette nouvelle librairie. A coup sûr, les applications y gagneront une meilleure organisation, le code une bien meilleure clarté, et le programmeur beaucoup de temps, denrée ô combien chère et précieuse.

■ Frédéric Mazué - fmazue@programmez.com

Keyrus : une irrésistible ascension



Avec Micropole Univers et Business & Decision, Keyrus est l'une des principales SSII spécialisée sur le décisionnel. Créée en 1996, la société a réalisé en 2004 un chiffre d'affaires de 52,3 M€, en hausse de près de 13%, pour un effectif de 550 personnes. Elle vise les 100 M€ en 2007.

Dans une première phase, entre 1996 et 1999, Keyrus s'est développée par croissance organique. Entre 2000, date de l'introduction en bourse, et 2003, elle a procédé au rachat d'une dizaine de sociétés. Après une pause dans ses acquisitions en 2004, elle compte à nouveau faire quelques opérations de croissance externe, soit très ciblées sur des niches technologiques, soit de taille plus importante, voire de sa taille.

Trois pôles d'activité

Le métier " historique " de Keyrus est la business intelligence. Un segment porteur à l'heure des projets de rationalisation de reporting. Un marché en croissance régulière, de l'ordre de 15% par an " depuis quelques années et pour encore quelques années ", souligne Eric Cohen, P-DG de Keyrus.

Le second pôle, e-business, est issu de l'activité d'ingénierie dans les technologies client-serveur. Keyrus a ensuite connu la vague des webagencies avant de se recentrer sur les aspects d'architecture, notamment autour de des plates-formes Open Source et, depuis deux ans, de .net.

Eric Cohen indique vouloir se renforcer, en se développant par exemple sur certaines niches, comme l'EAI ou la voix sur IP. Ce pôle e-business " repart plutôt bien ", indique Eric Cohen, autour de projets comme les portails, ou touchant à l'intégration de flux de données et d'optimisation de la circulation de l'information. Il note également des projets autour des télécoms et de la mobilité.

Le troisième pôle, plus atypique, est constitué de Cyborg, spécialisé sur le mid-market, c'est-à-dire les PME réalisant un chiffre d'affaires compris entre 20 et 100 M€, pour lesquelles elle intègre les solutions de gestion de Sage, Adonix et Sap BusinessOne. Cyborg réalise un chiffre d'affaires de 18 M€ pour un effectif de 170 personnes.

Une approche verticalisée

Depuis plusieurs années, Keyrus privilégie les approches sectorielles, en ayant par exemple développé une offre packagée à destination des collectivités locales, ou encore dans le



domaine de la santé, en intégrant les problématiques métier et celles du reporting réglementaire au ministère de la Santé. Dans le secteur des télécoms, Keyrus travaille avec SFR et France Telecom, et a capitalisé son savoir-faire fonctionnel sur les problématiques de gestion de la relation client (CRM) ou de facturation (billing). Des approches également déclinées,

sur le métier e-business, dans les secteurs bancaires ou pharmaceutiques. Une double lecture métier et technologique qu'Eric Cohen souligne privilégier depuis 1998.

Une centaine de recrutements en 2005

Sur la partie Business intelligence, Keyrus recrute des consultants juniors, par exemple titulaire d'un DESS en Systèmes d'information d'aide à la décision, et les forme, avec ses partenaires, afin de les certifier. Elle recherche également des profils plus confirmés sur certains métiers, par exemple des spécialistes de gestion de risque dans le secteur bancaire pour des offres autour de Bâle II, toujours avec un angle décisionnel.

Sur le pôle e-business, Keyrus recherche des architectes et directeurs de projets, principalement sur les technologies .net et surtout des profils seniors. Eric Cohen relève qu'il y a de plus en plus d'appels d'offres sur des sujets comme la tierce maintenance applicative et la tierce recette applicative. Sont donc recherchés des chefs de projet ayant déjà une expérience de TMA ou de TRA.

Eric Cohen souligne l'effort accompli en termes de gestion des équipes et de fidélisation. Keyrus a mis en place des techniques de suivi des collaborateurs. Les plus jeunes ont un " parrain " au sein de l'entreprise. Un effort particulier est apporté à la formation et la communication interne, afin de " développer un sentiment d'appartenance ". Fin 2003 a été créé un club de managers Keyrus, qui se réunit trimestriellement pour " partager les bonnes pratiques ". Eric Cohen

explique que la société s'étant développée très vite, avec beaucoup d'opérations de croissance externe, il est nécessaire d'avoir " un socle et un discours commun " à diffuser aux équipes. Une initiative dont il dit ressentir les effets dans la dynamique de l'entreprise.

■ Carole Pitras

Les Ecoles fleurissent à Cergy

Nous vous avons dressé il y a quelques mois le portrait de l'EISTI. Cergy est décidément une mine en matière de technologies de l'information : A côté de l'EISTI, deux autres écoles forment de futurs informaticiens : d'une part l'ENSEA, une école d'ingénieur généraliste, publique, qui possède notamment des options informatique & systèmes, et réseaux & télécommunications, d'autre part, l'ITIN, spécialisée dans l'apprentissage, et qui possède notamment une formation de chef de projet informatique.



ENSEA

Ecole d'ingénieur généraliste en électronique, l'ENSEA délivre deux types de diplômes :

- Diplôme d'ingénieur de l'ENSEA, sur trois ans, en formation initiale ou continue, après admission sur concours au niveau Bac+2 ; profil : études, recherche & développement (effectif par promotion : 200 élèves ingénieurs) ;
- le Diplôme d'ingénieur des techniques de l'industrie de l'ENSEA, spécialité électronique & informatique industrielle (dit cycle "Iseea"), sur trois ans en formation continue

et en formation initiale par la voie de l'apprentissage, après sélection au niveau Bac+2 ; profil : conduite de projet, vie des affaires & marché industriel (effectif par promotion : 18 apprentis et 12 stagiaires en formation continue).

Les deux premières années sont consacrées à la formation de base de l'ingénieur, complétée par une option en dernière année. L'objectif de la formation indique Michel Leclerc, responsable du département informatique et techniques numériques de l'ENSEA, est de permettre au diplômé d'être "immédiatement adaptable au milieu industriel". Pour cela, est

volontairement privilégiée une formation avec un spectre relativement large, mais qui rend les élèves très adaptables aux nouvelles technologies. L'enseignement de l'informatique commence dès la première année, avec une introduction à la programmation en C, retenu pour sa syntaxe commune avec d'autres langages. En deuxième année est étudiée la structure de données.

En troisième année du cycle ingénieurs, l'option informatique & Systèmes, auparavant consacrée à l'informatique industrielle, vise à : "permettre aux élèves de compléter leur formation d'ingénieur électronicien par une solide formation en informatique, leur donnant une possibilité de gérer et de maîtriser des projets et des applications en informatique industrielle. La première partie des enseignements est consacrée à l'étude des architectures de machines, aux systèmes d'exploitation et aux communications entre machines. La seconde partie est consacrée à l'étude et à l'expérimentation de nouvelles techniques dans les domaines des sciences cognitives, du traitement d'images et de la reconnaissance des formes."

Autre option faisant largement appel à l'informatique, celle consacrée aux réseaux et télécoms. Conception de réseaux informatique, sécurité, administration de réseaux y sont étudiées. Côté informatique, cette option privilégie le monde Java. Michel Leclerc souligne la "forte composante expérimentale" de l'enseignement, avec de nombreux travaux pratiques et une approche par projet. Le dernier semestre est d'ailleurs consacré au Projet de

Portrait

L'école forme de bons généralistes

Sylvain Rambault, promotion 86 de l'ENSEA, option informatique industrielle, qu'il avait intégrée après un DUT. Il est aujourd'hui responsable informatique Amec-Spie Ile-de-France Nord-Ouest. Après l'ENSEA, il a intégré une SSII. Il juge que "c'est une bonne façon de compléter son diplôme", même si cette expérience a été assez moyenne. Il a enchaîné avec une autre SSII, où il a eu des missions plus intéressantes et plus variées, chez Sagem, Renault, Thomson puis Alcatel. Il intègre ensuite Alcatel, dans la filiale fibre optique industrielle, où il a passé dix ans et suivi toute l'informatique, des outils de gestion fournis par le siège, au développement de spécifiques pour suivre la fabrication et les stocks de fibres optiques. Il encadre alors une équipe d'une dizaine de personnes. Il rejoint ensuite une autre entité, Alcanet, dans le cadre de la refonte de l'informatique du groupe ; en 2002 il intègre Amec Spie, puis en 2004 devient responsable informatique d'une filiale.

Selon Sylvain Rambault, l'ENSEA forme de "bons généralistes". Il estime que le point fort de la formation est de pouvoir suivre les évolutions technologiques, sans forcément les maîtriser, et de "comprendre les problèmes connexes aux points que l'on traite". Sur la partie développement, il souligne "qu'avoir beaucoup développé à un niveau proche du processeur permet de rentrer dans le cœur de la machine", et que la "formation est suffisamment ouverte pour s'adapter à tous les langages". Il fait régulièrement appel à des stagiaires de l'ENSEA, et remarque qu'ils "raisonnent bien" et que "la formation reste de qualité".

fin d'études à temps plein dans l'industrie. Sa durée est comprise entre quatre mois et demi et huit mois et demi.

L'ITIN forme des chefs de projets

Née en 1988, l'école supérieure d'informatique, réseau et télécom s'est, dès l'origine, spécialisée dans la formation en alternance et l'apprentissage, explique Alain Gourdin, directeur opérationnel. Dépendant de la chambre de commerce et d'industrie de Versailles, et donc à ce titre gratuite, elle forme d'un niveau bac à Bac +5, avec une majorité de Bac+4. Parmi les formations proposées, celle de chef de projet informatique, qui met naturellement l'accent sur la mise en situation et la réalisation de projets pour des organismes ou des associations par exemple. L'objectif est triple, précise M. Gourdin : les diplômés doivent maîtriser les architectures techniques (réseaux, OS supervision) et les applicatifs (ERP, bases de données), ils doivent savoir analyser un problème, modéliser une solution, développer une application, la tester et la déployer à grande échelle, et enfin, ils doivent avoir un esprit d'entrepreneur, qui connaît le modèle économique d'une société. A cette fin, des séminaires sur la création d'entreprises ou d'associations sont proposés aux élèves. Cette formation se déroule sur deux ans et chaque année, l'ITIN forme 150



L'école supérieure d'informatique, réseau et télécom.

chargés de projets. L'admission se fait sur dossier, test et entretien. Chaque année, l'ITIN reçoit 1000 dossiers, provenant essentiellement de titulaires de DUT ou de licences, en sélectionne 300 pour leur faire passer tests et entretiens, et en retient 150. A noter que les entreprises, qui vont par la suite accueillir les apprentis, participent à l'entretien. Le panel d'entreprises accueillant ces apprentis est large, de l'entreprise unipersonnelle à l'éditeur ou à l'entreprise industrielle. Peu de SSII en revanche, précise Alain Gourdin, ces dernières ayant un peu de mal à gérer l'organisation en alternance.

Alain Gourdin souligne privilégier l'apprentis-

sage en mode projet et une approche rigoureuse : sont ainsi prises en compte la qualité des livrables et la qualité du processus de développement. La formation comprend 1000 heures de cours, travaux pratiques et travaux dirigés : l'apprenti passe deux mois en entreprise, puis deux mois à l'école. Sont proposées sept dominantes, de la sécurité des réseaux à l'administration de systèmes et applications. Côté technologies, Alain Gourdin indique essayer d'équilibrer entre le monde Open Source et celui des éditeurs propriétaires. Jusqu'ici, l'Open Source était privilégié sur les couches basses : sécurisation des sites web par exemple. Cela est en train de changer, avec le développement de couches " hautes ", comme les ERP, en Open Source. Il souligne toutefois vouloir montrer aux étudiants les deux aspects, dans un souci de développer leur capacité d'adaptation. Il remarque un retour vers le développement, après les années 90 marqué par les ERP, et ceci, notamment en raison du développement des terminaux mobiles, qui change le marché de l'appli-

catif, et induit de nouvelles problématiques comme la sécurité, qui impacte la conception des applications, ou amène à s'attacher davantage aux problématiques d'ergonomie et d'interface homme/machine. Autant d'évolutions prises en compte dans l'enseignement : Alain Gourdin souligne que 15% du programme est changé chaque année.

Un des projet de l'ITIN est de dupliquer ses formations, de franchiser son diplôme : c'est déjà le cas à Meaux, et prochainement à Laval. En l'adaptant au tissu économique local : informatique industrielle à Meaux, banque/assurance à Laval.

■ Carole Pitras

TOUT sur le code !



Steve MC CONNELL • 912 p.
59 € • ISBN : 210 048753 1

Ce manuel, ressource inépuisable, fournit toutes les méthodes et les "meilleures pratiques" aux professionnels du développement et de la programmation pour optimiser leur travail au quotidien.

Que vous soyez débutant ou expert, vous trouverez toutes les clés pour écrire du code efficacement.

Au sommaire :

- Planifier un projet.
- Appliquer la bonne technique de développement au bon logiciel.
- Utiliser efficacement les données.
- Trouver les erreurs.
- Gérer les étapes de développement.

Egalement disponible



Ecrire du code sécurisé
M. Howard et D. Leblanc
736 p. • 59 €
ISBN : 210 006 750 8

Microsoft
Press
La maison d'édition de Microsoft

XML

Entre l'universel et l'incompatible

Révolution ou pas, XML a réussi en quelques années à s'imposer comme une technologie quasi universelle dans les outils et logiciels. Ce meta-langage devient même une marque de sérieux. Les éditeurs aiment à dire que leurs solutions fonctionnent avec, ou sur XML. Dès que l'on invente un nouveau langage, un nouveau standard, il est de bon ton de dire qu'il utilise XML. Si XML est bien une technologie pratique et souple, les dérives actuelles inquiètent.



Pour beaucoup, XML est l'esperanto idéal de l'informatique. Il sert avant tout à faciliter les échanges entre partenaires. Il permet aussi de mieux gérer la diversité des données et des applications en utilisant un transport et un format unique. XML comme format d'échange n'est plus à remettre en cause. C'est d'ailleurs sa première fonction. Si XML est un format structuré, il a l'inconvénient d'être verbeux. Le volume de ce verbiage est 3-4 fois plus important que les données transportées. C'est une de ses limites. C'est pour cela, qu'utiliser XML comme format de stockage à grande échelle dans un SGBD n'est guère conseillé pour des questions de place, de traitement et de performances. Il faut bien distinguer l'échange, le transport, le stockage et le format des données.

Utilisez avec discernement XML

Le risque avec XML est de croire qu'il peut tout faire, dans tous les cas et pour toutes les fonc-

tions. Non, cette approche est vicieuse et dangereuse. Dangereuse, car les risques d'un effacement des performances, une saturation de bande passante, sont prévisibles, sans compter les manipulations à réaliser et à implémenter. Vicieuse, car on biaise alors l'utilisation du meta langage. Il faut savoir où et

comment utiliser XML dans son application, son environnement applicatif. En tant que format d'échange, format de données, XML vous fournit une intégration, une interopérabilité de haut niveau (même si des tests d'interopérabilité demeurent indispensables). En Web Services, ou dans un bus applicatif, XML

Macromedia et son MXML

Le player Flash implémente un parser depuis la version 5 et il est possible d'attacher un fichier XML comme source de données à un document Flash. Le développeur peut préciser la structure et/ou le DTD utilisé. Par contre, Dreamweaver ne permet pas (encore) cela. Chez Macromedia, XML est très visible avec la technologie de client riche, Flex. XML (MXML) est utilisé en qualité de descripteur d'interface comme XUL ou XAML. Quand on crée un document Flex avec l'outil Flex Builder, l'interface créée est en MXML (le fonctionnel étant codé en ActionScript). C'est à partir du fichier MXML que le serveur Flex génère le fichier SWF (fichier Flash). Il n'est pas possible d'utiliser un fichier de description XUL ou XAML dans Flex à la place de MXML, même si, cela aurait un sens pour le développeur. Sur la diversité des dialectes, Macromedia adopte une position prudente et pragmatique : " on prend le parti de ne pas prendre parti " dixit Florent Pajani (Macromedia France). " Si avoir des spécifications est une bonne chose, avoir des utilisateurs et des porte-drapeaux est encore mieux... "

semble incontournable. En intégration, la communication inter application et interdonnées peut se réaliser en XML. Si vous avez des données de petites tailles à stocker, XML peut, dans ce cas, être intéressant. Pour de gros volumes, passez plutôt par de la transformation en E/S de la base. En terme de performance, l'impact sera limité.

XML permet une forte abstraction dans les échanges, les données et les applications. Vous pouvez donc garder un environnement applicatif / données hétérogène, tout en appliquant une couche d'échange et de transport parfaitement homogène. Par exemple, vous souhaitez mapper rapidement et facilement un formulaire InfoPath et une base de données. Le fait que SQL Server 2005 implémente nativement XML, le mapping se fait en toute transparence. Vous gagnez en temps de développement.

De plus, XML impose un découpage des couches et vous pouvez à loisir contrôler la bonne formation du document et même l'intégration des données. Ainsi, dans des échanges et traitement de données, vous pouvez inclure des mécanismes de contrôle des données contenues dans un fichier XML avant même son introduction dans un traitement applicatif.

XML mène-t-il à une perversion ?

Nous venons de voir les atouts de ce meta langage. Plusieurs points négatifs (hormis son verbiage) sont à noter. Si XML en tant que tel est interopérable, si l'implémentation respecte les spécifications, un réel souci vient de la

multiplication des dialectes et organisations de spécifications.

Depuis des mois, toutes les semaines, on annonce de nouveaux dialectes utilisant XML. Les éditeurs et organismes (W3C, OASIS pour ne citer qu'eux) multiplient les annonces et les publications. Comme XML permet de définir sa propre grammaire, il est aisé dans un projet, de créer sa propre spécification XML. On pourrait alors résumer ainsi : XML est la couche basse commune, et les dialectes se créent par métier, par besoins (= par marchés verticaux). Or, utiliser des dialectes fournit une rupture d'interopérabilité. Car bien souvent, les dialectes ne peuvent communiquer entre eux, sauf à mettre en place des transformations inter dialectes... Bref, une belle usine à gaz. Choisir un dialecte, c'est bien souvent, choisir un camp, un éditeur, avec tous les risques que cela induit.

Rich Client, descripteur d'interface, fichiers XML, etc.

Deux grands enjeux, en partie liés, de XML concernent le client riche et la description d'interface. Les dialectes XML en tant que descripteurs d'interface commencent à s'imposer, au moins dans l'esprit. Les plus connus sont MXML (Macromedia), XUL (Mozilla), XAML (Microsoft). Le but est de décrire l'interface utilisateur, via un dialecte XML. C'est à partir de ce dialecte, que l'interface est générée, soit statiquement, soit dynamiquement. L'avantage d'une telle architecture est que l'on découple l'interface de l'application, ainsi, on peut générer une interface spécifique pour chaque utilisateur, pour chaque type de terminal ! Une partie de l'offre client riche repose sur cette idée. Malheureusement, chaque dialecte de description est incompatible (du moins actuellement), il faut donc choisir la technologie.

■ François Tonic

XML et Microsoft

Depuis plus de six ans, Microsoft mise beaucoup sur XML. Depuis l'an 2000, l'éditeur implémente ce meta langage dans l'ensemble de ses outils, que ce soit côté serveur, ou côté client. L'éditeur ne considère pas XML comme trop verbeux. XML permet une séparation stricte des différentes couches. Dans SQL Server 2005, un des formats de stockage des données sera du XML, tout comme pour le requêtage (Xquery, Xpath). L'éditeur précise que l'utilisateur aura le choix du format et que le XML ne sera pas proposé par défaut. Sur XAML, Microsoft avance avec prudence, le langage et les outils n'étant pas encore réellement disponibles, même si certains outils sont déjà disponibles (notamment dans la communauté Open Source). S'il est indépendant des OS, il demeure lié à Windows, pour le moment.

XML et Software AG

Software AG, se proclamant " XML company ", a rapidement misé sur XML. Pour l'éditeur, le rôle le plus important de XML est l'échange de données, secondairement le stockage. " Dès que les données se déplacent, XML a sa place " explique Joel Milgram (Software AG). Pour le format de stockage XML, Joel Milgram a son idée sur la question, " cela a un intérêt dans certains cas, par exemple dans un bus XML, ou dans le messaging. On s'aperçoit aussi que XML se développe dans les aspects métiers. Sur les dialectes, on peut voir une sorte de babélisation. La tendance est d'inventer son propre vocabulaire. Le fondement demeure XML mais il n'est plus forcément interopérable. Il y a du mapping à faire. C'est un réel danger, même si les avantages du XML sont gardés. Nous n'incitons pas nos clients à réinventer un vocabulaire, il faut déjà voir ce qui existe."

webMethods mise aussi sur XML

L'éditeur d'outils d'intégration a très tôt pris le parti de XML, afin d'intégrer et de gérer la diversité. " XML est un format d'échange simple à comprendre. Dans le B2B, il faut un format d'échanges " précise Régis Mauger. Par contre, l'éditeur ne préconise pas systématiquement l'utilisation d'XML. " C'est de la responsabilité de l'architecte, du chef de projet de déterminer où XML est pertinent ". L'éditeur mise beaucoup sur l'interopérabilité du XML et son côté standard. Les solutions webMethods utilisent naturellement le format XML lors de la mise en place de Web Services.

XML et Adobe

Comme d'autres éditeurs, Adobe intègre XML dans l'ensemble de ses outils, même si cela ne se voit pas. Dans la partie serveur, XML est utilisé comme langage de script pour piloter les outils serveurs de l'éditeur. XML est même présent dans la structure des PDF via des schémas. Actuellement, l'éditeur travaille beaucoup sur les spécifications Xform. Si Xform définit ce que doit être un formulaire web, il est très incomplet. Adobe propose XFA (XML Form Architecture), en s'appuyant sur la société Accelio, rachetée il y a quelques mois. Pour étendre les possibilités de Xform. Pour pouvoir manipuler des informations PDF en XML, on dispose de XDP (XML Data Package). On encapsule alors un document PDF dans un document XML pour que les logiciels puissent les lire. Tout le monde ne lit pas forcément le PDF (notamment dans les outils GED)...

XML : il faut s'y mettre !

