Méthodes | Architectures | Technologies | Algorithmie | Langages | Outils | Opinions

DÉVELOPPEUR RÉFÉRENCE

LA LETTRE BIMENSUELLE DU DÉVELOPPEMENT

www.devreference.net

v2.05 | 18 décembre 2001

EDITO

Malgré une période un peu agitée puisqu'en cette fin d'année, le groupe IDG déménage de locaux, nous avons tenu à vous offrir notre fameux coktail pour les fêtes de Noël : une pincée de méthode, une rasade d'Open Source, une plâtrée de Java et une bonne dose de .NET, voilà votre Développeur Référence. Un numéro copieux donc, qui arrive un peu tardivement, avec toutes nos excuses, mais qui ne nous en empêchera pas de vous proposer un deuxième numéro ce mois-ci, bimensuel oblige. Notre déménagement ne changera pas grand chose, si ce n'est sans doute un ou deux jours d'indisponibilité du site Web. Retrouvez nos nouvelles coordonnées dans l'ours page 2, valables à partir du 24 décembre.

Pierre Tran
Rédacteur en chef

Open Source

Eclipse/WSAD

Retour d'expérience

Après avoir présenté les enjeux du projet Open Source Eclipse et l'architecture de cette plate-forme, nous montrerons la pertinence de WSAD pour le développement d'applications côté serveur, qu'elles soient J2EE, XML ou services Web.

Le 5 novembre, IBM a annoncé qu'il donnait sous licence Open Source une plate-forme de développement d'applications nommée Eclipse. Ce cadeau d'IBM à la communauté des développeurs est estimé à 40 millions de dollars.

Cette annonce sonne le glas de VisualAge for Java au profit d'une toute nouvelle famille d'outils basés sur Eclipse, qui reprend le nom WebSphere Studio. Le premier outil de cette génération est WSAD (WebSphere Studio Application Developer).

Les atouts de l'Open Source

Dans cet article, nous reprendrons pour le terme "Open Source" la définition de l'OSI (Open Source Initiative). On utilisera également le terme logiciel libre comme synonyme, bien que les deux termes ne soient pas forcément toujours équivalents.

Un raccourci rapide permet de dire qu'un logiciel est Open Source si le code source de ce logiciel est disponible, si ce code source

[suite page 6]

Sommaire

DÉVELOPPEUR RÉFÉRENCE

LETTRE BIMENSUELLE ÉDITÉE PAR

IDG COMMUNICATIONS FRANCE

5, rue Chantecoq
92808 Puteaux Cedex
Tél. : 01 41 97 61 61

Adresse électronique :
devreference@idg.fr

Directeur de la publication : Ted Bloom
Commission paritaire : en cours
Dépôt légal : 4e trimestre 2001

Rédaction

Editeur
Michel Crestin

Rédacteur en chef
Pierre Tran tran@idg.fr

Secrétaire de rédaction
Catherine Maupu maupu@idg.fr

Assistantes de rédaction
Isabelle Bounan, Katheline Renaudin

Ont collaboré à ce numéro
Jean-Louis Bénard, Sandrine Chilotti, Gérard Frantz, Didier Girard, Eoin Lane, Guillaume Louel, Méderic Morel

Mise en page

Pierre Tran

Directeur artistique groupe
Patrice Servage

Illustrations/Photographies
Marc Guillaumot

Infographies
Pascal Bidaud, Pierre Tran

Service Abonnements
Développeur Référence
BP 6 – 59718 Lille Cedex 9
Tél. 03 20 12 11 17
Fax 03 20 12 86 09

Tarif :
1 an 20 numéros : 200 euros
Prix de lancement : 150 euros
(jusqu'au 28/02/2002)

Renseignements :
Lucienne Bosser bosser@idg.fr

Les news

Evénements | Tendances | Nouveaux produits

Retrouvez chaque jour les news sur www.devreference.net

Embarcadero : une suite pour les bases de données

Embarcadero Technologies propose une suite comprenant trois modules : Design, Develop et Administer. Ces modules offrent la possibilité de concevoir, développer et administrer des bases de données et permettent aux développeurs de bases de données de construire et maintenir les applications.
Cette suite s'articule autour des solutions DB Artisan, ER/Studio, Describe et Rapid SQL.
Les technologies Embarcadero permettent également de prendre un contrôle intégral d'un environnement serveur et d'atteindre un niveau d'automatisation supplémentaire dans la gestion de cycle de vie des bases de données.
Les outils d'Embarcadero sont utilisés sur les postes clients Windows et supportent les serveurs de bases de données des principaux éditeurs du marché, tels qu'Oracle, Sybase, Microsoft et IBM, qu'ils soient sous Windows NT, Linux ou Unix.
S. C.
http://www.embarcadero.com/