XML (eXtensible Markup Language, que l'on peut traduire par langage à balise extensible) est une sorte de HTML évolué permettant de définir de nouvelles balises et de constituer des documents grâce à ces balises. Du fait qu'il n'est ni figé, ni prédéfini, il peut être considéré comme un métalangage qui permet de créer de nouveaux langages, ou grammaires, par la mise en place de nouvelles balises.

Grâce à son extensibilité, XML permet de décrire n'importe quelle structure de données, et ce, quel que soit le domaine technique ou fonctionnel auquel on l'associera. Il structure et définit le vocabulaire et la grammaire des données qu'il représente (figure 1). Les balises XML décrivent donc le contenu et non la présentation (contrairement à HTML) et permettent de dissocier ces deux éléments. Cela autorise, par exemple, l'affichage d'un même document sur des applications ou des périphériques différents, sans qu'il soit nécessaire de créer autant de versions différentes du document que de représentations possibles.



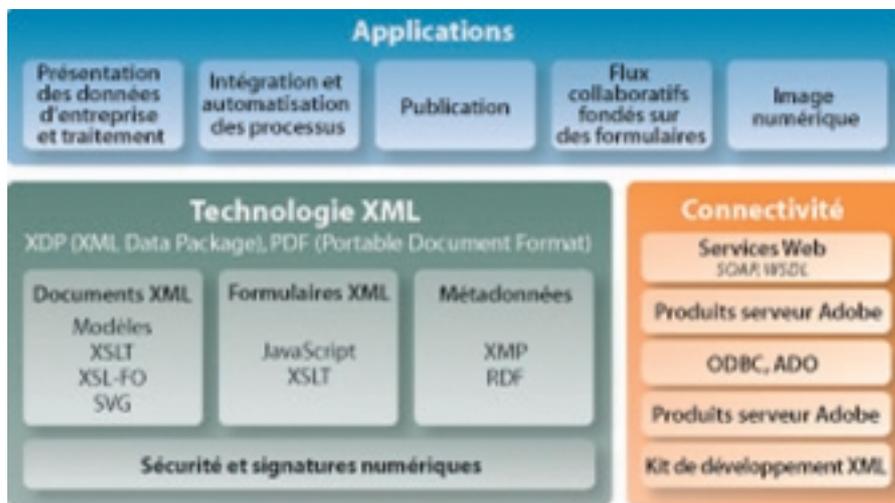
Figure 1 : exemple de fichier XML

L'état des spécifications

La norme XML a engendré de nombreuses spécifications et recommandations périphériques pour répondre aux diverses applications qu'elle a entraîné dans son sillage.

Parmi celles-ci, nous pouvons citer :

- XHTML : XHTML a pour objet de faire converger XML et HTML. XHTML emprunte des éléments des précédents travaux du W3C sur HTML 4, et peut donc être interprété par la quasi-totalité des navigateurs Internet existants.
- XLink (XML Linking Language), actuellement en version 1.0, permet d'insérer des éléments au sein de documents XML afin de créer et décrire des liens entre différentes ressources. Il se fonde sur XPath pour créer des structures capables de décrire aussi bien de simples liens hypertextes tels qu'on les utilise dans du HTML, que des liens plus sophistiqués (plus de deux fichiers).
- XML Encryption : l'objectif de la norme XML Encryption est de développer un protocole



permettant d'encrypter et de décrypter des contenus numériques (incluant des documents), ainsi qu'une syntaxe utilisable pour représenter le contenu encrypté et les informations permettant à l'application destinataire de le décrypter. XML Encryption est également un peu plus sophistiqué puisqu'il permet le transport d'éléments chiffrés au côté d'autres éléments en clair. Le résultat du chiffrement est ensuite formaté sous forme d'éléments XML.

- XML Key Management 2.0 est une spécification qui définit un socle d'architecture d'enregistrement et de distribution de clés publiques de chiffrement. Il repose notamment sur le protocole XKRSS, qui explique comment récupérer et enregistrer les informations relatives aux clés publiques d'un service Web, et sur le protocole XKISS, chargé de définir comment une application peut déléguer à un service le traitement des clés à des fins d'authentification et de confidentialité. On associe ce protocole à XML Signature et XML Encryption.
- XML Signature définit un ensemble de règles de syntaxe et de traitement adaptées à la topologie des documents XML et qui per-

mettent la signature partielle ou complète de documents XML.

- XPointer 1.0 est une spécification dont l'objet est de permettre la désignation d'un fragment de document XML, disponible en ligne, c'est-à-dire accessible depuis une URL.
- XPath 2.0 est une syntaxe (non XML) permettant de désigner une portion d'un document XML. Initialement créé pour fournir une syntaxe et une sémantique aux fonctions communes à XPointer et XSL.
- XQuery (XML Query) est un langage de requêtes permettant d'extraire des informations d'un document XML. Sur le plan sémantique il est proche du SQL et repose sur la syntaxe XPath pour adresser des parties spécifiques d'un document XML. Il se différencie de XPath par sa plus grande richesse fonctionnelle. Le groupe de travail XML Query (XQuery) se base aussi sur les travaux des prochains standards XML pour la recherche, mais aussi pour la sérialisation, la recherche full-text, la fourniture d'un modèle de données XML, etc.
- DTD : La DTD est un fichier (ou plusieurs fichiers utilisés simultanément) qui décrit de façon précise les balises spécifiant un type

d'éléments, ses attributs, et les relations qui les lient entre eux.

- XML Schema : Les schémas XML enrichissent le concept de la DTD, en permettant de définir des types pour les données. Par ailleurs, ils peuvent être édités et manipulés à partir de tout outil d'édition ou de traitement XML. Les schémas XML permettent de définir des grammaires et de partager des règles préalablement définies entre différents systèmes. Ils sont utilisés pour définir la structure, le contenu et la sémantique des documents XML.
- XSL-T 2.0 décrit les transformations de documents XML. L'objectif principal est la transformation vers un autre document XML ou un dialecte XML (XHTML, XSL-FO, etc.). Il autorise aussi les transformations vers d'autres types de documents, qu'ils soient au format texte ou au format binaire (bien que ce ne soit pas nativement prévu par la recommandation XSL-T). XSL-T repose sur XPath pour désigner un fragment d'une arborescence XML.
- XSL-FO (XSL Formatting Objects) est un vocabulaire de spécification des mises en forme de documents XML, indépendamment du support chargé de les représenter: écran, papier, périphériques nomades ou audio.
- XForms 1.0 est une spécification de formulaires Web destinée à être utilisée avec HTML, XHTML, WML, ou bien SVG, et accessible depuis une large variété de plates-formes, PC de bureau, ordinateur portable, assistant personnel, ou papier.

Manipulation de documents XML

Pour traiter et manipuler des documents XML, il est nécessaire d'avoir recours à un outil appelé parser (ou analyseur syntaxique) qui permet de parcourir un document et d'extraire ou modifier les informations qu'il contient (figures 2 et 3).

Il existe des implémentations de parsers XML pour les différents langages de programmation, ce qui autorise la manipulation de fichiers XML dans tous les types d'applications. Un parser XML peut être validant, c'est-à-dire qu'il permet de vérifier qu'un document est conforme à une DTD (ou à un schéma), ou bien

```

public static Document parseFile(String filename) {
    Document document = null;
    try {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        // Do not do validating parsing
        factory.setValidating(false);
        // Do not use that unfortunate namespace
        factory.setNamespaceAware(false);

        try {
            DocumentBuilder builder = factory.newDocumentBuilder();
            // Parse the XML into memory the Document
            document = builder.parse(new File(filename));
        } catch (Exception e) {
            System.out.println("Parsing error, line " + e.getLineNumber() +
                " = " + e.getMessage());
            // See the namespace exception, if any
            Exception a = e;
            if (e instanceof SAXException) {
                a = e.getMessage();
            }
            System.out.println("SAXException: " + a.getMessage());
            // See the namespace exception, if any
            Exception a = e;
            if (e instanceof SAXException) {
                a = e.getMessage();
            }
            System.out.println("SAXException: " + a.getMessage());
        }
    } catch (Exception e) {
        System.out.println("Exception: " + e.getMessage());
    }
    return document;
}
    
```

Figure 2 : exemple de manipulation XML depuis Java.

```

class XMLParser {
    public void parseFile(String filename) {
        try {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            // Do not do validating parsing
            factory.setValidating(false);
            // Do not use that unfortunate namespace
            factory.setNamespaceAware(false);

            try {
                DocumentBuilder builder = factory.newDocumentBuilder();
                // Parse the XML into memory the Document
                Document document = builder.parse(new File(filename));
            } catch (Exception e) {
                System.out.println("Parsing error, line " + e.getLineNumber() +
                    " = " + e.getMessage());
                // See the namespace exception, if any
                Exception a = e;
                if (e instanceof SAXException) {
                    a = e.getMessage();
                }
                System.out.println("SAXException: " + a.getMessage());
            }
        } catch (Exception e) {
            System.out.println("Exception: " + e.getMessage());
        }
    }
}
    
```

Figure 3 : exemple de manipulation XML depuis C#.

non validant. Le parser non validant se contente de vérifier que le document XML est bien formé, c'est-à-dire qu'il respecte au moins la syntaxe XML de base. Par ailleurs, il existe deux familles de parsers XML qui se distinguent par la façon dont ils traitent le document : DOM, et SAX.

DOM (Document Object Model) se base sur une approche hiérarchique du document XML. Un parser DOM construit une arborescence d'objets qui représente les éléments du document et propose des méthodes d'accès aux propriétés.

Il existe deux spécifications de DOM :

- DOM level 1 se divise en deux catégories :
 - Core DOM level 1 pour les documents de façon générale,

- HTML DOM level 1 qui ne retient que les méthodes applicables à HTML

- DOM level 2 qui adjoint des fonctionnalités de prise en compte des feuilles de style CSS dans la hiérarchie d'objets.

SAX (Simple API for XML), repose quant à elle sur un mode événementiel qui permet aux applications qui l'implémentent de réagir à des événements (ouverture ou fermeture d'une balise, ouverture ou fermeture d'un document, etc.) lors de l'analyse du document XML. Une application implémentant un parser SAX repose généralement sur des gestionnaires d'événements qui permettent de réaliser des opérations spécifiques lorsqu'ils rencontrent une balise donnée.

Une interface événementielle telle que l'API SAX permet de créer des événements à partir de la lecture du document. Ainsi, une application basée sur SAX peut gérer uniquement les éléments dont elle a besoin sans avoir à construire en mémoire une structure contenant l'intégralité du document.

Comparaisons de DOM et de SAX

DOM nécessite de reproduire en mémoire la hiérarchie représentant l'intégralité des éléments du document. Dès lors, pour des documents de taille importante, DOM devient très contraignant et très peu performant, car trop consommateur de ressources mémoire. A contrario, SAX, permet de ne manipuler

que ce qui est utile au sein du document XML, et ce, quel que soit le volume de ce dernier. Du côté des plates-formes techniques, Microsoft fait la part belle à XML avec son futur Indigo, qui offrira notamment un bus de messages XML pour l'échange de services Web. De son côté, Sun poursuit son travail sur les évolutions des bibliothèques XML pour Java (JAXB, JAXP, SAA, et JAXRPX), et de son Java Web Services Developer Pack.



■ **Mickael Maindron**
 mickael.maindron@neoxia.com
 Architecte chez NEOXIA, intégreur de systèmes spécialisés dans les nouvelles technologies de l'information.

XML : Comment un Français a changé la face de Microsoft



Si XML a changé le monde informatique, et encore ce n'est qu'un début, Microsoft a en quelques années fait sa révolution en prenant le pari du tout XML. Cette révolution radicale a été accomplie par la volonté de Bill Gates, mais surtout, grâce à un homme, un Français, Jean Paoli. Une histoire étonnante qu'on vous dévoile.

Jean Paoli est une figure de l'informatique française depuis plus de 20 ans. Il aime à parler de XML qu'il a fortement contribué à créer. Comment définir XML et sa genèse ? "Tout le monde utilise des applications, il faut de bons concepts. Or, il n'y avait pas de bon niveau d'abstractions. XML rencontre deux démarches : les données et le document." précise-t-il d'emblée.

Aux Ponts et Chaussées, promotion 1984, il a comme professeurs Bernard Lang et Gilles Kahn, président de l'INRIA. Celui-ci était à la pointe de la représentation des informations semi-structurées. Le projet Mentor est né pendant cette période.

Les débuts de Mentor

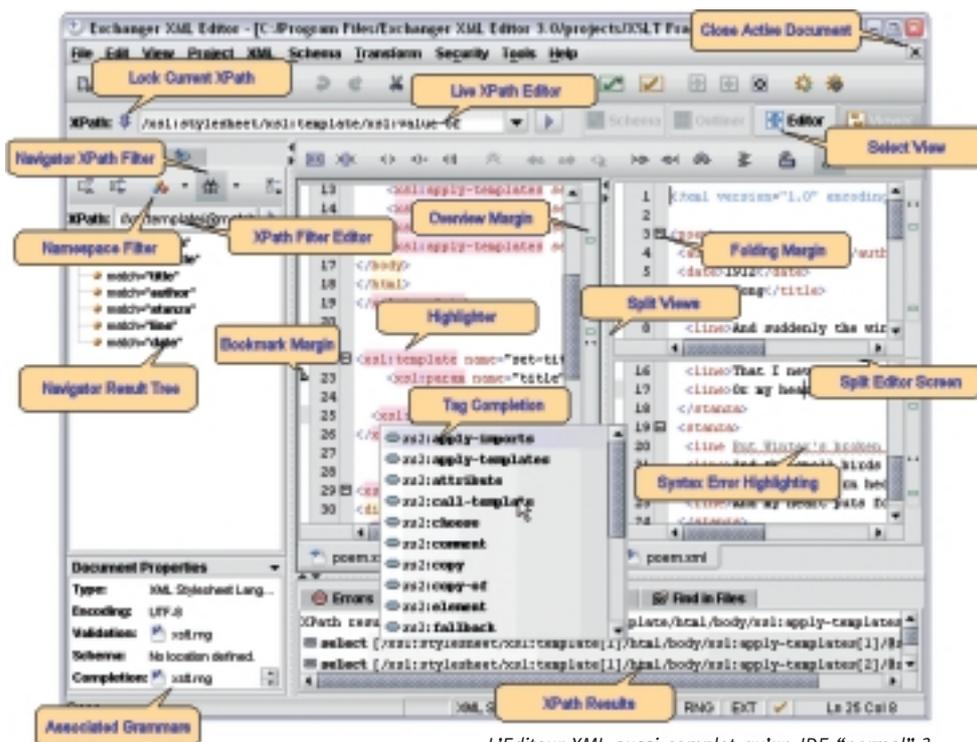
Après les Ponts, il part rejoindre la Sema dans les technologies avancées. Il s'agissait d'aider à mettre Mentor sur le marché. "C'était mon premier boulot". Nous sommes en 1986. "Il s'agissait de savoir comment aider les développeurs à développer des outils pour les langages de programmation. L'INRIA avait déjà participé à la création du langage ADA. On a alors découvert que Mentor n'était pas commercialisable en soi. On s'est aperçu que les mêmes idées pouvaient aider à créer des documents structurés. Vincent Quint, de l'INRIA Grenoble, et Irene Vatton du CNRS avaient conçu le prototype Grif servant à la création de documents.

"Le but était de faire de Grif, un véritable produit. Il fallait savoir ce que l'on gardait du prototype initial, ce que l'on devait changer. Dès le départ, on a rencontré un certain succès, au sein d'une startup de l'INRIA. Des personnes du DOD travaillaient sur le SGML, les normes

CALS. La Défense avait des besoins, car ils avaient des documents hétérogènes. Les Américains et les Européens avaient les mêmes besoins et ont poussé."

"L'information est semi-structurée : des données rigides d'une base et les notions de documents. Il fallait quelque chose au milieu pour pouvoir représenter des données et des documents et avoir une structure flexible. C'est la base du XML : la rencontre entre le document et les données. On était à peine 3 ou 4 à travailler sur l'industrialisation de Grif, dont Xavier Franck, Stéphane Querel, Bertrand

Melese. On commençait à avoir des contacts avec la Marine Nationale.". Tout commence à s'accélérer. Jean Paoli travaille au niveau européen dès 1986, sur des projets Esprit et Eureka. Peu après, la société Grif SA est mise sur orbite. "J'étais le directeur technique avec une équipe d'une vingtaine de personnes. On était alors parmi les leaders de l'édition SGML dans le monde. Comme c'est un petit monde, on se rencontre tous. Lors de rencontres et de conférences, on refait le monde à notre façon. Il s'agissait de prendre des informations de différentes plates-formes, de mettre les don-



L'Editeur XML aussi complet qu'un IDE "normal" ?

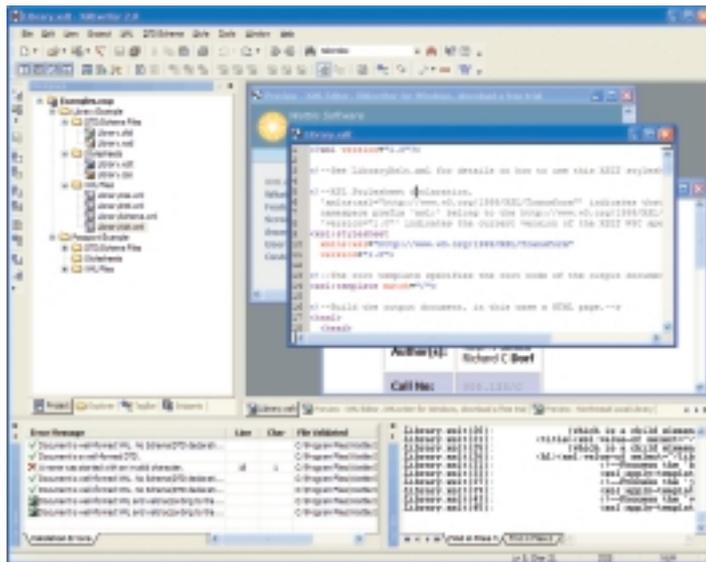
nées dans un document puis de le renvoyer. On savait que SGML était trop compliqué. Il n'existait qu'un seul parser dans le monde."

"C'est alors que l'on démontre Grif à l'inventeur du Web au CERN.". Au fur et à mesure des travaux et recherches, les prémices du W3C naissent à cette époque. "Les gens se faisaient confiance", confie Jean Paoli. La première fois où j'ai vu Internet pour le grand public, c'était en 1995 à San Francisco sur une pub Coca Cola, avec l'indication de leur site ! Je sentais qu'il était temps de démocratiser nos travaux." Fin 1995 se déroule une importante conférence qui aborde largement HTML, mais aussi SGML. J'y avais fait une conférence. Ce mois-ci, Bill Gates avait publié son manifeste pour le Web. Le soir à l'hôtel, on m'appelle, c'était un responsable de Microsoft. On parle. J'avoue avoir été intrigué."

Objectif Microsoft

Jean Paoli part alors pour Redmond, sur l'immense campus Microsoft. Il rencontre les équipes de développement. "Je m'apprêtais à prendre l'avion. On me dit alors d'aller voir quelqu'un à San Francisco, Adam Bosworth. On se rencontre à l'aéroport. Il me demande ce que je voulais faire" poursuit Jean Paoli. Le Français, assis par terre avec son interlocuteur, dit que le HTML n'est pas suffisant. Sur le web, il faut des données. "Viens nous l'expliquer me dit-il". L'objectif de cette invitation était de démontrer que la vision de Jean Paoli était réalisable et que cela servait à quelque chose. "En avril 1996, je pars donc pour les Etats Unis. Je rejoins l'équipe Internet Explorer de Microsoft, pour y développer mon idée, que HTML ne suffisait pas. Au départ, nous n'étions que quelques personnes, mais toutes très motivées et pragmatiques. Les 6 premiers mois, la journée je travaillais avec ma petite équipe de développeurs, et le soir, au téléphone, je travaillais avec mes amis de la communauté SGML à créer, au sein du W3C, le standard XML, avec un objectif prioritaire : simplifier SGML." poursuit Jean Paoli.

"Je donnais aux développeurs MS les spécifications nécessaires pour voir si on pouvait le



Aujourd'hui XML est partout et les outils se multiplient.

faire. Il fallait quelque chose de simple, car la technologie était destinée à des millions d'utilisateurs. Il fallait rapidement et simplement créer un fichier XML. Fin 1996, une grande date. Durant une conférence SGML, XML est présenté.

Qu'est que XML au bout du compte ? Jean Paoli le résume assez bien : "c'est la rencontre de la communauté SGML, de l'expérience semi structurée, de l'INRIA, et de Microsoft". Si en six mois, la majorité de la définition du XML était conçue, il fallait abattre un immense travail pour affiner tous les détails et le langage. Ce chantier durera presque 2 ans ! C'est alors que la technologie commençait à faire du bruit sur la toile. "Je leur ai dit que c'était de l'information et non de la présentation, donc il ne faut pas utiliser HTML. On a alors mis au point le CDF, le premier langage XML du réseau, le RSS d'aujourd'hui est son héritier ". CDF a vraiment aidé à populariser XML, à en trouver une utilisation à très grande échelle, d'autant que CDF s'est retrouvé au cœur d'un débat médiatique sur le " push ". Jean Paoli était aux anges...

Début 1997, l'implémentation du parser XML avait commencé. On soumet au W3C les premières briques. Chez Microsoft, les développeurs en parlaient en dehors de l'équipe IE. 1997 constitue l'année charnière. Microsoft proposait, avec la collaboration de partenaires extérieurs, les spécifications qui deviendront l'ensemble des standards XML : Microsoft travaillait sur les namespaces. En à peine une semaine, on avait créé la proposition initiale de XSLT. Puis tout s'enchaîne pour définir le

cœur du XML. Microsoft était partant. "Il fallait aller vite. On est allé voir Bill Gates. Il nous a donné son feu vert. En septembre 97 durant le Seybold Conference, tout le monde disait que XML était très bien, avec de bonnes intentions. Microsoft a alors présenté une implémentation XML dans IE. C'était IE 4 avec le parser XML." Microsoft était alors la première et seule grande société à avoir implémenté XML. Le monde informatique commençait à regarder XML d'une autre manière... A cette occasion, Bill Gates annonce " XML est une technologie qui bouleverse en profondeur le paysage de l'informatique ".

Netscape a suivi. Puis, "En 1998, nous sommes allés rencontrer IBM pour présenter et expliquer XML.". IBM s'y est mis ainsi qu'Oracle et tant d'autres. "Les gens ont eu initialement du mal à comprendre que Microsoft fasse du standard, soit a la pointe..." lâche Jean Paoli.

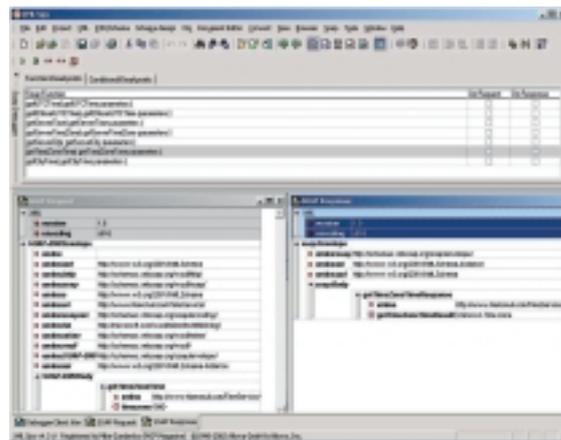
XML s'impose

Durant plus de 4 ans, de 1996 à 2000, XML avait été mis dans IE, SQL Server, Biztalk, etc. Les concurrents l'implémentaient aussi. C'est alors qu'un nouvel environnement commence à se dessiner à l'intérieur de Microsoft. Bill Gates annonce durant l'an 2000, la stratégie XML - Web Services ! C'est l'acte de naissance de .NET ! Il fallait aller du niveau serveur au niveau client. "On a commencé avec le back office, puis on s'est attaqué au front office, avec notamment Office précise Jean Paoli. "On a beaucoup travaillé sur XML au niveau serveur. Les gens n'ont pas compris ce que l'on faisait à l'époque. En 1999, j'ai rejoint l'équipe d'Office. Il fallait maintenant que l'on fasse un éditeur XML pour le grand public. C'est alors que nous avons inventé InfoPath. La conception du prototype a nécessité un an de travail. En 2001, Bill mise sur InfoPath." raconte Jean Paoli. Et le futur ? Jean Paoli se montre discret. "Je travaille sur XML, je suis un passionné des scénarios complets " end to end ", du partage de XML depuis le poste client, le PC, le téléphone jusqu'au serveur, en passant par le workflow. Je travaille avec un peu toutes les équipes à Microsoft. XML est le langage l'ouverture".

■ François Tonic

Panorama des outils

Désormais, la plupart des éditeurs intègrent des fonctions ou des outils XML à leur environnement de développement, serveur ou d'intégration. Dans la plupart des cas, inutile d'aller chercher une solution extérieure. Cependant, pour l'édition et la vérification de documents XML, pour réaliser des transformations ou créer des XSLT, mieux vaut utiliser un éditeur extérieur, souvent bien plus performant. Voici donc un petit panorama, loin d'être exhaustif, du marché.



Les éditeurs XML

Les éditeurs sont désormais de véritables IDE et/ou RAD dédiés à XML et souvent à d'autres dialectes (Xpath, XSLT, Xquery, etc.). Les éditeurs permettent de créer, gérer, valider, transformer rapidement et simplement des documents XML, des schémas.

Conglomerate	Open Source http://www.conglomerate.org	Editeur simplifié pour cacher la complexité de XML aux utilisateurs novices.
EditML Pro	Commercial http://www.netbryx.com	Génération automatique de DTD et XML Schema. Fonction completion, importation de données en XML, Support Unicode. Transformation XML en HTML.
Editix	Commercial http://www.japisoft.com/	Editeur et débogueur XSLT, vérification de la syntaxe à la volée. Transformation XSLT, XSL-FO, DocBook, moteur de recherche, gestion de projet, support de JAXP
Epic Editor	Commercial http://www.arbortext.com/	Edition XML – SGML. Possède un éditeur visuel, fonctions de localisation, environnement personnalisable
EWebEditPro+XML	Commercial http://www.ektron.com	Editeur visuel pour XML et XSLT, validation multicritères. Extensible.
Exchanger XML Editor	Commercial http://www.exchangerxml.com/	IDE visuel compatible Xpath. Support XML Schema, DTD. Transformation XSLT, Xquery, XSLFO. Analyseur WSDL. Fonctions de refactoring, navigation de code et de projet, fonction de completion.
JXMLPad	Commercial http://www.japisoft.com/	Composant 100 % Swing pour éditer des documents XML et XHTML. Pour un usage javaBeans. Fonction de completion, vérification syntaxe à la volée. Support unicode, Compatible Java 5.
Jaxe	Open Source http://jaxe.sourceforge.net/jaxe.html	Editeur de référence du monde Open Source. Configuration par schéma, écrit en Java. Fonctionne uniquement avec fichier XML et fichier de configuration, pas visuel
Morphon	Gratuit http://www.morphon.com/	Editeur visuel. Ecrit en Java. Support DTD et schéma XML. Extensible.
oxygen	Commercial http://www.oxygenxml.com/	Multi-plate-forme. Editeur de source, débogueur XSLT, vue par arborescence, validation DTD, Relax NG, validation par schéma XML, completion. Support Xpath. Génération de documentation
Stylus Studio 6 XML	Commercial http://www.stylusstudio.com	Editeur XML, XSLT avec débogueur. Outils de mapping et SOAP. Editeur de DTD et pour XHTML, compatible .NET. Fonctions pour Xpaths et import/export. Support de XQuery
Serna	Commercial http://www.syntext.com	Editeur visuel XML orienté entreprise. Support des formats de schémas courants, d'unicode et de la validation à la volée des documents.
TurboXML	Commercial http://www.tibco.com/	IDE XML pour gérer et développer des assets XML. Support des XML Schemas, DTD, XDR, SOX, Biztalk. Editeur de schémas.
xmloperator	Open Source http://www.xmloperator.net/index_fr.htm	Ecrit en Java. Propose une vue arborescente des nœuds du document. Importe schéma RELX NG ou DTD
XMLWriter	Commercial http://xmlwriter.net	Support de la transformation XSL, validations des documents, support des dernières spécifications W3C, fonction Code Snippets, conversion DTD vers XSD, gestion de projet.
Xlsmaker	Commercial http://www.xlsmaker.com/	Edition XSL pour gérer et construire du contenu web. Génération XSL et CSS à partir du XML. Outil de reporting.
xmImind	Commercial http://www.xmlmind.com/xmleditor/	Editeur de validation XML. Mixte XML données et XML documents. Extensible. Support des schémas, XSLT, Xpath, Xinclude, XHTML, multi-plate-forme
xmlcooktop	Gratuit http://www.xmlcooktop.com/	Editeur et IDE de conception XML, DTD et XSLT. Fonctions de tests et de debug. Formatage du XML via Tidy.

Les outils divers...

Les utilisateurs et les développeurs ont besoin d'outils pour manipuler rapidement et convenablement des documents XML, les transformer, les envoyer, les valider. Aujourd'hui, des dizaines et des dizaines d'outils commerciaux ou non existent et ce, dans tous les domaines. Les vérificateurs et les convertisseurs tiennent tout de même le haut du pavé.

Cocoon	Framework	Open Source http://cocoon.apache.org/	Framework de développement Web. Il permet, entre autres, d'interagir avec des données natives XML.
Dtd2xs	Convertisseur	Open source http://www.lumrix.net/	Convertit des DTD en schéma XML
PsychoPath	Xptah	Open Source http://sourceforge.net/projects/psychopath	Dédié à Xpath 2.0, s'appuie sur Xerces pour l'implémentation DOM.
PyXSLdoc	Documentation	Open Source	Permet de générer la documentation d'un XSLT en format XHTML.
Relax NG Converter	Convertisseur	Open Source http://www.sun.com	Outil permettant de convertir des schémas écrits en divers langages en équivalent Relax NG.
Tamino	Serveur	Commercial www.softwareag.com	Serveur XML orienté entreprise pour créer, stocker, gérer et publier des documents XML. Développement des applications en Java possible.
TEXTML Server	Serveur	Commercial http://www.ixiasoft.com	Serveur XML natif. Il contient désormais un outil de conversion XML et est compatible avec .NET. Support de XPath
Visual XSLT	IDE	Commercial http://www.activestate.com/	Editeur visuel pour XSLT. Possède un débogueur. Plug-in .NET.
XML to C	Mapping	Commercial http://www.unicoi.com	Permet de faire un mapping entre XML et le C au niveau structures de données.
XML Judge	Vérificateur	Commercial http://www.topologi.com	Permet de vérifier les documents XML, dans les règles métiers, le respect des standards, etc.
XMLStarlet	Suite	Open Source http://xmlstar.sourceforge.net/	Suite d'outils en ligne de commande pour XML, XPath...
Xflow	Gestion	Commercial http://www.xflows.com/	Outil de gestion XML et XSLT

Les parseurs

Un outil est incontournable dans l'univers XML, le parseur (ou analyseur). Le parseur permet d'extraire les données du document XML (c'est le parsing) et de vérifier la validité du document, si on le demande. De très nombreux parseurs existent en C, Java, Windows, VB, etc. Les éditeurs faisant de l'XML proposent en standard leur parseur ou un parseur du marché.

Chilkat XML .NET	Parser	Gratuit - http://www.chilkatsoft.com/dotNetXml.asp	Parser haute performance pour .NET
Expat XML Parser	Parser	Open Source - http://sourceforge.net/projects/expat/	Parser écrit en C
Libxml2	Parser	Open source	Parser C pour Gnome.
Pathan	Parser	Open source - http://software.decisionsoft.com/	Module pour Xerces-C.
Parsifal	Parser	Gratuit - http://www.saunalahti.fi/~samius/toni/xmlproc/	Parser minimaliste écrit en C. basé sur SAX2.
StAX Pull Parser	Parser	Pré-version - http://www.oracle.com/technology/tech/xml/xdk/staxpreview.html	Parser créé par Oracle.
Xerces2 Java Parser	Parser	Open Source - http://xml.apache.org/xerces2-j/	Parser XML d'Apache.
XML Parser	Parser	Open Source - http://www.destructor.de/xmlparser/	Parser pour Delphi
XML Parser	Parser	Gratuit - http://www.mycgserver.com/~unoksoftgroup/	Parser pour environnement PowerBuilder

Un éditeur XML : Altova

Depuis 1998, cet éditeur commercialise des outils XML, dont son best seller : XMLSpy. Altova propose quatre produits XML : XMLSpy, MapForce, StyleVision et Authentic. XMLSpy 205 incorpore un éditeur de schéma XML, supporte XSLT 2.0 ainsi qu'un mode debug, Xpath 2.0, un éditeur et debug Xquery. Il s'intègre aux environnements Eclipse, Visual Studio .NET, Forte, Jbuilder. Il génère le code pour C#, Java et C++ (à partir du schéma). XmlSpy est un IDE très visuel pour faciliter la conception XML. MapForce est un environnement pour gérer les fichiers plats, XML et les bases de données et permet de générer le code de mapping adéquat (et même de XML à XML). StyleVision est un outil de génération de formulaires électro-

niques transformant du XML et des données en fichiers HTML, PDF, Word (à noter que InfoPath n'est pas supporté). Ici aussi, on bénéficie du XSLT 2, Xpath 2. Authentic est un éditeur XML gratuit afin d'explorer rapidement les données de documents XML. L'éditeur vient de sortir SchemaAgent 2005. Il s'agit d'un outil visuel client/serveur pour modeler et gérer des schémas complexes et les composants collaboratifs. Auparavant, l'outil était intégré à XMLSpy. Il est disponible tout seul ou dans les éditions entreprises et professionnelles de XML Suite 2005. L'outil se connecte à un répertoire de schéma ou une collection de schémas située sur un serveur. Le développeur visualise les liens entre schémas et facilite leur création. Pour accéder au contenu d'un schéma, il est possible d'utiliser la vue XML Schema de XMLSpy 2005. www.altova.com

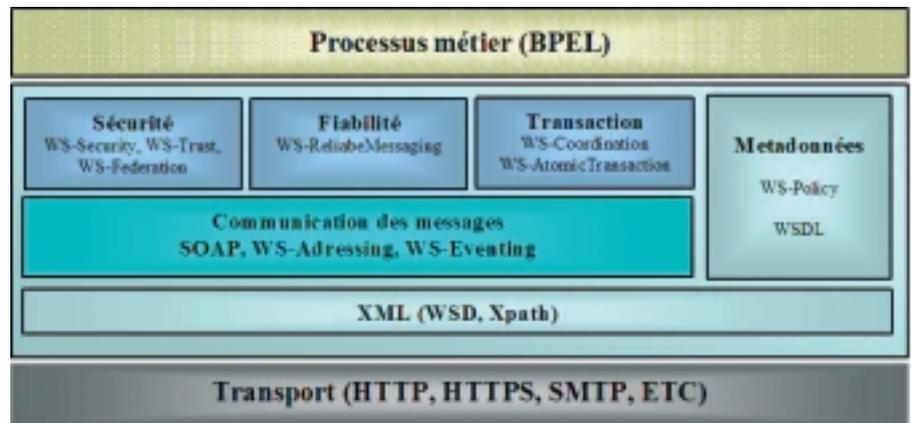
■ F. T.

Panorama des nouvelles spécifications des Web services

Les besoins sans cesse croissants de communication et d'interopérabilité entre applications ont largement contribué au déploiement de technologies de transfert d'information, connues sous le nom de middleware.

Cette situation a notamment permis l'émergence de technologies telles que : Corba, RMI/IIOP, ou encore DCOM. Avec l'avènement d'Internet comme bus de communication permettant de véhiculer des messages entre différentes entreprises, ces besoins ont franchi le cap de la simple communication intra-entreprise. Dès lors, la nécessité de créer un middleware interopérable entre les différentes plates-formes du marché, et prenant en compte tous les mécanismes permettant la réalisation de véritables services, est à l'origine de la conception des services Web. L'enjeu est d'encapsuler les fonctionnalités des applications nouvelles et existantes dans une interface normalisée, et de rendre ces services accessibles, via des protocoles standards du Web. Par ailleurs, les Web services offrent une réponse séduisante aux problématiques d'intégration et d'échange entre les applications, et reposent sur des concepts et standards reconnus, simples à comprendre et surtout à implémenter. Cela a permis à cette technologie de devenir un point de convergence technique de l'ensemble des acteurs du marché de l'informatique. C'est principalement ce point qui fait des Web Services une technologie révolutionnaire et universelle.

Les principales briques technologiques des Web Services ont démontré leur bon fonctionnement et sont arrivées aujourd'hui à maturité. Cependant, ces standards conviennent à des usages très simples. Avec l'émergence des architectures orientées services (SOA), le besoin de réaliser des applications distribuées et des processus métiers plus complexes s'est accentué. Pour y répondre, les standards de base des Web services butent sur des lacunes



WSA : Web Service Architecture

en matière de sécurité, de gestion de transactions et de processus d'échange. Ainsi, le besoin d'une réelle infrastructure s'impose. Bien entendu, la somme des efforts requis pour atteindre un tel résultat est à la mesure des espérances des acteurs qui se sont regroupés pour définir les futures orientations de ces Web services. Ces travaux ont donné lieu à la publication de plusieurs spécifications communes. Ces spécifications, appelées WS*, sont conçues pour fonctionner ensemble et fournir ainsi un environnement de Web services étendu et interopérable. Ces spécifications rentrent dans le cadre d'une nouvelle vision d'architecture globale appelée WSA : Web Service Architecture (voir figure 1).

Vers une architecture étendue des web services

Cette architecture est organisée en couches, chaque couche offre un ensemble de spécifications des fonctionnalités qui peuvent être utilisées par les couches supérieures :

La couche de transport : elle constitue le socle de cette architecture et propose différents protocoles (HTTP, HTTPS, SMTP, etc.). Cette couche a suscité de nombreux efforts de recherche lors de l'émergence des réseaux, mais elle n'est pas à remettre en cause pour l'étude des services Web.

Le langage XML : il s'agit du standard universel dont la syntaxe est le fondement de la défi-

inition des autres composants de cette architecture.

Communication des messages : pour garantir l'interopérabilité des Web services, il est nécessaire de s'assurer que les différentes entités impliquées dans l'échange utilisent le même langage. Dans ce cadre, diverses spécifications ont donc été proposées pour gérer différents aspects liés à la communication des messages tels que l'adressage, le routage, le formatage, la gestion des pièces jointes et la notification des événements. Parmi les spécifications ayant vu le jour, certaines se sont révélées limitées dans leur champ d'application et sont vouées à disparaître. Aujourd'hui, il semblerait qu'il y ait une convergence de l'industrie de Web services vers le choix des spécifications WS-Addressing (adressage et routage), XOP (formatage des données binaires), MTOM (encapsulation des pièces jointes au format XOP) et WS-Eventing qui propose une modèle de publication-souscription.

Sécurité des services Web : WS-Security spécifie des mécanismes garantissant l'intégrité, la confidentialité et l'authentification des messages de bout en bout. En ce qui concerne l'intégrité d'un message, la spécification détaille la représentation d'une signature cryptographique et son association avec des parties spécifiques du message SOAP. L'approche permet à des fragments bien formés du message d'avoir des signatures séparées. D'une façon

similaire, la confidentialité est obtenue par le cryptage de fragments bien formés du message. L'authentification est obtenue en utilisant des signatures numériques. La spécification WS-Security décrit les mécanismes de sécurité standard utilisés aujourd'hui, mais ne ferme pas la porte à l'ajout de nouveaux mécanismes dans le futur. WS-Security présente la brique fondamentale de la sécurité des Web services sur laquelle viennent reposer de nouvelles spécifications : WS-Privacy qui propose un modèle de confidentialité, WS-Trust qui définit la structure destinée à établir des relations de confiance, WS-SecureConversation qui permet le partage d'un contexte de sécurité et enfin WS-Authorization qui définit des stratégies d'accès à un service web. WS-Security est un exemple du consensus industriel sur l'évolution des services Web. Cette spécification est aujourd'hui un standard de l'OASIS.

Fiabilité des Web services : de nombreuses causes peuvent provoquer l'interruption d'un échange de messages entre deux services. Le problème devient important lorsqu'un protocole de transport non fiable comme HTTP ou SMTP est employé, ou lorsqu'un échange de messages met en jeu diverses couches de transport. Des messages peuvent être perdus, dupliqués, réordonnés, ce qui peut entraîner un dysfonctionnement des services Web. WS-ReliableMessaging est un protocole qui assure une livraison fiable des messages en se fondant sur des caractéristiques de garantie de livraison spécifiques. En raison de la conception indépendante du transport, toutes les garanties de livraisons, sont assurées, indépendamment du transport ou des combinaisons de transports utilisés. WS-Reliable Messaging simplifie le développement du système, puisqu'il diminue le nombre d'incidents potentiels que le développeur doit prévoir.

Transaction et coordination : lorsque plusieurs Web services doivent coopérer pour aboutir à un objectif commun, ils doivent s'entendre sur les protocoles à utiliser. Cette coordination minimale est inévitable. WS-Coordination, permet ainsi d'en exprimer le protocole et le scénario. Ce protocole introduit un bloc d'en-tête SOAP, nommé contexte de coordination pour identifier de façon unique le travail en collaboration qui doit être entrepris. Pour commencer, un service Web envoie un contexte de coordination à un ou plusieurs services cibles. L'arrivée du contexte de coordination alerte le

service destinataire qu'une demande de collaboration est formulée. Le contexte de coordination présente suffisamment d'informations au destinataire pour que ce dernier puisse déterminer s'il souhaite participer ou non. Les informations exactes placées dans un contexte de coordination varient selon le type de travail demandé. Par ailleurs, WS-AtomicTransaction décrit les mécanismes des transactions ACID traditionnelles pour des services Web. Dans le contexte d'un type de coordination de transaction courte et atomique, trois protocoles sont définis : un protocole d'achèvement (Completion) et deux variantes d'un protocole de validation à deux phases. Le protocole Completion initialise le traitement d'une validation. Un service Web enregistré pour le protocole Completion a la capacité de se signaler au coordinateur désigné quand le traitement de validation commence. Ce protocole définit aussi des messages pour communiquer le résultat final de la transaction à son initiateur (demandeur). Toutefois, le protocole ne requiert pas que le coordinateur vérifie si le résultat a été traité par l'initiateur. A signaler que WS-AtomicTransaction remplace la première partie de la spécification WS-Transaction.

Métadonnées : toute interaction avec un service Web est réalisée en échangeant des messages SOAP. Pour constituer un environnement de développement et d'exploitation robuste, les services sont décrits par des métadonnées lisibles par les machines. Les métadonnées autorisent l'interopérabilité. Elles décrivent les formats de messages que le service prend en charge, et les modèles d'échange de messages valides pour un service. Les métadonnées expriment aussi les capacités et les exigences d'un service. Cette dernière forme de métadonnées est appelée " stratégie " d'un service. Les formats d'échange de message et les modèles d'échange de message sont exprimés en WSDL. Les stratégies sont formulées en utilisant WS-Policy. WS-Policy fournit un modèle général et une syntaxe permettant de décrire et de communiquer les stratégies d'un Web service. Il définit un ensemble fondamental de constructions qui peut être utilisé et étendu par d'autres spécifications de services Web. Il devient ainsi possible de décrire une large gamme de capacités et de conditions pour un service. WS-Policy inclut une grammaire simple et extensible pour exprimer des assertions de stratégie et un modèle de traitement pour les interpréter. Les assertions de stratégie permettent au program-

meur d'ajouter des métadonnées appropriées aux informations du service, soit lors du développement, soit lors de l'exécution. Comme exemples de stratégies définies lors du développement, citons la taille maximale allouée à un message ou la version exacte d'une spécification prise en charge. Un temps d'arrêt obligatoire du service ou l'indisponibilité d'un service Web pendant une tâche d'administration (maintenance régulière du matériel) constituent pour leur part, des exemples de stratégies définies lors de l'exécution.

Gestion du processus métier : il s'agit de la couche située au sommet de l'organisation WSA et permet de définir la gestion du processus métier. C'est notamment à ce niveau qu'on peut voir émerger la véritable notion de service, pièce maîtresse des architectures orientées services. Le langage BPEL (Business Process Execution Language) s'impose comme le standard incontournable pour la gestion des processus métier. Ce langage a été lancé à l'initiative de Microsoft et IBM, en réponse à l'initiative de BPMI. Depuis, il a reçu le support de la plupart des acteurs du marché, y compris de BPMI. BPEL a été normalisé par OASIS et est devenu la clé de voûte de l'orchestration et de l'automatisation des processus opérationnels.