JavaOne en Chine l'année prochaine

Sun envisage de déplacer sa conférence développeurs JavaOne en Chine l'année prochaine, cela afin de favoriser Java dans la communauté des développeurs de ce pays.
Lors du dernier JavaOne, John Gage, directeur des recherches chez Sun, avait mentionné la Chine et l'Inde comme étant susceptibles d'héberger JavaOne en 2002, au lieu du traditionnel San Francisco.
La Chine est un marché immense pour le développement d'applications. Déjà plus de 1 000 universités ont introduit Java dans leurs cursus au cours des trois dernières années. Gage voit également dans cet événement un rapprochement avec le Japon et les autres pays de la région. Le Japon et la Chine sont destinés à travailler ensemble du fait que les jeux de caractères de ces pays nécessitent l'utilisation de caractères à double octet. Il est donc plus facile de traduire un logiciel entre ces deux langues que d'une langue à simple octet, comme c'est le cas pour la plupart des logiciels en langue anglaise.
Avec près de 10 millions de téléphones cellulaires équipés de Java au Japon, un des focus de la JavaOne sera l'utilisation du système en environnement mobile. En Chine, il reste encore beaucoup de chemin à parcourir. Il n'existe que deux modèles de mobiles équipés de Java. L'enseignement en université porte sur le Java de base, pour développer des applications d'entreprise, mais rien de spécifique au mobile ou aux autres types d'applications.
P.T.
http://www.sun.com/

Développement collaboratif avec Borland TeamSource DSP

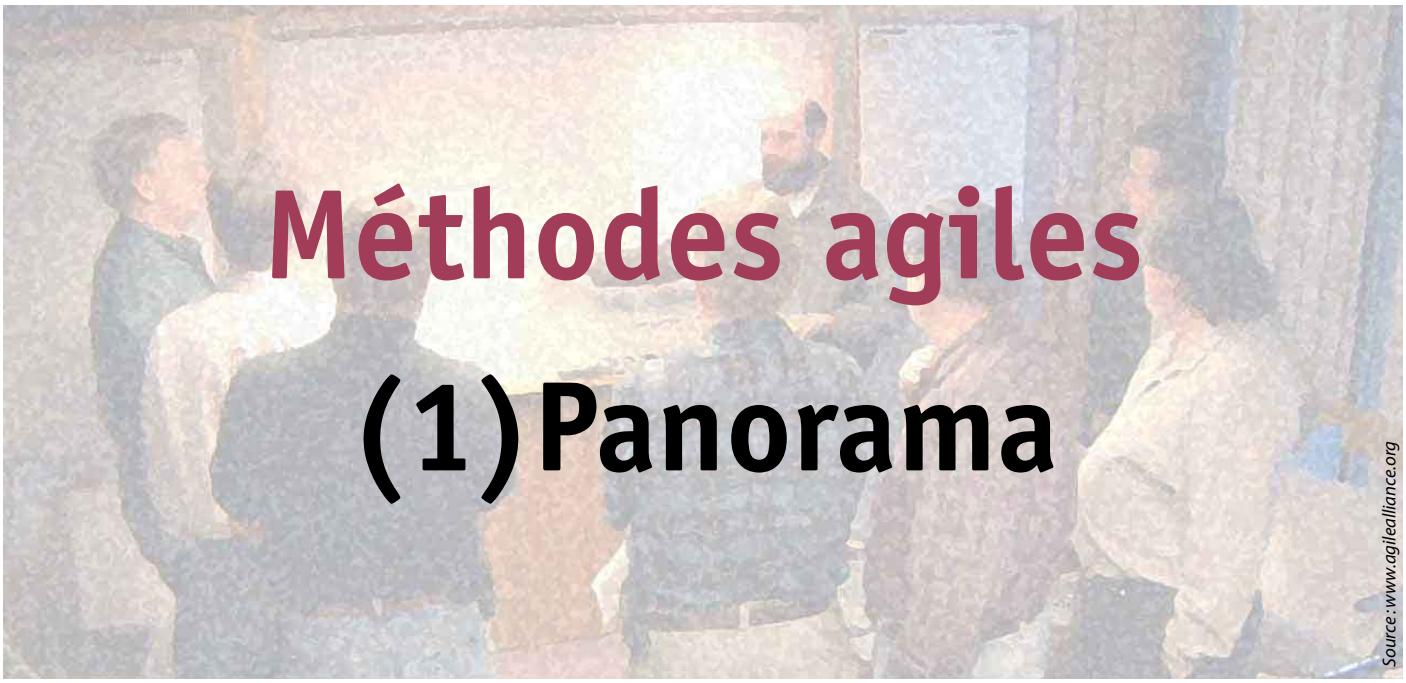
Les services de développement hébergés permettent de relier des équipes de développement occupant des fonctions diverses et travaillant dans des régions ou des entreprises différentes.
TeamSource DSP répond à ce besoin de développement distribué d'applications. Basée sur la plate-forme J2EE, sur les standards relatifs aux services Web et sur le protocole HTTP sécurisé, cette plate-forme de développement offre différents services :
- TeamSource Core Service est l'infrastructure de base. La plate-forme propose des API dédiées à l'intégration de plates-formes, une gestion de la bande passante réseau, et à la sécurité. Une équipe est chargée de la sécurité et du fonctionnement 24/24 et 7/7.
- TeamSource Code Service permet de gérer les configurations logicielles (stockage, mise à jour des versions et partage des fichiers) sur une plate-forme compatible Internet. A l'aide des standards Open Source, les utilisateurs gardent le contrôle de leurs données.
- TeamSource Collab Service fournit un environnement structuré, sécurisé, permettant aux équipes distribuées de coopérer en temps réel. Il intègre une messagerie instantanée qui archive et consigne dans un journal tous les échanges de documents. TeamSource DSP est disponible en version hébergée ou locale. Pendant une période limitée, Borland proposera un "Starter Pack" TeamSource DSP, incluant un abonnement de trois mois pour 20 utilisateurs et une remise de 40 % sur les tarifs standard de formation.
S.C.
http://www.borland.com/teamsource/