Conclusion

Les nouvelles spécifications d'extension des web services sont en croissance permanente, principalement du fait de la concurrence accrue entre les différents éditeurs du marché et de l'émergence des SOA. Cela permet d'enrichir et d'étendre les fonctionnalités des applications existantes, et de conquérir de nouveaux utilisateurs. En revanche, les web services risquent d'aller à l'échec si l'on continue à vouloir à tout prix les hisser au même niveau de robustesse et de richesse que certains standards comme CORBA. Il faudrait donc éviter les pièges technologiques et ne pas oublier que c'est la légèreté qui a été le moteur du succès des Web Services, et que c'est précisément la complexité qui a tué CORBA. Il faudrait également que les grands éditeurs cessent de se soumettre aux organismes de standardisation et imposent leurs spécifications. Au risque toutefois de mettre en péril l'interopérabilité des services Web.

■ Tawfik Es-Sqalli

Tawfik Es-Sqalli est consultant chez NEOXIA, intégrateur de systèmes spécialisé dans les nouvelles technologies de l'information.
tawfik.essqalli@neoxia.com



Berkeley DB Java Edition comparé aux mécanismes de mapping O/R

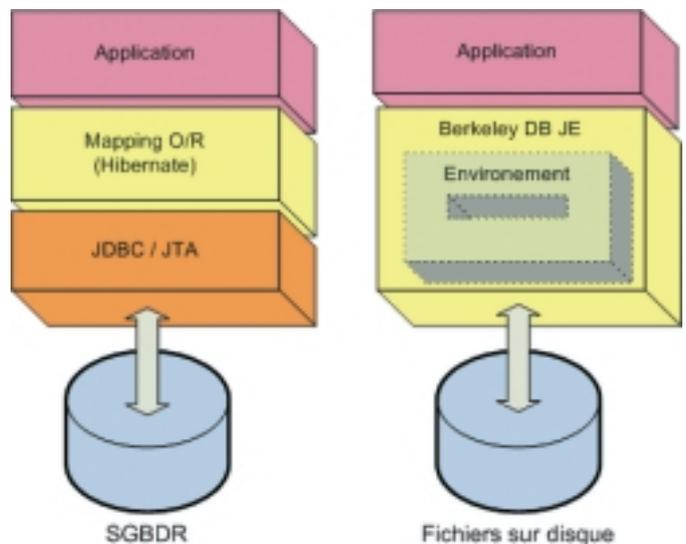
Parmi les solutions de persistance objet de la plate-forme Java, les produits basés sur les mécanismes de mapping Objet/Relationnel se sont imposés comme une alternative quasi incontournable.

Des logiciels tels que Hibernate ou les différentes implémentations JDO (la norme JDO n'impose pas forcément l'utilisation d'un SGBDR) proposent une approche performante et transparente de la persistance, basée sur l'enrichissement des classes Java simples (POJO, Plain Old Java Object). S'appuyant sur une base de données relationnelles, ces solutions tirent profit des deux mondes : le modèle objet, et les requêtes de type SQL/HQL/OQL à travers le modèle de données relationnelles. Autant une base de données offre des fonctionnalités intéressantes, autant, pour certaines applications plus légères, l'utilisation et l'exploitation d'un SGBDR peuvent s'avérer superflues. Berkeley DB Java Edition (BDBJE) propose de combler ce manque en mettant à disposition les mécanismes Java de persistance transactionnelle et de récupération de données sur un système de fichiers traditionnel. Léger, compact et rapide, BDBJE offre une alternative viable pour certains cas de figures. A travers cet article, nous allons illustrer les fonctionnalités de BDBJE et brosser un comparatif succinct avec les solutions de mapping O/R.

Présentation de Berkeley DB Java Edition

BDBJE a été développé par Sleepycat Software, qui distribue également le logiciel homologue Berkeley DB, une implémentation en C d'une solution de persistance transactionnelle. Cette dernière est dotée de nombreuses bibliothèques pour différents langages (Java inclus), leur donnant la possibilité de bénéficier de fonctionnalités de persistance équivalentes. BDBJE est la réécriture en Java de ce produit (ce qui se fait sentir à la consultation de l'API). Sleepycat est une société qui a développé des services autour de ses logiciels distribués en Open Source. La licence de BDBJE est double : le produit est distribué avec une licence similaire à la fameuse "Gnu Public License" (GPL : force toute société qui développe un produit reposant sur un logiciel GPL à mettre à disposition ses sources). Une licence commerciale est disponible pour les sociétés qui préféreraient ne pas divulguer leurs sources. Cette approche n'est pas sans rappeler celle de MySQL AB qui distribue le SGBDR et drivers du même nom sous licence GPL.

BDBJE propose des mécanismes de base de données de bas niveau, dépourvus de structure relationnelle ou de langage de requêtes. Le fonctionnement s'apparente essentiellement à une Map (ensemble de clés/valeurs) pouvant avoir une représentation sur disque. Les données manipulées sont des objets Java pouvant être sérialisés sous forme de tableau d'octets et non pas des types SQL (tel que VARCHAR ou BLOB). BDBJE est 'threadsafe' et offre la possibilité d'effectuer une série d'opérations de façon transactionnelle, garantissant ainsi un état sain des données stockées sur disque. Ce dernier point lui confère un atout majeur dans le cadre d'applications critiques d'entreprise. Au final, le



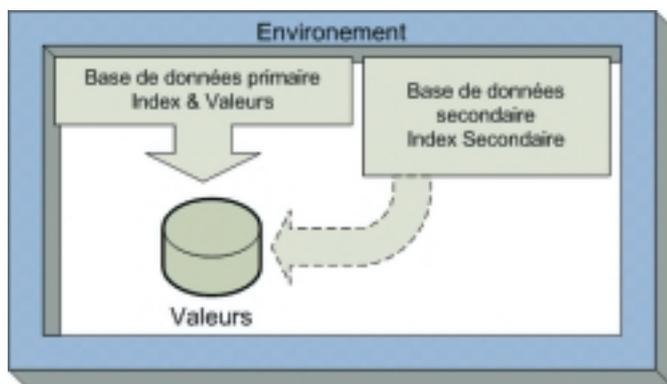
périmètre fonctionnel de BDBJE est suffisamment restreint pour lui conférer une empreinte mémoire réduite et permettre d'être embarqué (le binaire ne pèse pas plus de 450KB) dans une application. Autre point intéressant, BDBJE étant une implémentation Java, il est par conséquent portable sur toutes les architectures supportant une JVM 1.4.2, les fichiers de base de données pouvant être copiés d'une machine à l'autre sans contrainte particulière.

Une instance de base de données telle que la propose BDBJE pourrait s'apparenter à une TreeMap Java (arbre de type b-tree) agrémentée de fonctionnalités de persistance transactionnelle. Un utilisateur place des valeurs dans une base en les associant à une clé, et récupère ces valeurs à l'aide de la même clé. De plus, il est possible de définir plusieurs bases secondaires adressant les mêmes valeurs de la base primaire, mais avec des clés différentes. Un ensemble d'opérations (ajout, lecture, suppression) peut se faire dans un contexte transactionnel afin de garantir une cohérence au niveau des données.

L'utilisation de BDBJE se résume aux points suivants :

- définition d'un environnement (regroupant dans un fichier un ensemble de bases au sens BDBJE) avec notamment la taille/ratio du cache associé,
- définition des conversions de données Java en tableaux d'octets (et vice versa),
- ajout, chargement ou suppression de données dans une base (primaire) à l'aide du couple clé valeur,

- possibilité de définir une base secondaire permettant d'accéder à une base primaire à l'aide de clés différentes,
 - utilisation de curseurs pour itérer sur les valeurs d'une base,
 - la prise en compte d'un ensemble d'opérations de façon transactionnelle.
- Comparé à une solution de mapping O/R, BDBJE est dépourvu de mécanisme de requêtes (tel que HQL pour Hibernate). De plus, BDBJE demande de définir la conversion d'une instance en tableau d'octets et ne gère pas le chargement des relations d'un objet. Hibernate, par exemple, propose le chargement des relations objet 1..n/1..1/n..1/n..n en toute transparence. Avec BDBJE, l'utilisateur devra prendre en charge manuellement la récupération des relations.



Fonctionnement de BDBJE

BDBJE est adapté de la version C avec une API proche. Il n'est pas étonnant de retrouver des approches qui rappellent celles du langage C, à savoir, essentiellement : une utilisation modeste des exceptions Java et des paramètres in et out lors des appels de méthode.

Nous allons présenter successivement les différentes étapes d'utilisation de BDBJE, avec leur représentation programmatique.

Création d'un environnement

L'environnement regroupe un ensemble de bases (il pourrait s'apparenter à la notion de base de données dans un SGBDR). L'environnement permet surtout de définir l'emplacement sur le système de fichiers dans lequel seront stockées les données de chaque base.

L'environnement prend également en charge la gestion du cache (taille et ratio mémoire) et la gestion des opérations transactionnelles.

```
Environment environment = null;

try {

    EnvironmentConfig envConfig = new EnvironmentConfig();
    envConfig.setAllowCreate( true );
    envConfig.setCacheSize( 128*1024*1024 );
    envConfig.setTransactional( true );
    environment = new Environment( new File( "/home/al/works/bdbje/db/db1" ), envConfig );

} catch ( DatabaseException dbe ) {
    dbe.printStackTrace();
}
```

A travers ces quelques lignes, nous avons indiqué que :

- setAllowCreate : l'environnement devrait être créé si inexistant
- setCacheSize : la taille du cache devrait représenter 128K
- setTransactional : la prise en compte des transactions

Création d'une base

Une base de données au sens BDBJE est l'équivalent d'une TreeMap Java un peu plus sophistiquée. Les bases sont regroupées dans un environnement utilisé pour leur instantiation.

```
DatabaseConfig dbConfig = new DatabaseConfig();
dbConfig.setAllowCreate(true);
database = environment.openDatabase(null, "database1", dbConfig);
```

Binding de classe & définition des comparateurs

Binding

Les valeurs sont stockées sous forme de tableaux d'octets dans les bases. Il est donc nécessaire d'indiquer comment transformer une instance d'un objet en tableau d'octets (et inversement lors du chargement).

BDBJE propose deux approches :

- en s'appuyant sur les mécanismes de sérialisation natifs de la plateforme Java : cette solution a l'avantage d'être simple, mais en revanche n'est pas forcément performante. La sérialisation Java ne concerne pas uniquement les données utiles d'une instance, c'est pourquoi BDBJE propose une API permettant de bénéficier de la sérialisation Java et d'en extraire les données nécessaires.
- en donnant la possibilité au développeur de définir sa correspondance (binding) Java ↔ ByteArray.

Dans le cas de la sérialisation, le principe est de stocker dans une base séparée les informations de structure de la classe (ce qui évite de dupliquer cette information pour chaque entrée).

```
MyClass myData = new MyClass();

// Création de la base qui va stocker les information de structure des classes
Database classBase = environment.openDatabase(null, "classBase", dbConfig);
// Définition de la base comme catalogue de structures de classes
StoredClassCatalog classCatalog = new StoredClassCatalog( classBase );
// Création de l'outil de conversion instance ũ bytarray
EntryBinding dataBinding = new SerialBinding( classCatalog, MyClass.class );
// Entrée qui sera stockée en base (via la méthode put)
DatabaseEntry dataEntry = new DatabaseEntry();
// Conversion de l'instance myData (paramètre in) en tableau,
// stocké directement dans l'instance dataEntry (paramètre out)
dataBinding.objectToEntry( myData, dataEntry );
```

A travers ces quelques lignes, nous laissons BDBJE transformer les informations de l'instance myData en tableau d'octets dans l'entrée dataEntry qui sera utilisée pour le stockage en base.

Les structures de notre classe "MyClass" (Serializable) sont stockées dans une base dédiée (classBase).

Cette API simplifie le travail du développeur : BDBJE extrait les données

des instances sous forme de tableau d'octets pouvant alors être persisté dans la structure de la base sur disque.

Comparateur

Le couple clé/valeur est stocké dans une base organisée sous forme de B-Tree. Autrement dit, les informations sont organisées sous forme d'arbre trié en fonction de l'index. Un comparateur sur l'index de chaque nœud est utilisé pour classer les sous-éléments.

Par défaut, la comparaison est effectuée sur chaque élément du tableau d'octets. Cependant, afin d'optimiser le tri des données selon un ordre différent, il est possible d'implémenter un comparateur (`java.util.Comparateur`) spécifique pour l'index et de l'associer à une base :

```
dbConfig.setBtreeComparator( MyIndexComparator.class );
```

Nous spécifions ainsi le comparateur "MyIndexComparator" (qui implémente l'interface `Comparator`) sur l'index de notre base (dont la configuration est définie dans l'instance `dbConfig`).

Ajout, chargement & suppression d'une clé/valeur dans une base

Nous pouvons maintenant stocker nos données dans la base que nous avons créée.

Les données manipulées pour les ajouts, chargements, suppressions sont de type `DatabaseEntry`. Elles contiennent les tableaux d'octets qui sont obtenus par les opérations `binding`.

Les opérations s'effectuent sur une instance de base.

L'ajout s'effectuera :

```
DatabaseEntry key = new DatabaseEntry( keyStr.getBytes("UTF-8") );
database.put( tx, key, dataEntry );
```

La méthode `'put'` utilise 3 paramètres :

- la transaction courante (ici `tx = null` : nous utilisons le mode `auto-commit`)
- l'entrée pour la clé
- l'entrée pour la valeur (obtenue précédemment à l'aide de l'API de sérialisation)

Le chargement :

```
if ( database.get( tx, key, dataEntry, LockMode.DEFAULT ) ==
    OperationStatus.SUCCESS )
{
    ...
}
```

Le fonctionnement est similaire au `'put'` avec la possibilité de spécifier un mode de lecture à travers le `'LockMode'` (permettant par exemple de lire les données non `'commit-ées'`).

La suppression :

```
database.delete( tx, key );
```

Il est également possible de vider l'intégralité de la base à l'aide de la méthode :

```
environment.truncateDatabase( tx, "database1", false );
```

Définition d'une base secondaire

Pour des raisons de performance, il peut être intéressant d'accéder aux valeurs d'une base mais avec un autre index. BDBJE permet d'associer à une base primaire, une seconde base avec un index différent pointant sur le même ensemble de valeurs. Si une valeur est modifiée dans la base primaire, elle est identique si accédée à travers la seconde base. L'index de la base secondaire évite alors d'itérer sur l'ensemble des valeurs pour trouver un élément en fonction d'un attribut autre que la clé primaire.

```
database = environment.openDatabase( null, "database1", dbConfig );
SecondaryConfig config2 = new SecondaryConfig();
config2.setKeyCreator( keyCreator );
database2 = environment.openSecondaryDatabase( null, "database2",
    database, config2 );
```

Le `'keyCreator'` est une instance d'une classe qui implémente l'interface `'SecondaryKeyCreator'` (nous ne détaillerons pas cette implémentation dans cet article).

Curseur

Il est possible d'itérer sur l'ensemble des valeurs d'une base à l'aide d'un curseur.

```
Cursor cursor = database.openCursor( tx, cursorConfig );
while ( cursor.getNext( foundKey, foundData, LockMode.DEFAULT ) ==
    OperationStatus.SUCCESS )
{
    byte[] keyByteArray = foundKey.getData();
    byte[] valueByteArray = foundData.getData();
}
```

Le champ de prospection du curseur peut être réduit sur un sous-ensemble d'index en utilisant les méthodes de la classe `'Cursor'` :

```
getSearchKeyRange(DatabaseEntry key, DatabaseEntry data, LockMode
lockMode)
getSearchBothRange(DatabaseEntry key, DatabaseEntry data, LockMode
lockMode)
```

Le curseur se positionne alors en fonction de la clé(/valeur) de recherche fournie.

Pour réduire l'étendue d'une recherche, plusieurs curseurs définis sur plusieurs index de base secondaire peuvent être associés pour sélectionner un ensemble très restreint de valeurs de la base primaire.

Transactions

BDBJE supporte les principes d'Atomicité, Consistance, Isolation & Durabilité pour les transactions. La fenêtre de transaction est définie manuellement puis passée en paramètre de chaque opération si tant est que la base supporte les transactions.

```
Transaction tx = environment.getEnv().beginTransaction(null, null);
...
try
```

```

{
    database.put( tx, key, value );
}
catch (DatabaseException dbe)
{
    txn.abort();
    throw dbe;
}
txn.commit();
    
```

L'appel à la méthode 'commit' vide les informations du contexte et, éventuellement (en fonction du mode transactionnel), les synchronise sur disque. En cas d'exception, la transaction est annulée (abort) et l'état initial est restitué.

Comparatif mapping O/R

Comparé aux solutions de mapping O/R telles qu'Hibernate, BDBJE demande plus de responsabilités au développeur qui doit s'occuper d'un ensemble de tâches manuelles. Là où Hibernate effectue en toute transparence les opérations, l'utilisateur devra se charger de les programmer avec BDBJE. C'est particulièrement le cas avec les chargements des relations objets. De plus, BDBJE étant complètement dépourvu de langage de requêtes, il force également l'utilisateur à développer son propre code pour récupérer les sous-ensembles de données.

En contrepartie, l'application embarque la base et donc, à ce titre, s'avère extrêmement efficace au niveau des performances.

En réalité, les deux solutions ne sont pas au même niveau. BDBJE pourrait bénéficier de mécanismes fortement inspirés de la programmation aspect pour simplifier son utilisation et s'apparenter ainsi aux produits tels que Hibernate.

En revanche, si la persistance doit forcément s'appuyer sur un SGBDR pour des raisons d'administration ou de centralisation, la solution O/R est incontournable.

Conclusion

BDBJE offre un compromis intéressant entre performance et simplicité, mais demande au développeur beaucoup de travail dans le cas de modèles d'envergure, nécessitant des requêtes complexes. Couplé avec un framework de plus haut niveau, BDBJE pourrait proposer un compromis très efficace. BDBJE est souvent utilisé comme brique de base d'applications proposant des services de plus haut niveau. BDBJE est proposé, par exemple, dans le cadre d'un cache objet transactionnel pour JBoss ou comme stockage transactionnel pour serveur de message Active MQ (implémentation de la norme JMS). Sleepycat propose également une implémentation de Collection Java en surcouche de BDBJE, bénéficiant des fonctionnalités transactionnelles.

Dans un autre registre, Sleepycat a également développé une base de données XML (avec le support de la norme XQuery optimisée au niveau des index) s'appuyant sur l'implémentation C de Berkeley DB.

Références

Berkeley DB JE : <http://sleepycat.com/products/db.shtml>

Hibernate : <http://hibernate.org/>

JDO JSR : <http://www.jcp.org/aboutjava/communityprocess/first/jsr012/>

Cache JBoss avec BDBJE : <http://www.sleepycat.com/jeforjboss/cache/>

Active MQ : <http://activemq.codehaus.org/>

Berkeley DB XML : <http://sleepycat.com/products/xml.shtml>

■ Alexis AGAHI - Expert XML/J2EE

Homsys-Aston



Homsys-Aston, filiale de Homsys Group résultant de la fusion de Aston, Homsys, Kedros et Objet Direct, est spécialisé autour des domaines suivants : la Business Intelligence, les Architectures Systèmes et Réseaux, les technologies Objet et Internet, le Workflow et le BPM. Homsys-Aston est implanté à Paris, Lyon, Marseille, Bordeaux, Toulouse, Grenoble et Rennes et fédère plus de 500 collaborateurs. Homsys Group est coté en Bourse www.homsysgroup.com



- L'actualité
- Le téléchargement
- Le forum technique : questions réponses
- Les archives du magazine : les articles publiés depuis 2002

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

Couche de persistance avec Spring



Spring fait partie d'une nouvelle génération de frameworks qui ont pour objectif de simplifier le développement J2EE. On utilise pour ces frameworks le terme de conteneurs légers ("light-weight container") par opposition aux conteneurs "lourds", c'est à dire les conteneurs EJB.

L'objectif de Spring est donc de simplifier J2EE en permettant :

- un développement totalement objet qui utilise de simples classes Java ("Plain Old Java Object"),
- un développement testable unitairement.

Pour cela, le fonctionnement de Spring est basé sur :

- un pattern : l'injection de dépendance,
- un type de programmation : la programmation orientée aspect qui complète la programmation orientée objet.

Spring fournit des services de haut niveau pour les développements J2EE, dans les différentes couches de l'application : des services que l'on ne trouvait jusque là que dans les conteneurs EJB, comme une gestion transactionnelle déclarative.

Ainsi Spring propose entre autres :

- un conteneur d'objets qui, au delà de l'injection de dépendance, permet aussi de gérer le cycle de vie complet de ces objets,
- la possibilité d'utiliser la programmation orientée aspect,
- un mécanisme de DAO qui facilite l'utilisation de JDBC et de différentes solutions de mapping objet-relationnel (dont Hibernate et bientôt Toplink),
- une gestion transactionnelle puissante,
- un framework web MVC très souple. Laissant le choix de la technologie à utiliser sur la partie vue,
- des classes qui facilitent l'utilisation de JMS, RMI, JAX RPC et des web services en général.

Malgré sa richesse, Spring est très modulaire : on peut très bien décider de n'utiliser qu'une partie de ces possibilités. Il n'essaie pas de réinventer la roue et se positionne plutôt comme un intégrateur de frameworks existants. Une combinaison très populaire est celle de Spring et d'Hibernate. Dans ce cas Spring apporte des classes qui forment une couche d'abstraction. Elles permettent d'éviter de manipuler directe-

ment des concepts techniques comme la sessionFactory. L'écriture du code est simplifiée. Les risques de bug sont moindres.

Spring peut fonctionner avec ou sans serveur applicatif. Il est parfaitement adapté à un processus de développement agile.

Dans cet article nous montrerons le fonctionnement de l'injection de dépendance, de l'accès aux données par JDBC et de la gestion transactionnelle déclarative.

1 Installation et mise en oeuvre de l'injection de dépendance

L'injection de dépendance est le principe de fonctionnement de base de Spring (et de tous les conteneurs légers). Ce pattern veut que si un objet A a besoin d'un objet B, ce dernier lui est fourni. A n'instancie pas B. Il reçoit "automatiquement" une instance de cet objet. Cela signifie que les objets ne sont pas dépendants des implémentations dont ils ont besoin, mais seulement des interfaces. Cela a plusieurs avantages :

- un faible couplage des composants dans l'application. La modification d'une implémentation n'a aucun impact sur le reste de l'application. Si par exemple, le mécanisme de persistance change, le reste de l'application n'est pas impacté.
- un développement incrémental et modulaire est alors possible,
- les classes peuvent être testées unitairement en fournissant des "mock objects" comme implémentation des objets dont on est dépendant.

L'injection de dépendance est implémentée dans Spring par la BeanFactory qui est chargée d'instancier les objets et de résoudre leurs dépendances les uns avec les autres. Ces dépendances sont définies dans un fichier XML. La BeanFactory est donc le conteneur de nos objets. Cependant, dans la pratique, au lieu d'utiliser directement la

BeanFactory, il est préférable d'utiliser l'ApplicationContext qui étend la BeanFactory et propose des services supplémentaires dont :

- la possibilité d'utiliser la programmation orientée aspect,
- le support de l'internationalisation des messages,
- l'utilisation de fichiers properties pour externaliser la configuration (exemple : la configuration JDBC),
- l'utilisation de plusieurs fichiers XML (un pour la couche métier, un pour la couche de persistance...) de définition des dépendances.

Comment cela fonctionne-t-il ? Imaginons une application dans laquelle nous voulons enregistrer des entreprises et leurs coordonnées postales. Pour cela nous avons :

- deux DAO qui contiennent des méthodes simples de select, insert, update et delete pour l'objet entreprise ("CompanyDAOImpl") et l'objet adresse ("AddressDAOImpl").
- une classe de type "service" appelée depuis un contrôleur du framework MVC de notre choix qui gère la logique applicative de la fonctionnalité d'enregistrement d'une nouvelle entreprise ("CompanyServiceImpl").

Par exemple, pour la DAO "CompanyDAOImpl" nous avons :

```
public class CompanyDAOImpl implements CompanyDAO {

    public Company findById(String Id) {
        ....
    }

    public void insert(Company company) {
        ...
    }

    public void update(Company company){
        ...
    }

    public void delete(Company company) {
        ...
    }

}
```

La classe "CompanyServiceImpl" est la suivante :

```
public class CompanyServiceImpl implements CompanyService {

    private CompanyDAO companyDAO;

    private AddressDAO addressDAO;

    public CompanyServiceImpl(CompanyDAO companyDAO,
        AddressDAO addressDAO) {
        this.companyDAO = companyDAO;
        this.addressDAO = addressDAO;
    }

}
```

```
...

    public void add(Company company) {
        companyDAO.insert(company);
        addressDAO.insert(company.getAddress());
    }

    ...
}
```

La classe "CompanyServiceImpl" a une dépendance avec deux DAO. Nous allons demander à Spring de résoudre ces dépendances. Ainsi, quand Spring fournira une instance de la classe service, il se chargera de lui injecter les bonnes implémentations de ces DAO. Pour cela nous allons déclarer ces objets (et leur dépendance) dans un fichier XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.
springframework.org/dtd/spring-beans.dtd">

<beans>

<!-- ===== SERVICE OBJECTS ===== -->

    <bean id="companyService"
class="fr.sqli.spring.exemple.domain.CompanyServiceImpl"
        singleton="true" dependency-check="objects">
        <constructor-arg index="0"><ref bean="companyDAO"/>
</constructor-arg>
        <constructor-arg index="1"><ref bean="addressDAO"/>
</constructor-arg>
    </bean>

<!-- ===== DATA ACCESS OBJECTS ===== -->

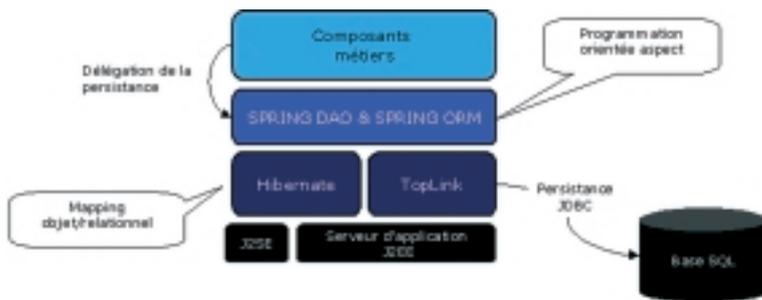
    <bean id="companyDAO"
class="fr.sqli.spring.exemple.data.CompanyDAOImpl"
        singleton="true">
    </bean>

    <bean id="addressDAO" class="fr.sqli.spring.exemple.data.Address
DAOImpl"
        singleton="true">
    </bean>

</beans>
```

Pour chaque classe il faut un spécifier, un id et son nom complet. Puis il est possible d'indiquer si cette classe est un singleton. Dans cet exemple l'injection se fait par le constructeur de la classe "CompanyServiceImpl".

Il existe alors plusieurs possibilités pour demander à Spring l'instance d'un objet. La plus simple est d'utiliser le ContextSingletonBeanFactory



Locator. Le (ou les fichiers) de contexte XML sont alors parsés lors de la première demande d'une instance, puis une factory d'objets est maintenue en mémoire. Pour demander une instance, il suffit d'interroger le ContextSingletonBeanFactoryLocator :

```
ContextSingletonBeanFactoryLocator.getInstance().useBeanFactory(
    "fr.sqli.spring").getFactory().getBean("companyService);
```

2 Service d'accès aux données.

Spring facilite énormément l'écriture des DAO grâce à un mécanisme de classes abstraites et de template. Les exemples que nous allons donner ici sont basés sur JDBC. Cependant, le principe de fonctionnement est identique pour les autres solutions de persistances supportées par Spring (Hibernate, Ibatis, JDO, ...)

Tout d'abord nos DAO doivent hériter d'une classe abstraite. Dans notre cas "JdbcDaoSupport" :

```
import org.springframework.jdbc.core.support.JdbcDaoSupport;

public class CompanyDAOImpl extends JdbcDaoSupport implements
CompanyDAO {
    ...
}
```

Cette classe abstraite nous fournit des méthodes permettant d'utiliser facilement JDBC. En l'occurrence, plus besoin de rechercher la DataSource, d'ouvrir et de fermer les connexions, c'est le framework qui s'en charge. L'écriture d'un ordre SQL devient extrêmement simple :

```
import org.springframework.jdbc.core.support.JdbcDaoSupport;

public class CompanyDAOImpl extends JdbcDaoSupport implements
CompanyDAO {

    public void insert(Company company) {

        getJdbcTemplate().update("INSERT INTO Company
(id, name) VALUES (?, ?)",
                                new Object[] { company.getId(),
company.getName() });
    }
}
```

Si nulle part dans notre code nous ouvrons et fermons une connexion JDBC, c'est parce que Spring va utiliser l'injection de dépendance pour fournir la connexion, ou plutôt la DataSource, à utiliser à la DAO. Il faut donc compléter le fichier XML de configuration de cette façon (l'injection est réalisée par setter) :

```
<bean id="companyDAO" class="fr.sqli.spring.exemple.data.Company
DAOImpl"
    singleton="true" dependency-check="none">
    <property name="dataSource"><ref bean="dataSource"/></property>
</bean>

<bean id="addressDAO" class="fr.sqli.spring.exemple.data.Address
DAOImpl"
    singleton="true" dependency-check="none">
    <property name="dataSource"><ref bean="dataSource"/></property>
</bean>
```

La DataSource est définie ainsi :

```
<bean id="dataSource" class="org.apache.commons.dbcp.BasicData
Source" destroy-method="close">
    <property name="driverClassName"><value>${jdbc.driverClassName}</value></property>
</value></property>
    <property name="url"><value>${jdbc.url}</value></property>
    <property name="username"><value>${jdbc.username}</value>
</property>
    <property name="password"><value>${jdbc.password}</value>
</property>
    <property name="maxIdle"><value>${jdbc.maxIdle}</value></property>
    <property name="minIdle"><value>${jdbc.minIdle}</value></property>
</bean>
```

Pour ne pas indiquer directement dans le fichier XML la configuration de la DataSource, nous utilisons un mécanisme de Spring qui permet de les externaliser dans un fichier properties :

```
<bean id="jdbcProperties" class="org.springframework.beans.factory.
config.PropertyPlaceholderConfigurer">
    <description>Connection pool specific options</description>
    <property name="location">
        <value>jdbc.properties</value>
    </property>
</bean>
```

Voici le fichier "jdbc.properties" :

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/exemple
jdbc.username=toto
jdbc.password=
jdbc.maxIdle=100
jdbc.minIdle=5
```

Dans le cas d'une persistance gérée par Hibernate, le mécanisme est semblable. Au lieu de faire hériter nos DAO de JdbcDaoSupport, nous les faisons hériter de HibernateDaoSupport. Et au lieu de leur injecter une DataSource, c'est une SessionFactory qui est utilisée.

3 Gestion transactionnelle

Spring fournit une puissante gestion transactionnelle qui peut être programmatique ou déclarative. Cette dernière possibilité est très intéressante, car elle permet d'externaliser du code les démarcations transactionnelles. De plus, comme pour les DAO, le mécanisme mis en oeuvre est identique quel que soit le type de persistance. Seul, le gestionnaire de transaction choisi change. Tout n'est plus qu'histoire de configuration.

Nous allons donc rendre transactionnel, l'ajout d'une entreprise et de ses coordonnées bancaire dans notre application.

Pour cela nous avons besoin de déclarer un gestionnaire de transaction pour la DataSource que nous utilisons :

```
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
  <property name="dataSource"><ref local="dataSource"/></property>
</bean>
```

Dans le cadre des DAO, la solution la plus simple est que leurs méthodes soient déclarées systématiquement transactionnelles suivant un schéma de nommage :

```
<!-- Transaction interceptor -->

<bean id="transactionInterceptor" class="org.springframework.transaction.interceptor.TransactionInterceptor">
  <property name="transactionManager"><ref bean="transactionManager"/>
</property>
  <property name="transactionAttributeSource"><ref bean="transactionAttributes"/>
</property>
</bean>

<!-- Déclaration des niveaux transactionnels des méthodes -->

<bean id="transactionAttributes" class="org.springframework.transaction.interceptor.NameMatchTransactionAttributeSource">
  <property name="properties">
    <value>find*=ISOLATION_DEFAULT, PROPAGATION_SUPPORTS,
ISOLATION_DEFAULT, -Throwable, readOnly
      insert=ISOLATION_DEFAULT, PROPAGATION_REQUIRED,
ISOLATION_DEFAULT, -Throwable
      update=ISOLATION_DEFAULT, PROPAGATION_REQUIRED,
ISOLATION_DEFAULT, -Throwable
      delete=ISOLATION_DEFAULT, PROPAGATION_REQUIRED,
ISOLATION_DEFAULT, -Throwable
    </value>
  </property>
</bean>
```

```
<!-- Declare toutes les DAO comme objets transactionnels -->

<bean id="autoProxyCreator" class="org.springframework.aop.framework.autoproxy.BeanNameAutoProxyCreator">
  <property name="interceptorNames"><value>transactionInterceptor</value>
</property>
  <property name="beanNames"><value>*DAO</value></property>
</bean>
```

Dans ce cas, Spring va utiliser ces possibilités de programmation orientée aspect pour nous permettre de définir les différents comportements transactionnels de nos méthodes.

Le cas d'une des classes de type service est un peu différent, car elles ne doivent pas être systématiquement transactionnelles. Nous allons donc indiquer uniquement CompanyService comme transactionnelle :

```
<bean id="companyServiceTarget" class="fr.sqli.spring.exemple.domain.CompanyServiceImpl"
  singleton="true" dependency-check="objects">
  <constructor-arg index="0"><ref bean="companyDAO"/>
</constructor-arg>
  <constructor-arg index="1"><ref bean="addressDAO"/>
</constructor-arg>
</bean>

<bean id="companyService"
  class="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
  <property name="transactionManager"><ref bean="transactionManager"/>
</property>
  <property name="target"><ref bean="companyServiceTarget"/>
</property>
  <property name="transactionAttributes">
    <props>
      <prop key="add">PROPAGATION_REQUIRED</prop>
      <prop key="update">PROPAGATION_REQUIRED</prop>
      <prop key="delete">PROPAGATION_REQUIRED</prop>
    </props>
  </property>
</bean>
```

Conclusion

Avec tous les services qu'il propose, Spring est un framework complet. Il est un concurrent direct aux conteneurs EJB, en particulier lorsqu'il est associé à Hibernate. Son appropriation est facile. Bref, il va rapidement devenir un outil incontournable du développement J2EE.



Référence :

Springframework : <http://www.springframework.org>
L'injection de dépendance :
<http://martinfowler.com/articles/injection.html>

■ Arnaud PROST, SQLI



J2EE 1.4 : un Web Service à café

Avec les spécifications J2EE 1.4, les Web Services font maintenant officiellement partie de l'univers de Java. Découvrons par la pratique, comment les mettre en oeuvre.

Les Web Services ne sont pas totalement nouveaux en Java car il existait déjà des implémentations Java de SOAP, la plus connue étant probablement AXIS de la fondation Apache. Toutefois, les Web Services font désormais partie des spécifications de J2EE 1.4. C'en est même l'innovation majeure, un peu comme l'étaient les Beans pilotés par messages de J2EE 1.3. En outre, Sun fournit désormais une implémentation de référence, avec laquelle nous allons travailler.

1 Notre but et les outils

Notre but est de mettre en place un Web Service en Java de type "Hello World" et d'écrire un client statique en Java. Les clients dynamiques sont faisables, mais nous les réservons à un prochain article. Nous avons encore écrit un client en Python, pour le plaisir. Pour tout cela nous avons travaillé sous Linux, puis testé sous Windows également. Étant donné le peu de code à écrire, nous avons utilisé Emacs pour coder et compiler avec une JDK 1.5, le minimum requis étant une JDK 1.4.2. Nous avons codé et compilé localement et déployé sur un serveur distant. Comme dit plus haut nous utilisons l'implémentation de référence de Sun qui peut être téléchargée à <http://java.sun.com/j2ee/1.4>. Les spécifications J2EE 1.4 peuvent être téléchargées depuis cet endroit. Nous avons en outre utilisé les spécifications "JAX-RPC" et les spécifications "Web Services for J2EE 1.1 Requierevements" pour établir les descripteurs de déploiement notamment. L'implémentation de référence de Sun est très intéressante lorsqu'il s'agit de se former à tous les aspects de la technologie J2EE, ou de faire des essais, en raison de sa légèreté. Son outil de déploiement est relativement agréable à utiliser. Par contre, la plate-forme ne supporte pas la montée en charge et ne peut être utilisée en production, ainsi que l'indique d'ailleurs Sun. Son installation ne pose aucun problème quel que soit le système hôte. Sous Windows, on démarrera le serveur depuis le menu du système. Sous Linux on saisira dans un terminal :

```
asadmin start-domain domain1
```

après avoir pris soin d'exporter le sous-répertoire bin du serveur dans le PATH. On stoppe le serveur comme ceci :

```
asadmin stop-domain
```

Normalement le port 8080 est écouté par défaut pour HTTP. Toutefois, si au moment de l'installation ce port n'est pas libre, le port 3099 sera vraisemblablement employé à la place. Ceci peut être configuré depuis la console d'administration que l'on ouvre dans un navigateur à l'URL <http://machine:4848>. Nous disons machine car, pour une raison indéfinissable, localhost ne fonctionne pas toujours très bien sous Linux. Donc, nous donnons le nom de la machine. De plus, nous avons remarqué que le serveur doit être lancé sous le compte root pour éviter des surprises, même si l'on écoute un port au delà de 1024 et que l'on a tous les droits. Enfin, nous recommandons vivement de ne jamais redéployer par dessus un déploiement antérieur, mais de toujours supprimer

ce dernier depuis la console d'administration (plutôt que depuis l'outil de déploiement) avant de redéployer. Le bon fonctionnement de l'installation peut être vérifié à l'URL <http://localhost:8080>. Une page de garde devant apparaître dans votre navigateur. Enfin, pour compiler et faire fonctionner les exemples, le CLASSPATH doit pointer un certain nombre de packages de l'implémentation de référence. Pour vous faciliter la vie nous donnons un script Shell, setenv, qui met à jour le CLASSPATH pour vous, mais que vous devrez adapter à votre système et éventuellement transformer en fichier .bat pour Windows. Vous trouverez tous les fichiers de code ou descripteurs sur le Cd-Rom accompagnant le magazine.

2 Qu'est-ce qu'un Web Service ?

Nous nous référons à la définition moderne du Web Service qui nous dit qu'il s'agit d'une architecture permettant l'invocation de procédures distantes au travers du protocole HTTP. Ceci présente l'avantage de ne pas poser de difficulté avec les pare-feu. En outre, les Web Services sont indépendants des OS et des langages, ce qui permet de travailler dans les environnements les plus hétérogènes qui soient. Bien évidemment, dans un contexte J2EE, le Web Service est un moyen d'accéder aux EJB autrement que par Servlets ou JSP. Plus précisément maintenant, un Web Service est au minimum un empilement de 5 couches. Les voici du niveau le plus haut au niveau le plus bas.

Couche	Standards
Découverte des services	UDDI
Description des Services	WSDL
Messages	SOAP
Encodage	XML
Transport	HTTP

La couche de découverte de services permet aux clients de localiser et analyser des Web Services à la volée. Nous réservons ce sujet pour un article à venir. Aujourd'hui nous nous limitons à un client Java statique, précompilé en connaissance de cause. La couche de description des services sert à exposer l'interface sur le réseau, normalement par le biais d'un document WSDL. Avec J2EE nous écrivons un document XML à la place et compilerons celui-ci avec l'outil wscompile de l'implémentation, ce qui aboutira à la génération du document WSDL requis. WSDL utilise entre autres les notions de port, opération et parties (parts). Sous un angle Java nous comprendrons les choses comme ceci :

- Un port WSDL est analogue à une interface Java.
 - Une opération WSDL est analogue à une méthode. Nous ne nous soucions pas ici de savoir à quelle classe appartient la méthode. Le compilateur wscompile résout le problème pour nous.
 - Les parts sont analogues aux arguments de méthodes et valeur de retour.
- La couche de messages est gérée par le protocole SOAP. En Java ce protocole est implémenté par l'API JAX-RPC, cette dernière se basant sur l'API SAAJ. Toutes ces APIs viennent avec l'implémentation de référence

et ne poseront pas de problème, pour autant qu'elles soient pointées par le CLASSPATH, comme dit plus haut. La couche d'encodage est basée sur XML comme avec tout Web Service qui se respecte et l'on ne présente plus le protocole HTTP.

3 Au travail

A tout seigneur tout honneur, nous commençons par l'interface. Pour un service ProgrammezService, l'interface se nomme conventionnellement ProgrammezServiceIF et en voici le code :

```
package programmez.fred.ws;

import java.rmi.*;

public interface ProgrammezServiceIF extends Remote {
    public String helloWorld() throws RemoteException;
}
```

Rien d'extraordinaire ici. Comme avec RMI et non sans logique, interface doit étendre Remote et chaque méthode est susceptible de lever RemoteException. Nous pouvons donc implémenter notre interface, ce qui est une formalité. La classe a également un nom conventionnel.

```
package programmez.fred.ws;

public class ProgrammezServiceImpl implements ProgrammezServiceIF {
    public String helloWorld() {
        return "\nProgrammez!\nEt Abonnez vous :-)\n";
    }
}
```

Compilons nos deux fichiers et demandons au compilateur de les placer dans les répertoires correspondant aux packages :

```
javac -cp $CLASSPATH -d . *.java
```

L'étape suivante consiste à écrire le descripteur XML (fichier service-config.xml). Le voici :

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
  <service
    name="ProgrammezService"
    targetNamespace="urn:programmezService"
    typeNamespace="urn:programmezService"
    packageName="programmez.fred.ws">
    <interface name="programmez.fred.ws.ProgrammezServiceIF"/>
  </service>
</configuration>
```

Nous remarquons qu'en plus de donner le nom de package, nous devons obligatoirement qualifier complètement le nom de l'interface. Compilons maintenant ce fichier XML pour obtenir le document WSDL de notre Web Service.

```
wscvcompile -define -nd . -classpath . service-config.xml
```

4 Déploiement

Pour déployer, nous avons d'abord besoin d'un descripteur, dit mapping, et dont le rôle est d'établir la correspondance entre les objets Java et le Web Service tel que décrit par le WSDL. Ce descripteur est lui aussi un document XML :

```
<?xml version="1.0" encoding="UTF-8"?>
<java-wsdl-mapping xmlns="http://java.sun.com/xml/ns/j2ee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://www.
  ibm.com/webservices/xsd/j2ee_jaxrpc_mapping_1_1.xsd"
  version="1.1">
  <package-mapping>
    <package-type>programmez.fred.ws</package-type>
    <namespaceURI>urn:programmezService</namespaceURI>
  </package-mapping>
</java-wsdl-mapping>
```

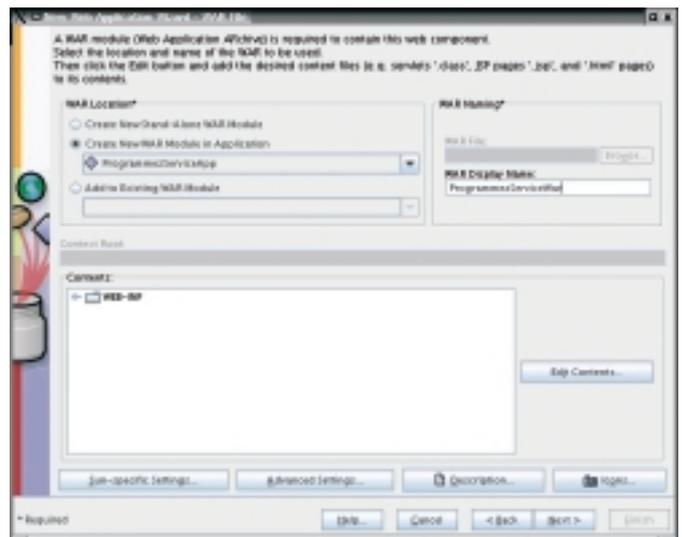


Fig. 1 : Création du composant Web.

Nous pouvons maintenant déployer. Pour cela, lancez l'utilitaire deploy-tool de l'implémentation de référence. Depuis le menu de l'outil, choisissez FileNewApplication et donnez par exemple ProgrammezServiceApp comme nom de fichier. ProgrammezServiceApp étant sélectionné dans la liste arborescente, faites FileNewWeb Component. Assurez-vous que 'Create New WAR Module in Application' soit coché et donnez par exemple comme nom, ProgrammezServiceWar (Fig. 1). Cliquez ensuite sur le bouton Edit. Nous allons sélectionner les fichiers qui doivent faire partie du composant Web. Ces fichiers sont au nombre de quatre.

- ProgrammezServiceIF.class
- programmezServiceImpl.class
- ProgrammezService.wsdl
- mapping.xml

Sélectionnez les dans la partie supérieure de la fenêtre et cliquez sur 'Add'. La partie inférieure de votre fenêtre doit alors être la même que celle de la figure 2. Cliquez sur Ok puis sur Next. Dans le dialogue suivant cochez Web Services Endpoint. Puis cliquez sur 'Next'. Dans le dialogue suivant sélectionnez, à partir des boîtes listes déroulantes, le document WSDL et le document de mapping (Fig. 3). Cliquez sur 'Next'.

Dans le dialogue suivant sélectionnez la classe d'implémentation depuis la boîte liste déroulante (voir ci-contre) (Fig. 4). Cliquez sur Next. Dans le dialogue suivant, il s'agit de définir le point terminal du service (Service Endpoint Interface). Comme nous n'avons écrit qu'une interface il n'y a pas à se tromper. Remplissez chacune des boîtes listes déroulantes avec l'unique choix qu'elles proposent. (voir ci-contre) (Fig. 5). Puis cliquez sur 'Next' et enfin sur Finish.

Dans l'outil de déploiement, sélectionnez ProgrammezServiceApp puis sélectionnez l'onglet 'Context Root' et donnez par exemple demo-jaxrpc (Fig. 6). Cliquez ensuite sur 'ProgrammezServiceImpl' puis sur l'onglet 'Aliases'. définissez l'alias programmez, comme montré (Fig. 7). Enfin

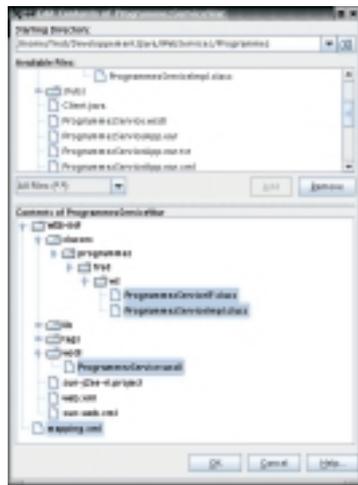


Fig.2 : Insertion des fichiers dans le composant Web.

puis depuis l'onglet Endpoint nous devons à nouveau choisir le point terminal : Dans la boîte liste déroulante, remplacez ProgrammezServiceImpl par l'alias programmez précédemment défini. (Fig. 8). Nous en avons enfin terminé. Faites une sauvegarde (Save All) puis cliquez à nouveau sur ProgrammezServiceApp. Depuis le menu allez à Tools/Verify J2EE compliance. Si vous n'avez pas fait d'erreur, tout sera Ok, il ne reste plus qu'à aller à Tools/Deploy pour déployer sur le serveur. L'opération effectuée, nous

pouvons vérifier que notre Web Service est en état. Pour cela saisissez dans un navigateur

<http://machine:3099/demo-jaxrpc/programmez?WSDL>

Si tout va bien, le navigateur va afficher le WSDL du Web Service.

5 Le client

Il est maintenant temps d'écrire un client. Celui-ci sera statique comme mentionné plus haut, ce qui signifie que sa compilation générera des stubs de manière assez semblable à ce qui se passe avec RMI (cf. Programmez! 58) Les stubs sont des classes qui prennent en charge la communication sur le réseau. Ecrire un client est aisé :

```
package client;

import programmez.fred.ws.ProgrammezServiceIF;
import programmez.fred.ws.ProgrammezService_Impl;

import javax.xml.rpc.Stub;

public class Client {
    public static void main(String[] args) {
        try {
            Stub stub = (Stub)
                (new ProgrammezService_Impl().getProgrammezServiceIFPort());
            ProgrammezServiceIF proxy =
```

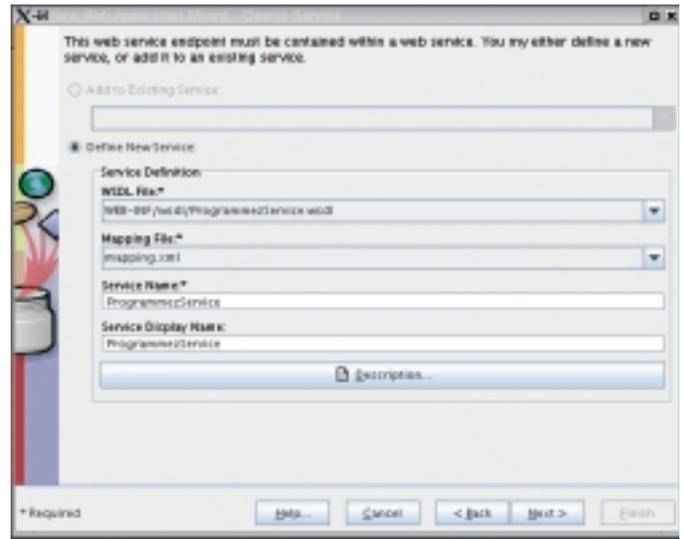


Fig.3 : Sélection des descripteurs.

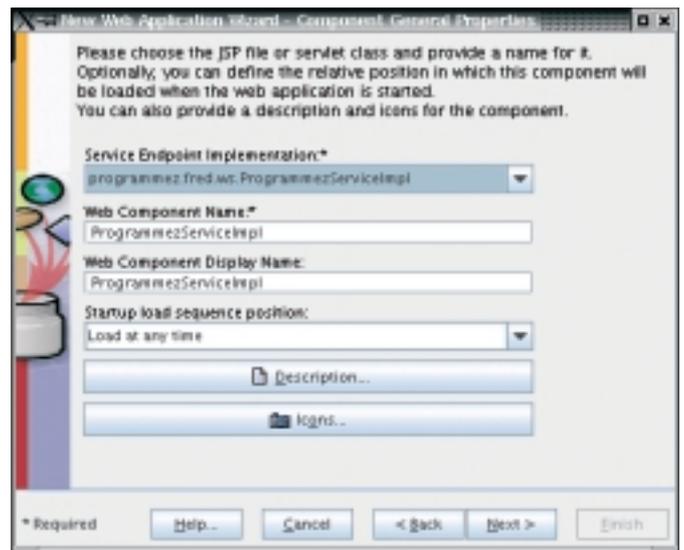


Fig.4 : Sélection de la classe d'implémentation.

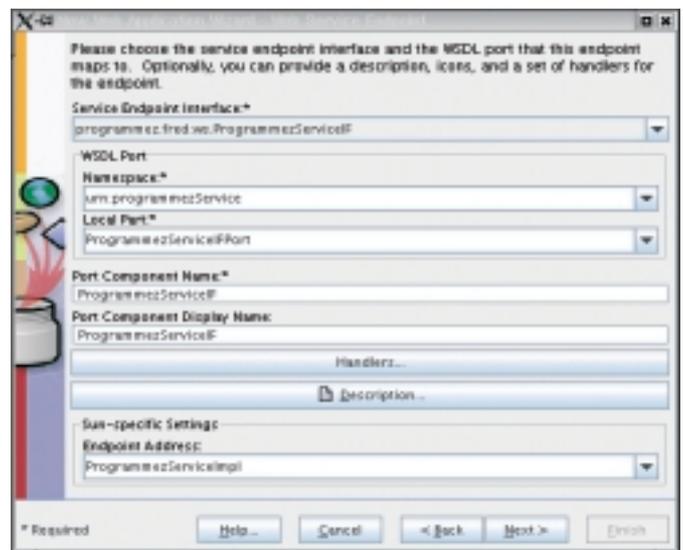


Fig.5 : Sélection du point terminal.

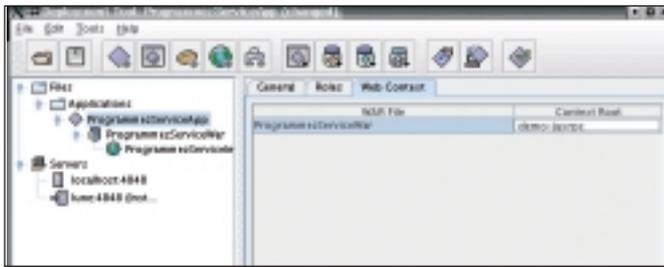


Fig.6 : Définition de la racine du Web services.

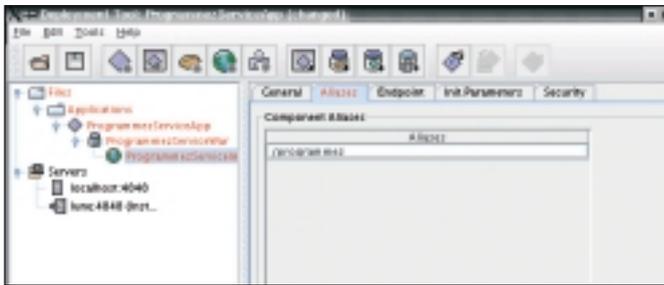


Fig.7 : Définition d'un alias pour notre Web Service

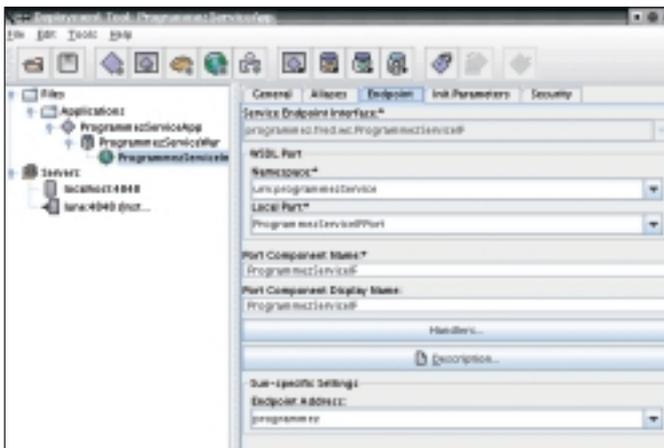


Fig.8 : Sélection du point terminal par son alias.

```
(ProgrammezServiceIF)stub;

// Et voilà :)
String result = proxy.helloWorld();
System.out.println(result);
} catch(Exception ex) {
    ex.printStackTrace();
}
}
```

Le principe est simple, Grâce aux stubs, nous créons une instance de la classe `ProgrammezService_Impl`, obtenons grâce à elle le port du service que nous transtypions sur notre interface. Tout se passe apparemment comme si tout était fait en local. Pour l'instant nous ne pouvons pas compiler ce client car la classe `ProgrammezService_Impl` n'existe pas encore. Nous devons d'abord écrire un descripteur de client. Il s'agit encore d'un fichier XML baptisé cette fois `client-config.xml` :

Un client en Python pour notre Web Service

Comme bien souvent avec Java les choses sont assez (trop ?) lourdes. Vous trouverez sur le Cd-Rom un fichier `client-javaws.py` qui fait le même travail avec 3 lignes de code que l'on pourrait réduire à une :-), mais, et surtout, sans qu'il soit besoin de compiler le moindre stub. Un client dynamique Java, ainsi que nous le verrons une prochaine fois, n'a lui non plus rien de la simplicité offerte par Python.

```
#!/usr/bin/env python
```

```
import SOAP
```

```
javaws = SOAP.SOAPProxy("http://lune:3099/demo-jaxrpc/programmez")
ns = javaws._ns("urn:programmezService")
print ns.helloWorld()
```

En dehors du constat de l'efficacité de Python, nous voyons que notre Web Service est bien indépendant du langage utilisé. Vous trouverez le module SOAP que nous utilisons, simple fichier d'une centaine de Ko, à l'adresse donnée dans l'encadré ressources.

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration
  xmlns="http://java.sun.com/xml/ns/jax-rpc/ri/config">
  <wsdl location="http://lune:3099/demo-jaxrpc/programmez?WSDL"
    packageName="programmez.fred.ws"/>
</configuration>
```

Mon serveur est sur une machine nommée lune et écoute le port 3099. Vous devez adapter cela à votre configuration. Puis nous devons compiler ce fichier, le serveur étant actif, avec l'outil à tout faire `wscmpile`, comme ceci :

```
wscmpile -gen:client -d stubs -classpath . client-config.xml
```

Attention, contrairement à `javac wscmpile` ne crée pas de répertoires. Vous devez donc créer le répertoire `stubs` auparavant. A la fin de l'opération, le répertoire `stubs` est rempli d'une foule de classes... les stubs :-). Ce répertoire `stubs` doit désormais être pointé par votre `CLASSPATH`. A partir de quoi vous pouvez compiler le client.

```
javac -cp $CLASSPATH -d . Client.java
```

Il n'y a plus qu'à essayer notre Web Service à café :

```
java client.Client
```

Dans un prochain article à venir, nous approfondirons les Web Services dans le cadre de J2EE. Utilisation d'un bean Session comme Web Service, invocation dynamique, ou mise en place d'un Web Service avec état sont autant de sujets passionnants. A bientôt, si la machine virtuelle nous prête cycle de vie :-)

■ Par **Frédéric Mazué** - fmazue@programmez.com

Intéropérabilité entre Java et C# à l'aide d'IIOPChannel



.NET Remoting est un bus de communication générique de la plate-forme .NET, permettant de distribuer des services et de les invoquer à distance. Son architecture extensible supporte plusieurs protocoles et formats d'encodage pour le transport des messages via la notion de Channel. Nous allons vous en expliquer deux: TcpChannel (basé sur DCOM) et IIOPChannel (basé sur IIOP). Ce dernier canal nous permettra d'invoquer, à partir d'un client Java, des méthodes d'un service serveur écrit en C#.

L'intéropérabilité : le nouveau fer de lance de Microsoft

L'intéropérabilité est le nouveau fer de lance du marketing Microsoft. C'est en tous cas ce que nous déclare Bill Gates qui a publié, début février 2005, un message intitulé "Building Software That Is Interoperable By Design". Il s'agit d'un vaste programme, car la problématique dépasse le simple cas du programmeur qui a besoin d'intégrer des composants développés par des outils différents (comme le C# et Java). Par exemple, promettre aux entreprises une plus grande intéropérabilité, ce serait aussi publier les spécifications du format ".doc" utilisé par Microsoft pour sa suite bureautique, ou encore abandonner les brevets qui protègent la lecture et l'écriture de n'importe quel autre format d'échange de données...

Nous allons nous intéresser à l'utilisation de services proposés par un composant .Net Remoting. Tout d'abord, nous allons déterminer comment développer un canal d'échange TCP entre applications .Net, puis comment créer un canal d'échange entre un serveur codé en C# et un client codé en Java. Ceci est évidemment primordial si vous désirez récupérer en Java, à moindre coût, des objets déjà développés en .Net (et inversement).

Si vous êtes familier de RMI, sachez que les différences entre Java RMI et .NET Remoting sont finalement assez négligeables, mis à part que l'implémentation du serveur est faite dans une classe du côté de .Net remoting. Si un client Java via RMI ne peut dialoguer directement avec un service .NET et remoting, c'est simplement parce qu'il n'utilise pas le même langage de définition de service. RMI emploie Java, .Net remoting le CLR. Il sera donc nécessaire de les mettre d'accord de façon à ce qu'ils utilisent de manière commune l'IDL de CORBA.

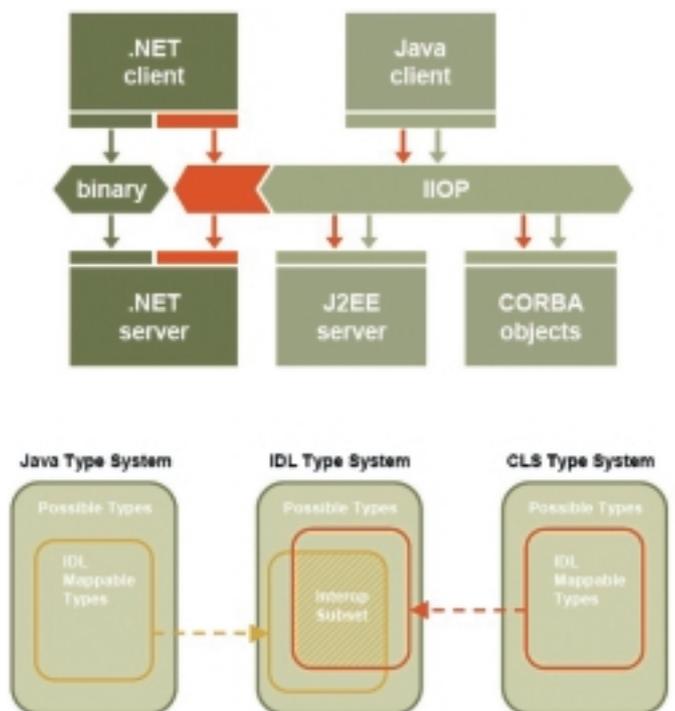


Schéma d'interopérabilité IIOP.NET

	.NET Remoting	Java Remote Method Invocation (RMI)
Proxy	Dynamique	Dynamique
Squelettes	Intégré au Framework	Intégré au Framework
Plate-forme	Windows (Linux avec Mono mais avec restrictions).	Multiples
Langages supportés	C#,VB, etc. (multiples)	Java
Objet distribué	Classes	Interfaces Remote
Configuration	Fichier XML	System Property
Protocoles de type	Channels	SocketFactoryImpl
Activation	SingleCall, Singleton	API coté serveur Activable objects
Protocoles existants	HTTP,TCP, SOAP, IIOP, etc.	JRMP, IIOP, ORMI , etc.
Gestion d'erreur	Exceptions Remote	Exceptions Remote

Présentation générale de .Net Remoting

.Net Remoting remplace le modèle DCOM. Du côté du serveur vous devez créer une application hôte chargée d'instancier le composant et de se tenir à l'écoute des requêtes. L'objet sera accessible à distance, via un canal spécifique, comme HTTP ou bien TCP. Avec HTTP, le flux utilisé est le protocole SOAP, ce qui revient à dire que toutes les communications seront sérialisées en XML. Avec TCP, le flux est binaire. Retenons simplement que TcpChannel est efficace mais n'est pas sécurisé, tandis que HttpChannel peut utiliser une couche SSL mais est plus lent. Ensuite, deux modes d'instanciation sont possibles "Singleton" et "SingleCall". Avec Singleton il n'existera qu'une seule instance de l'objet à la fois. Autrement dit, tous les messages clients sont traités par cette instance unique d'objet, ce qui permet de gérer l'état entre deux requêtes. Avec SingleCall, une nouvelle instance de l'objet est créée

Système d'exploitation : Windows XP et Linux
Environnement : Java SDK, Visual Studio.NET 2003, Mono
Langages : C#, Java
Besoin : Vous devez télécharger et compiler IOP.NET

pour chaque message d'un client, ce qui implique qu'il ne sera pas envisageable de gérer l'état entre deux requêtes.

Le programmeur, pour générer un objet distribué, encodera d'abord l'interface (IRemoteCom), et ensuite l'implémentation du serveur dans une classe (RemoteServerObject). C'est la classe de base MarshalByRefObject qui est ici utilisée pour créer un objet sérialisé par référence. Un paramètre de retour sera passé d'un domaine source vers un domaine cible ("marschallé"). Deuxièmement, l'enregistrement de l'objet est réalisé en appelant la méthode RemotingConfiguration.RegisterWellKnownServiceType(). Vous devez lui passer plusieurs paramètres, à commencer par le type de la classe distribuée typeof(RemoteServerObject), le nom du serveur (ici "Exemple_Programmez" : l'ensemble protocole : //hote:port/nom formant une URL, avec "protocole" désignant une des Channels enregistrés comme tcp ou http), et enfin, le mode d'activation de l'objet(WellKnownObjectMode.SingleCall).

```

Serveur_Canal_TCP.cs
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
namespace RemoteServer
{
    public interface IRemoteCom
    {
        string TraitementServeur(string message);
    }
    public class RemoteServerObject : System.MarshalByRefObject,IRemoteCom
    {
        public RemoteServerObject()
        {
            Console.WriteLine("Objet distant activé");
        }
        public string TraitementServeur(string message)
        {
            return ("LE SERVEUR A RECOLTE VOTRE PRENOM QUI EST : " + message);
        }
    }

    class RemoteServer
    {
        private int mPort;

        public RemoteServer()
        {
            mPort = 1010;
        }
    }
}

```

```

static void Main(string[] args)
{
    RemoteServer rs = new RemoteServer();
    try
    {
        TcpServerChannel channel = new TcpServerChannel(rs.mPort);
        ChannelServices.RegisterChannel(channel);
        RemotingConfiguration.RegisterWellKnownServiceType(
            typeof(RemoteServerObject),"Exemple_Programmez",
            WellKnownObjectMode.SingleCall);
    }
    catch(Exception e)
    {
        Console.WriteLine(e.Message);
    }

    System.Console.WriteLine("Serveur distant à l'écoute sur le port:{0}...", rs.mPort);
    System.Console.WriteLine("Appuyez sur une touche pour sortir",
rs.mPort);
    System.Console.ReadLine();
}
}
}

```

Le client récupérera un "proxy". C'est-à-dire un représentant de l'objet distant qu'il lui suffira d'invoquer pour appeler par délégation une méthode de l'objet distribué (via les mécanismes sous-jacents d'invo-cation, de transport, et de marshalling).

```

Client_Canal_TCP.cs
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
namespace RemoteServer
{
    public interface IRemoteCom
    {
        string TraitementServeur(string message);
    }

    class RemoteClient
    {
        static void Main(string[] args)
        {
            try
            {
                TcpClientChannel chan = new TcpClientChannel();
                ChannelServices.RegisterChannel(chan);
                IRemoteCom obj = (IRemoteCom)Activator.GetObject(typeof(
IRemoteCom),
                "tcp://192.168.0.39:1010/Exemple_Programmez");
                Console.WriteLine(obj.TraitementServeur("Xavier"));
            }
            catch(Exception e)

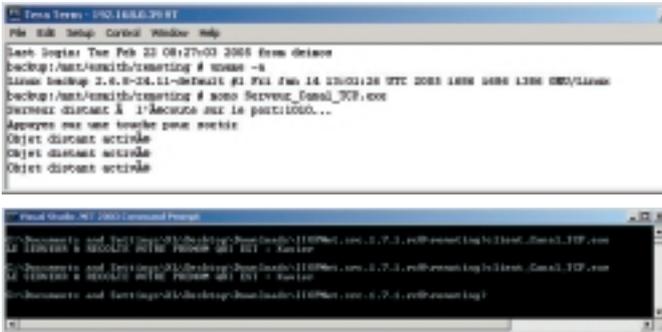
```

```

    {
        Console.WriteLine(e.Message);
    }
}
}
}
}

```

Le serveur (192.168.0.39) tourne sous Linux SuSe 9.2 via Mono (la version installée via YAST). Le client est exécuté sur une machine Windows XP avec SP2. Attention, tant sous Linux que sous Windows, vous devez ouvrir le port 1010 employé (mais vous pouvez choisir n'importe quel autre port à votre convenance).



Exécution du serveur NET REMOTING TcpChannel sous Linux et du client TcpChannel sous Windows XP.

Examinons maintenant l'interopérabilité entre une application serveur écrite en C# et une application Java.

1 Installons IIOp.NET

Microsoft a conçu la transmission par "channel" en le rendant complètement paramétrable (vous pouvez ainsi créer un fichier de configuration XML qui sera chargé par vos applications comme ceci : RemotingConfiguration.Configure(MonFichierConfig.xml). Par l'intermédiaire du canal vous implémentez en fait un sérialisateur / désérialisateur qui formate les données selon un protocole donné. La souplesse est bien au rendez-vous, car il est concevable de créer son propre "channel". IIOp (Internet Inter-ORB Protocol) est un protocole/canal permettant à des implémentations indépendantes d'ORB (Object Request Broker) d'interopérer entre elles au moyen de TCP/IP.

Vous devez disposer du SDK Java (>= 1.4), d'un compilateur C# (nous avons utilisé celui de Microsoft) et bien entendu du framework 1.0 ou 1.1. Après avoir décompressé le fichier IIOpNet.src.1.7.1.rc0.zip nous avons compilé les répertoires IIOpChannel et CLSToIDLGenerator sous Windows avec nmake (après avoir déplacé les fichiers ir.idl et orb.idl du SDK Java sous l'arborescence IDL). Ce qui donne l'exécutable CLSIDGenerator.exe et l'assemblage IIOpChannel.dll. Nous avons ensuite créé un répertoire remoting sous le répertoire racine d'IIOpNet et y avons recopié les deux fichiers précités. Enfin nous avons créé un fichier build.bat, chargé de compiler notre serveur et notre client IIOp dont la première ligne est la suivante :

```
cd C:\Documents and Settings\XL\Desktop\Downloads\IIOpNet .src.1.7.1.rc0\remoting
```

Notez bien qu'il existe plusieurs implémentations d'IIOpChannel comme

Remoting, Corba et Janeva. IIOp.NET a l'avantage d'être complète et sous licence LGPL (GNU Lesser General Public licence).

2 Créons un service distribué

Nous avons suivi le même schéma que par le biais du canal TCP, mis à part que c'est un canal IIOp qui est créé :

```
ChannelServices.RegisterChannel(new IioChannel(rs.mPort));
```

La ligne using Ch.Elca.Iioip; fait appel à l'assemblage IIOpChannel.dll (nous compilons le serveur par le biais de csc /r:IIOpChannel.dll Serveur_IIOp.cs, deuxième ligne de notre fichier build.bat).

Voici le code du serveur :

```

Serveur_IIOp.cs
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using Ch.Elca.Iioip;

namespace Serveur.IIOp {
    public class Composant : MarshalByRefObject {
        public string TraitementServeur(string message)
        {
            Console.WriteLine("Objet distant activé");
            return ("LE SERVEUR A RECOLTE VOTRE PRENOM QUI EST : " + message);
        }
    }

    class RemoteServer
    {
        private int mPort;

        public RemoteServer()
        {
            mPort = 1010;
        }

        public static void Main(string[] args) {
            RemoteServer rs = new RemoteServer();
            ChannelServices.RegisterChannel(new IioChannel(rs.mPort));
            RemotingServices.Marshal(new Composant(), "Exemple_Programmez");
            System.Console.WriteLine("Serveur distant à l'écoute sur le port:{0}
            ...", rs.mPort);
            System.Console.WriteLine("Appuyez sur une touche pour sortir");
            System.Console.ReadLine();
        }
    }
}

```

3 Créons les descriptions IDL

Qu'est ce que l'IDL (Interface Definition Language) ? Il s'agit du langage CORBA défini par l'OMG (Object Management Group) décrivant les interfaces d'un composant et ceci, indépendamment des langages de programmation C# ou Java. En effet, selon la norme CORBA le déve-

loppement d'applications distribuées sur des plates-formes hétérogènes nécessite une stricte séparation entre les interfaces et leurs implémentations. IDL définit les interfaces des méthodes sur un objet distribué (types des paramètres et résultats des méthodes).

Si vous voulez que le composant serveur soit accessible à partir d'un client java, vous devez commencer par générer l'IDL de l'assemblage serveur. Nous avons compilé précédemment CLSIDLGenerator.exe qui va nous servir maintenant à générer les IDL :

```
CLSIDLGenerator.exe -o java Serveur.IIOP.Composant Serveur_IIOP.exe cd Java
```

C'est dans ce dossier Java que nous créerons le code du client. L'utilitaire y a déjà créé une série de sous-répertoires (Ch/Elca/lio et /Serveur/IIOP/) qui contiendront les codes IDL (ainsi que java/ Predef.idl reprenant l'interface de l'exception remontée par le composant serveur).

4 Mappage des codes IDL

Nous devons maintenant indiquer au client Java ce qu'il peut utiliser comme interface. Comme IDL est indépendant du langage il y a lieu de mapper ce code IDL en code Java. Le SDK fournit l'utilitaire idlj, un compilateur IDL/Java, pour générer le code Java nécessaire :

```
idlj Serveur\IIOP\*.idl
idlj Ch\Elca\liop\*.idl
```

5 Code client et exécution

Le code client comporte la ligne import Serveur.IIOP.Composant; afin d'incorporer l'interface du composant serveur. CTX signifie "call context", et a pour but d'ajouter des informations additionnelles à l'entête d'un message.

La compilation est lancée comme ceci :

```
javac -classpath . Ch\Elca\liop\*.java
javac -classpath . Serveur\IIOP\*.java
javac -classpath . *.java
```

Voici le code de Client_Java_IIOP.java :

```
import java.util.Properties;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;
import Serveur.IIOP.Composant;

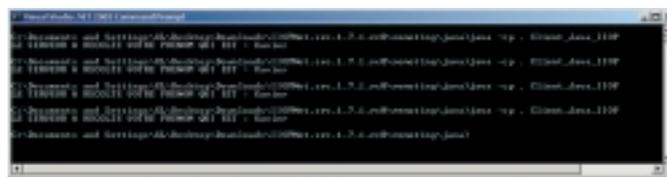
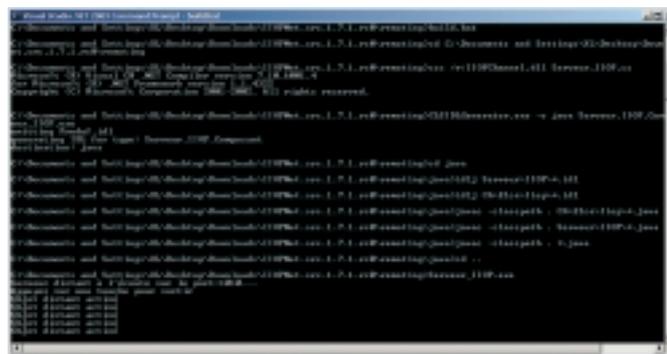
public class Client_Java_IIOP {
    public static void main(String[] args) {
        Properties props = new Properties();
        props.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.cosnaming.CNCtxFactory");
        props.put(Context.PROVIDER_URL, "iiop://localhost:1010");

        try {
            Context ctx = new InitialContext(props);
            Object objRef = ctx.lookup("Exemple_Programmez");
            Composant DotNetComponent = (Composant)PortableRemote
```

```
Object.narrow(objRef, Composant.class);
        System.out.println(DotNetComponent.TraitementServeur("Xavier"));
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}
```

Au final, après avoir exécuté le serveur vous lancerez le client, via une commande du style

```
java -cp . Client_Java_IIOP
```



Exécution du serveur IIOp.NET et du client sous Windows XP.

Conclusion

Avec IIOp.NET, nous utilisons ce que l'on appelle un "couplage fort" pour accéder à un composant distant. Il existe d'autres techniques pour créer une interopérabilité entre Java et .Net. Par exemple, des sociétés commercialisent des ponts RMI/.NET Remoting. Mais l'inconvénient de ces ponts est qu'il est nécessaire de mettre à jour ces logiciels propriétaires lorsque l'une ou l'autre API (Java ou .Net) évolue.

Au niveau du "couplage lâche", on parle souvent des services Web. Malheureusement, ceux-ci sont incapables de gérer des transactions distribuées, sans compter certains problèmes de compatibilité entre normes, et une moindre rapidité (messages XML non binaires). IIOp.NET apparaît donc comme une excellente solution, qui plus est, libre.

■ Xavier Leclercq - Xavier.Leclercq@programmez.com

Liste des liens :

Bill Gates, "Building Software That Is Interoperable By Design": <http://www.microsoft.com/mscorp/execmail/2005/02-03interoperability.asp>
 Nouveau site Microsoft consacré à l'interopérabilité : <http://www.microsoft.com/windowsserversystem/interop/default.msp>
 IIOp.NET <http://iiop-net.sourceforge.net/>
 Remoting.Corba <http://remoting-corba.sourceforge.net/>
 Janeva <http://www.borland.com/janeva/>

Refactoring C# avec Resharper

Une mise en garde : si vous essayez Resharper, vous le trouverez vite indispensable ! Une fois installé et configuré fini les erreurs de compilation, vous disposerez d'une barre verticale sur le côté de votre page de code qui vous signale les erreurs. Resharper est un outil payant, mais qui vaut son pesant d'or en terme de qualité de développement.

Vous avez dit refactoring ?

En tant que développeur, nous espérons que vous êtes persuadé que l'amélioration de la qualité diminue les coûts de développement. Ajoutons que la meilleure manière d'améliorer la productivité et la qualité consiste à réduire le temps passé à récrire le code. Seulement voilà : avec les techniques modernes de développement, comme l'eXtreme Programming (XP), l'application est en continuelle phase de modification dès le départ du projet. Le développement est itératif et chaque itération produit une release fonctionnelle qui sera livrée au client pour feedback. En pratique, les développeurs écrivent d'abord les tests (des classes de tests qui appellent les classes fonctionnelles), puis développent le code le plus simple qui les satisfera. Il est par conséquent obligatoire de compiler et de "nettoyer" régulièrement le code (tous les développeurs profitant du code le plus actuel). Cette refactorisation est une opération de maintenance qui consiste à retravailler le code source, non pas pour ajouter une fonctionnalité supplémentaire, mais pour améliorer sa lisibilité.

Le refactoring se pratique à plusieurs niveaux. Si vous ne voulez pas que votre code source tende à devenir de plus en plus complexe au fur et à mesure de son élaboration, vous devez faire en sorte de supprimer la redondance, simplifier les algorithmes, limiter le nombre et la complexité des classes. Vous pouvez refondre le design en modifiant la hiérarchie de classes composant l'application.

Vous pouvez renommer le nom des classes et des méthodes pour que leurs rôles deviennent plus clair. Vous pouvez supprimer le code mort de la version courante (partie du code non appelée par une autre partie du programme), car il est préférable d'utiliser un outil de contrôle de version comme CVS pour l'archiver. Vous pouvez reformater le code source pour qu'il présente un aspect plus propre, par exemple par indentation, c'est à dire par ajout de caractères de tabulation, ou par alignement des blocs de codes.

1 Comment compiler sans erreurs ?

Resharper n'est pas le seul outil dans sa catégorie (Coderush ou Refactory sont deux produits concurrents dont les liens figurent en fin d'article), mais il s'agit probablement du logiciel le plus facile d'apprentissage. C'est important : vous n'avez pas nécessairement le temps d'apprendre à maîtriser ce genre d'extension qui est censé vous faire gagner de l'argent et non le contraire (nous pensons par exemple au difficile apprentissage du langage macro-type de CodeRush). Avec Resharper tout se déroule assez intuitivement, mais vous devrez quand même connaître par coeur les raccourcis clavier si vous ne voulez pas toujours faire appel aux divers menus.

Resharper est disponible en version d'essai de 30 jours sur le site de l'éditeur,

Système d'exploitation : Windows XP

Environnement : Visual Studio.NET 2003

Langages : C#

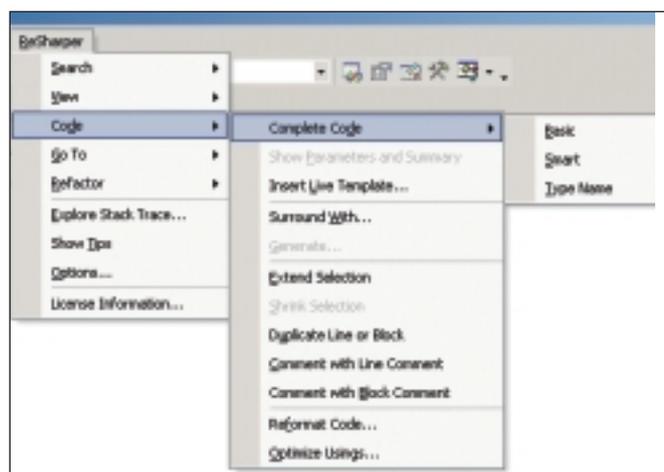
Besoin : Vous devez télécharger une version d'essai de Resharper ou l'acheter.

JetBrains. Il suffit de remplir un court questionnaire pour recevoir quelques minutes plus tard, par email, une clé de licence d'essai.

Cet outil, dont la version testée est la v1.0.115.6, se présente sous la forme d'un addin à Visual Studio 2003. Après le téléchargement (~ 7Mo) et l'installation, vous devez encoder au premier démarrage de VS.NET votre nom d'utilisateur (en respectant la casse), et votre clé d'utilisateur reçue par courrier électronique.

Lorsque vous chargez un projet Resharper l'analyse, affichant la boîte de dialogue "loading solution". Vous vous trouvez alors face à votre environnement habituel VS.NET à deux détails près : d'abord un menu Resharper apparaît en haut de votre écran et ensuite vous disposez d'une barre colorée verticale à droite de votre code.

Celle-ci sera teintée en rouge si l'outil détecte à la lecture de votre code



qu'une erreur de compilation surviendra. Cette visualisation des erreurs est géniale car elle nous fait gagner énormément de temps en nous évitant des compilations inutiles ! Par exemple, si nous tapons par distraction "thus.ClientRectangle. Left" en lieu et place de "this.ClientRectangle.Left", un carré rouge apparaît au sommet de cette barre verticale. Si nous pointons notre souris au-dessus, un popup nous indiquera "Analysis completed. 1 error found.". Si vous voulez localiser l'erreur rien n'est plus facile : cette même barre affichera un tiret rouge

à l'emplacement de la ligne erronée. L'erreur est elle aussi mise en évidence en rouge. En invoquant le raccourci Ctrl + Space, le logiciel vous propose de compléter les deux premières lettres "th" par "is". En appuyant sur tab, le tour est joué. Aussitôt le carré rouge se transforme en carré vert.



Si le carré est orange, vous avez affaire à différents messages d'avertissement (warnings). Par exemple, vous avez encodé par distraction plusieurs directives using redondantes. Vous cliquez sur un des tirets orange et vous pointez alors sur la ligne using en question. Une petite ampoule est affichée. Celle-ci, entre parenthèses, est aussi présente dans le cas d'une erreur fatale à la compilation, mais pour résoudre le "thus" en "this" nous avons choisi la complétion de code. Tenez-vous bien : si vous cliquez sur cette ampoule, le logiciel vous propose de nettoyer les usings redondants à votre place ("optimize usings").

Et le warning disparaît ! Époustouflant. Autre exemple, si vous avez la déclaration "private static void Main(string[] args)", mais que l'argument args n'est jamais utilisé dans le code, vous obtiendrez "parameter 'args' is assigned but never accessed". Si vous corrigez en "private static void Main()", le drapeau coloré de la compilation passera au vert. Attardons-nous maintenant à la facilité de navigation qui est de notre avis le deuxième point fort de ReSharper. Partons de l'extrait de code source suivant :

```
...
DESCryptoServiceProvider encode = new DESCryptoServiceProvider();

encode.IV = ASCIIEncoding.ASCII.GetBytes("A1C2D3E4");
encode.Key = ASCIIEncoding.ASCII.GetBytes("ABCDEXYZ");

CryptoStream crStream = new CryptoStream(stream, encode.Create
Encryptor(), CryptoStreamMode.Write);

byte[] données = ASCIIEncoding.ASCII.GetBytes("Ce message est secret.");
crStream.Write(données, 0, données.Length);
...
```

Si nous nous dirigeons au-dessus de "données" de la ligne "crStream.Write(données, 0, données.Length);" et que nous demandons à naviguer vers sa déclaration en appuyant sur CTRL + B (ou via le menu contextuel), ReSharper pointe vers "byte[] données = ASCIIEncoding.ASCII.GetBytes("Ce message est secret.);". Allons plus loin : avec Ctrl-Shift-T vous vous retrouvez sur son type dans votre source, ou sur le browser d'objets de Visual Studio .NET 2003 si nous avons affaire à une classe du framework ou d'un assemblage que nous avons ajouté en référence.

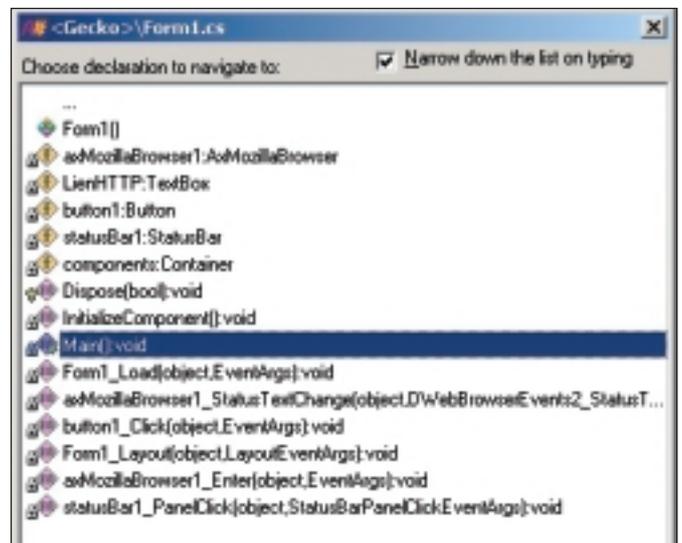
Pointons tout de suite du doigt une possibilité de refactoring : si vous sélectionnez le texte "Ce message est secret." vous pouvez demander au logiciel d'introduire une nouvelle variable. Idem avec le texte "A1C2D3E4" et "ABCDEXYZ". Au bout de trois clics et après avoir encodé les noms de chaque variable à l'aide d'une boîte de dialogue nous obtenons :

```
..
string charsIV = "A1C2D3E4";
encode.IV = ASCIIEncoding.ASCII.GetBytes(charsIV);
string charsKey = "ABCDEXYZ";
encode.Key = ASCIIEncoding.ASCII.GetBytes(charsKey);
CryptoStream crStream = new CryptoStream(stream, encode.Create
Encryptor(), CryptoStreamMode.Write);

string charsMessageSecret = "Ce message est secret.";
byte[] données = ASCIIEncoding.ASCII.GetBytes(charsMessageSecret);
..
```

2 Comment naviguer très rapidement dans le code ?

Ceci dit, les possibilités de navigation ne s'arrêtent pas là : nous pouvons nous diriger d'erreur en erreur (F2 / shift F2), aller vers la classe de base avec Ctrl-U et sur les classes fils avec Ctrl-Alt-B, employer Alt-Up pour atteindre la méthode précédente et Alt-Down pour atteindre la méthode suivante. ReSharper nous offre la possibilité de rechercher rapidement une méthode ou une propriété de votre code source en listant la structure (raccourci Ctrl-F12). Il s'agit d'une méthode visuelle : un pop-up s'affiche, ensuite vous repérez votre méthode et vous cliquez pour que l'affichage du code source pointe sur celle-ci. Lorsque le code source est lourd de centaines de lignes ou plus, c'est très efficace.



Un autre raccourci clavier est aussi particulièrement utile : si vous êtes dans un fichier source tierce, en tapant la combinaison Ctrl-N une fenêtre popup s'ouvrira. Vous pouvez y encoder le nom de votre classe et une liste se mettra à jour automatiquement. Si vous choisissez une entrée de la liste vous retrouvez son code source affiché dans l'éditeur. Et si vous utilisez Ctrl-Shift-N, ce n'est pas une liste de classes qui est affichée mais bien la liste de fichiers de votre solution.

3 Différentes techniques de Refactoring

Revenons aux fonctions de refactoring (nous avons déjà vu la création de variable). Vous pouvez changer de nom. Ici aussi il s'agit d'une fonctionnalité très puissante. Songez à toutes ces occasions où


```

}

public int traduction
{
    get
    {
        BabelFishService fish = new BabelFishService();
        int coderetour;
        try
        {
            textBox3.Text = fish.BabelFish(textBox1.Text, this.textBox2.Text);
            textBox4.Text = "Traduction réussie";
            coderetour = 0;
        }
        catch (WebException)
        {
            textBox4.Text = "Echec de la traduction (service Web indisponible ?)";
            coderetour = 1;
        }
        return coderetour;
    }
}

```

Mais nous ne sommes pas au bout de nos surprises. Imaginons que nous désirions isoler le code du try et celui du catch. Sélectionnons les trois lignes de codes du catch puis le menu contextuel refactor / extract method. Idem dans le cas du try. Nous créons de cette manière très rapidement deux nouvelles méthodes :

```

public int traduction()
{
    BabelFishService fish=new BabelFishService();
    int coderetour;
    try
    {
        coderetour = Traduction_Réussie(fish);
    }
    catch(WebException )
    {
        coderetour = Traduction_Echec();
    }
    return coderetour;
}

private int Traduction_Echec()
{
    int coderetour;
    textBox4.Text="Echec de la traduction (service Web indisponible ?)";
    coderetour = 1;
    return coderetour;
}

private int Traduction_Réussie(BabelFishService fish)
{
    int coderetour;

```

```

textBox3.Text=fish.BabelFish(textBox1.Text,this.textBox2.Text);
textBox4.Text="Traduction réussie";
coderetour = 0;
return coderetour;
}

```

4 Les Templates

Les templates ou modèles servent à créer rapidement des blocs de codes. Par exemple, si vous voulez créer une itérative, vous tapez sur CTRL + J, et un menu de templates s'affiche. Si vous cliquez sur "loop" vous créez un squelette de boucle for :

```

for (int i = 0; i < UPPER; i++)
{
}

```

Resharper montre toute sa puissance lorsque vous déclarez par exemple une liste comme "ListeDeNombres" avant d'insérer un "live template" foreach, car il génèrera automatiquement "foreach (int nombre in ListeDeNombres)". Et si jamais vous avez plusieurs listes possibles un menu déroulant vous les proposera.

```

int [ ] ListeDeNombres = new int [] {6,7,8,9};

```

Automatiquement, le logiciel détecte une possible énumération et vous la propose :

```

foreach (int nombre in ListeDeNombres)
{
}

```



D'autres modèles existent comme l'assertion (asrt), l'itération d'un dictionnaire, etc.

Pour conclure

Nous sommes extrêmement emballés par cet outil. Avec Resharper, vous serez sans conteste plus productif et le retour sur investissement sera immédiat. Visual Studio.NET 2005 ira dans ce sens, mais comme Resharper n'en est qu'à sa version 1.0 on peut déjà parier qu'il conservera encore longtemps une bonne longueur d'avance. Si vous développez en C#, vous pouvez le tester et pourquoi pas l'acheter les yeux fermés : vous ne serez pas déçu.

Liste des liens :

Coderush : <http://www.devexpress.com/products/NET/Coderush/>

C Sharp Refactory : <http://www.xtreme-simplicity.net/CSharpRefactory.html>

Téléchargement de ReSharper 1.0 : <http://www.jetbrains.com/resharper/download/>

Documentation : <http://www.jetbrains.com/resharper/documentation/index.html>

FAQ : <http://www.jetbrains.com/support/resharper/index.html>

■ Xavier Leclercq

Linux : monter un périphérique avec les droits de l'utilisateur

Au démarrage, votre système Linux monte les périphériques de fichiers qui peuvent l'être, comme les disques durs. En principe les périphériques dont le support de données peut être manipulé par l'utilisateur (Lecteur de Cd-Rom, lecteur de disquette) ne sont pas montés au boot, car le montage échoue en l'absence de CD ou de disquette. L'utilisateur doit monter manuellement avec la commande mount. Par exemple sur une Suse :

```
mount /media/cdrom
```

Il est fréquent que, par défaut, seul le super-utilisateur puisse invoquer cette commande, pour des raisons de sécurité. Si vous êtes le seul utilisateur de votre système vous préférerez pouvoir monter sans devoir être root. Pour cela, modifiez comme suit une ligne du fichier /etc/fstab

```
/dev/cdrom /media/cdrom auto ro,noauto,exec 0 0
```

qui devient :

```
/dev/cdrom /media/cdrom auto ro,noauto,user,exec 0 0
```

L'option noauto de la ligne signifie que le périphérique n'est pas monté automatiquement lorsque le système démarre ou, c'est la même chose, lorsque la commande mount -a est invoquée. Supprimez noauto si vous laissez toujours un CD dans votre lecteur. Celui-ci sera alors automatiquement monté au démarrage. Bien entendu le principe peut être appliqué aux lecteurs de disquettes. Exemple :

```
/dev/fd0 /media/floppy auto noauto,user,sync 0 0
```

Enfin l'option auto de la ligne ne concerne en rien notre problème mais demande au système de déterminer lui-même à quel système de fichiers il a affaire.

Mise à jour SuSe 9.0

L'astuce précédente ne fonctionne plus depuis que vous avez fait votre mise à jour de SuSe 8.0 vers SuSe 9.0 ? Pour une raison aussi bizarre qu'étrange, l'outil de mise à jour a modifié les droits d'accès aux fichiers spéciaux dits encore descripteurs de périphériques. Si par exemple vous ne pouvez pas monter de Cd-Rom, la commande :

```
chmod 777 /dev/cdrom
```

devrait régler le problème. Vous devez être root pour saisir la commande et vous devez vous attaquer au fichier de périphérique lui-même et non à un point de montage qui lui correspond. Ainsi avec une SuSe :

```
chmod 777 /media/cdrom
```

ne réglerait rien.

Firefox et les applets Java

Vous venez d'installer le navigateur Firefox dans votre Linux mais il n'affiche pas les applets Java. La raison est qu'il ne peut trouver seul le plug-in de votre JDK. Ce plug-in est normalement conçu pour Netscape mais fonctionnera très bien avec Firefox. Il réside sous le répertoire jre de votre JDK et pour l'installer dans Firefox, il suffit de créer un lien symbolique dans le répertoire plugins de ce dernier. Par exemple :

```
ln -s /usr/java/jdk1.5.0/jre/plugin/i386/ns7/libjavaplugin_oji.so /usr/firefox/plugins/
```

Bien entendu, l'astuce fonctionne également avec Mozilla.

Du Java plein la vue

Une seule chose vous manque pour écrire en Java le jeu qui tire plus vite que son ombre et qui fera date dans l'histoire des code-byte : comment faire pour obtenir un affichage en plein écran ? La chose est possible depuis Java 1.4 à condition que l'OS sous-jacent le supporte aussi, ce qui est normalement le cas de votre Windows ou de votre Linux. Vous devez vous intéresser aux classes GraphicsEnvironment et GraphicsDevice du package java.awt. Voici la recette magique :

```
import java.awt.*;
import javax.swing.*;

public class FullScreen {

    public static void main(String[] args) {
        JFrame f = new JFrame();

        GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
        GraphicsDevice gd = ge.getDefaultScreenDevice();
        if(!gd.isFullScreenSupported()) {
            System.out.println("Vous n'en aurez pas plein la vue");
            return;
        }
        try {
            f.setUndecorated(true);
            gd.setFullScreenWindow(f);
            try {
                Thread.sleep(2000);
            }
            catch (InterruptedException ex) {
                ex.printStackTrace();
            }
        }
        finally {
            gd.setFullScreenWindow(null);
            System.exit(0);
        }
    }
}
```

Si le mode plein écran est indisponible, ce que l'exemple vérifie, ce code peut encore être employé. Dans ce cas, la fenêtre sera agrandie pour recouvrir l'écran. Dans tous les cas, l'invocation de setUndecorated est requise pour faire disparaître barre et cadre de fenêtre. Ce mode plein écran permet un tracé direct, c'est à dire profitant de l'accélération matérielle (DirectX sous Windows). Une telle application peut perdre la main sur le matériel à tout moment. Pour cette raison, on veillera, entre autres, à tracer des images de la classe java.awt.image.VolatileImage. Enfin, remarquez le bloc finally qui assure de revenir à un affichage normal en cas de levée d'exception.

■ Frédéric Mazué - fmazue@programmez.com

Album photo en Flash et XML



Jamais dans toute l'histoire d'Internet, une technologie ne s'est imposée comme un standard multi-plate-forme incontournable. Et pourtant, la technologie Flash n'est pas loin de réussir cet exploit. En effet, il a fallu moins de 3 ans à Macromedia pour diffuser son plug-in sur 98% des ordinateurs connectés sur Internet.

Aujourd'hui, force est de constater que la technologie Flash s'est imposée comme le standard de sur Internet. Le format Flash est le seul capable de créer des sites contenant des graphiques vectoriels et bitmap, de l'animation, du son, de la saisie de formulaires et de l'interactivité de plus en plus avancée.

Mais les capacités de cette technologie ne s'arrêtent pas là. Depuis la version MX, Flash permet d'aller plus loin, beaucoup plus loin, dans la gestion des données en ligne et l'interconnexion dynamique avec des bases de données. Le programme que nous vous présentons avec cet article illustre parfaitement ce qu'il est possible de concevoir simplement en Flash dynamique. Couplé à un document XML externe, le programme Flash développé permettra l'intégration d'un album photo dynamique et évolutif, le tout diffusable sur un simple site Internet.

Le mystère Flash

Pour beaucoup, Flash est une petite boîte noire figée, fermée et mystérieuse. Cette approche est faussée par le fait simple que Flash n'était pas un environnement dédié au développement et à la programmation. A sa création en 1995, le logiciel FutureSplash (l'ancêtre de Flash), ne proposait alors qu'une plate-forme de création et d'animation vectorielle pour les designers émergeant avec l'avènement d'Internet. Aucune

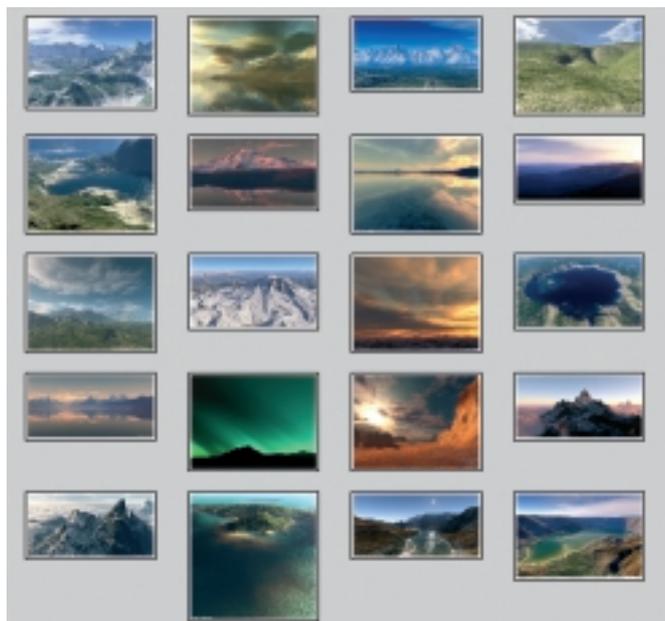


Figure 2 : le duo Flash et XML permet de concevoir des applications Web dynamiques, souples et interactives, comme ce simple exemple d'album photo virtuel sur Internet

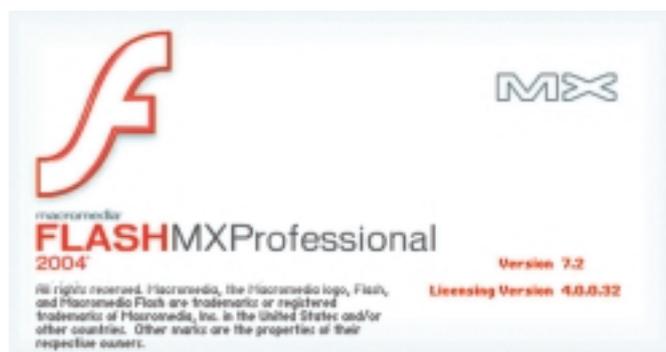


Figure 1 – Macromedia Flash MX 2004 – version 7.2

interactivité n'était alors possible. Mais avec l'apparition du langage de script, ActionScript (qui reprend la syntaxe ECMA du langage C ou du JavaScript), Flash a pu évoluer progressivement vers un environnement de développement très intéressant. La dernière version en date, Flash MX 2004 (version 7.2) propose ainsi un environnement permettant la programmation de contenu dynamique couplé à des bases de données en ligne.

Flash et XML, la simplicité au service du dynamisme

D'autre part, l'intégration d'un parseur XML dans l'environnement de Flash permet depuis quelques années de développer des applications online performantes et souples. Notre programme présente simplement l'interaction possible entre un document XML en ligne et une application conçue avec Flash.

L'histoire du standard XML est encore plus récente que celle de Flash. Ce protocole est né d'une observation simple : l'échange de données entre différents programmes est une tâche complexe. La plupart des systèmes qui tentent de partager des données doivent parler le même langage, mais ces données sont souvent hétérogènes, non uniformes et rarement structurées de la même façon. Ceci est logique bien entendu, il est normal que les données provenant par exemple d'un environnement boursier soient très différentes, structurellement, d'un environnement sportif comme autre exemple. La diversité des informations disponibles sur Internet explique la diversité des formats et des structures de données possibles. (Fig.2)

L'analyse XML avec Flash

Le format XML, dit aussi " extensible ", contourne ce problème en présentant une architecture à partir de laquelle des langages de communication peuvent être construits. Les développeurs accoutumés au format XML conçoivent généralement un DTD (Document Type Definition) ou schéma XML. Ce schéma, ratifié par le W3C, est une nouvelle façon de définir les langages basés sur le XML. Il sert à définir le jeu de balises utilisé par un langage.

Cependant avec Flash, le parseur XML est de type non validant, ce qui implique que Flash ne travaille pas de manière intrinsèque avec des schémas XML. Ainsi, lors de la réception par Flash des données, il est nécessaire de mettre en place un script qui vérifie et valide la cohérence des structures de données reçues.

La preuve par l'exemple

Passons à présent à l'exemple que nous proposons avec cet article. Pour éditer et compiler cet exemple, vous aurez besoin de la dernière version de Macromedia Flash MX 2004, téléchargeable en version d'évaluation complète sur le site de Macromedia :

<http://www.macromedia.com/fr/software/flash/>

Sélectionnez de préférence la version Flash MX Professional 2004. L'autre édition personnelle devrait également suffire.

Ouvrez ensuite le fichier portfolio fla. Il contient le code source du programme, éditable dans la fenêtre " Actions ".

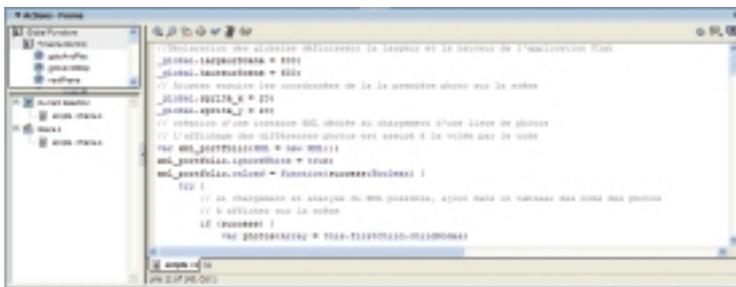


Figure 3 : La fenêtre " Actions " de l'environnement Flash permet d'éditer le code source d'un programme Flash

Tout le code source de cet exemple est disponible dans la première image du programme Flash. Pour le visualiser, il vous suffit de déployer la fenêtre " Actions " et de cliquer sur la première frame du scénario. Le programme va commencer par lire le contenu du fichier XML et l'intégrer dans un objet Flash XML. Les étapes suivantes seront donc effectuées :

- 1 • Définition d'une fonction qui sera exécutée lors du chargement des données
- 2 • Création d'un nouvel objet XML
- 3 • Ajustement du paramètre ignoreWhite sur true pour que le programme Flash ne gère pas les retours chariots ainsi que les espaces vides en tant que nœuds indépendants
- 4 • Définition d'une fonction utilisée après le chargement des données
- 5 • Chargement des données

Concrètement donc, le code commence par définir un certain nombre de paramètres liés à la taille de l'application et à la position de certains éléments du programme :

```
//Déclaration des globales définissant la largeur et la hauteur de
```

l'application Flash

```
_global.largeurScene = 800;
_global.hauteurScene = 600;
// Ajuster ensuite les coordonnées de la la première photo sur la scène
_global.sprite_x = 20;
_global.sprite_y = 20;
```

Ensuite, une instance XML du parseur Flash est définie afin de procéder au chargement du fichier XML externe. En cas de réussite de ce chargement, le programme génère un tableau de valeur intégrant les noms des photos de l'album qui seront affichées.

```
// création d'une instance XML dédiée au chargement d'une liste de photos
// l'affichage des différentes photos est assuré à la volée par le code
var xml_portfolio:XML = new XML();
xml_portfolio.ignoreWhite = true;
xml_portfolio.onLoad = fonction(success:Boolean) {
    try {
        // si chargement et analyse du XML possible, ajout dans un tableau
        // des noms des
        // photos à afficher sur la scène
        if (success) {
            var photos:Array = this.firstChild.childNodes;
            var tab_portfolio:Array = new Array();
            for (var n = 0; n<photos.length; n++) {
                tab_portfolio.push({src:photos[n].firstChild.nodeValue});
            }
            // cette fonction charge chaque photo du portfolio sur la scène
            showPortfolio(tab_portfolio);
        } else {
            // sinon, problème de chargement XML, afficher un message d'erreur
            throw new Error("problème de chargement du XML");
        }
    } catch (e_err:Error) {
        trace(e_err.message);
    } finally {
        delete this;
    }
};
// chargement du fichier XML
xml_portfolio.load("portfolio.xml");
```

Avec Flash, les données sont représentables graphiquement sous la forme d'arbre. Une fois les données chargées et analysées, l'objet XML dans lequel les données ont été chargées contient un lien unique vers la nouvelle structure de données, appelé firstChild (premier enfant). Cette propriété de l'objet XML pointe le nœud racine de l'arbre. Pour trouver le premier enfant de n'importe quel objet XML créé, il vous suffit de le référencer par this.firstChild.

L'analyse et la récupération des données XML étant terminées, il faut ensuite procéder à la création d'un tableau de données contenant les noms des fichiers JPEG externes (les photos). Ces noms de fichiers JPEG externes sont bien entendu disponibles dans le fichier XML précédemment analysé.

```
// création et affichage des movies clips sur la scène
```

```
// à partir du tableau précédemment généré via le XML
function showPortfolio(tab_portfolio:Array) {
    var nb_photos:Number = tab_portfolio.length;
    // boucle dans le tableau pour la création de chaque photo
    for (var n = 0; n < nb_photos; n++) {
        // création d'une instance de movie clip pour la gestion de la photo
        // la variable sprite_mc est un alias de l'instance du movie clip
        var sprite_mc:MovieClip = this.createEmptyMovieClip("image"
+n+"_mc", n);
        // chargement de l'image source courante vers l'instance de movie clip
        // utilisation de la classe MovieClipLoader
        mcl_loader_mc.loadClip(tab_portfolio[n].src, sprite_mc);
        // le symbole preloader de la bibliothèque de l'application
        // est lié sur la scène
        mc_preloader = this.attachMovie("preloader_mc", "preloader"+n+
"_mc", 5000+n);
        // ajustement de la propriété _xscale de la barre de chargement à 0%
        mc_preloader.bar_mc._xscale = 0;
        // ajustement d'une valeur par défaut dans la zone texte de la barre de
//chargement
        mc_preloader.progress_txt.text = "0%";
        // ajuster les coordonnées x et y du nouveau movie clip créé
        sprite_mc._x = _global.sprite_x;
        sprite_mc._y = _global.sprite_y;
        // ajuster la position de la barre de chargement
        mc_preloader._x = _global.sprite_x+0;
        mc_preloader._y = _global.sprite_y+150;
        // ajuster la disposition des clips sur 4 colonnes
        if ((n+1)%4 == 0) {
            // initialisation des positions x et y
            _global.sprite_x = 20;
            _global.sprite_y += 110;
        } else {
            _global.sprite_x += 150;
        }
    }
}
```

Avec Flash, il est souvent utile de définir une fonction permettant la visualisation d'une barre de progression pour afficher le chargement de fichier externe. L'exemple suivant utilise une instance de la classe MovieClipLoader qui, en Flash, permet de créer un objet dit "écouteur" et ainsi d'afficher, par exemple, le pourcentage de données reçues ou bien le pourcentage de données à recevoir au total.

```
// définition d'une instance MovieClipLoader et d'un objet écouteur
MovieClipLoader

var mcl_loader_mc:MovieClipLoader = new MovieClipLoader();
var mcl_listener:Object = new Object();
mcl_listener.onLoadStart = fonction() {
};
// pendant le préchargement du contenu, modification de la largeur de
la barre de progression
mcl_listener.onLoadProgress = fonction(mc_cible, data_loaded, data_total) {
```



Figure 4 : amateurs de photos virtuelles, présentez vos réalisations sous une forme dynamique et interactive

```
var pct_data:Number = Math.round(data_loaded/data_total*100);
// création d'un raccourci pour le chemin vers le movie clip de
préchargement
var mc_preloader = mc_cible._parent["preloader"+mc_cible.getDepth
()+ "_mc"];
mc_preloader.bar_mc._xscale = pct_data;
mc_preloader.progress_txt.text = pct_data+"%";
};
```

Le code suivant permet d'initialiser et de créer les vignettes de l'album photo sur la scène. Afin de mieux représenter une photo virtuelle, une bordure blanche extérieure entoure la photo. La fonction suivante montre comment générer cette bordure extérieure et afficher la photo. Elle permet aussi de gérer les interactions de la souris avec les vignettes, comme le clic gauche ou bien encore la fonction " glisser - déposer " qui permet de déplacer la photo sur la scène, ou encore le zoom (scale) simple de l'image après un clic souris sur une vignette. (Fig. : 4)

Pour aller plus loin avec Flash

Notre exemple d'analyse d'un fichier Flash est une étude de cas relativement simple. Pour concevoir une application plus complexe, Flash propose un service optimisant la performance et la simplicité lors de l'accès aux données externes : le service Flash Remoting, spécialisé dans la conception d'applications Web.

Le service Flash Remoting est une interface de programmation d'application (API) qui permet aux développeurs côté serveur de faire communiquer le code ActionScript avec des objets serveurs distants. Tout ceci permet donc aux développeurs Web de bénéficier de hautes performances, d'un haut niveau de sécurité, d'une vitesse accrue et d'une parfaite fiabilité, concurrençant avantageusement des technologies comme le trio HTML/DHTML/Javascript, plus conventionnel mais qui pourrait rapidement être remplacé par Flash Remoting.

Seul l'avenir nous confirmera cette permutation technologique du développement Web. Mais dans le domaine des nouvelles technologies, l'avenir, c'était déjà hier...

■ Laurent Jayr - media.crea@online.fr