Cajun, le framework de services Web de BEA

BEA montrera en février 2002, lors de son show BEA eWorld, un nouveau framework pour développer des services Web appelé Cajun, qui mettra en œuvre une technologie issue de l'acquisition de Crossgain.
Cajun comprend un runtime et un ensemble d'outils. L'un d'eux permet de construire des services Web tout en autorisant l'écriture de code Java. Le framework est conçu pour construire des services Web en Java s'appuyant sur le serveur d'applications WebLogic. Mais les développeurs pourront utiliser avec ce framework des outils tiers tels que ceux de Borland ou WebGain. Cajun sera en principe disponible en avril.
Déterminé à attirer les développeurs, BEA proposera aussi une nouvelle version de son site portail pour développeurs, leur offrant la possibilité de créer leurs propres portails personnalisés. Ce site devrait s'inspirer du MSDN de Microsoft.
Pour conduire cette nouvelle stratégie de séduction des développeurs, BEA a fait appel à Tod Neilsen et Adam Bosworth. Avant de fonder Crossgain, Neilsen et Bosworth avaient travaillé à la création de la communauté des développeurs Microsoft.
P.T.
http://www.beasys.com/

Séminaire SilverStream "Maîtrisez les services Web"

SilverStream organise un séminaire technique gratuit. Ce séminaire sera surtout axé sur la mise en œuvre de services Web en utilisant l'outil eXtend de SilverStream. eXtend est un environnement complet de services intégrés avec un moteur orienté services Web. Il fait appel à eXtend Composer pour créer des services Web à partir de tout système ou source de données back-end, ou bien de toutes pièces. eXtend fonctionne avec les principaux serveurs d'applications J2EE.
La session sera animée par Steve Benfield, directeur technique de SilverStream, contributeur au Java Developer's Journal et faisant partie du comité de rédaction du Web Services Journal de SYS-CON.
P.T.
http://www.silverstream.com



Méthodes agiles (1) Panorama

Source : www.agilealliance.org

Jean-Louis Bénard
est directeur technique de Business Interactif, qui met en place des solutions e-business (front, middle et back-office) sur des architectures multitiers à base d'objets métier.
jlb@businessinteractif.fr

Méthodes agiles : révolution ou coup de bluff ? Au cours d'une série d'articles, nous allons découvrir en profondeur les méthodes agiles. Extreme Programming, bien sûr, mais aussi d'autres méthodes moins connues qui constituent ensemble le "mouvement agile". Leur positionnement par rapport à des méthodes plus traditionnelles et leurs caractéristiques seront passées au crible, avec la nécessaire mise en perspective : est-ce vraiment nouveau ? Est-ce vraiment utile ?

Ce premier article permet de découvrir ce qui est à l'origine de ce nouveau mouvement, ainsi que la "charte" des méthodes agiles. Dans les articles suivants, nous découvrirons les principales méthodes agiles une à une et nous nous interrogerons sur leur réelle efficacité.

Une démarche plus radicale

Les méthodes de gestion de projet informatique connaissent, au même titre que les technologies mises en œuvre, une remise en cause permanente. La proportion importante d'échecs des projets vient souvent alimenter des réactions plus ou moins constructives aux méthodes mises en œuvre. Les évolutions des architectures technologiques et des outils de développement y contribuent également pour part importante. Ainsi, UML et Unified Process sont venues en réaction aux méthodes dites "merisiennes" (cycle en V, etc.) à la fois poussées par les technologies objet et les insuffisances de méthodes préconisant des cycles très longs.

Les méthodes agiles n'échappent pas à la règle. Mais elles s'inscrivent dans une démarche plus radicale, plus "extreme". De manière générale, leur but est d'augmenter le niveau de satisfaction des clients tout en rendant le travail de développement plus facile.

Les véritables fondements des méthodes agiles résident plus précisément dans deux caractéristiques principales :

Les méthodes agiles sont "adaptatives" plutôt que prédictives

Si les méthodes lourdes tentent d'établir dès le début d'un projet un planning à la fois très précis et très détaillé du développement sur une longue période de temps, cela suppose que les exigences et spécifications de base restent immuables. Or, dans tous les projets, les exigences changent au cours du temps et le contexte évolue également. Par conséquent, si elles fonctionnent sur des projets courts et de petite taille, ces méthodologies sont trop réfractaires au changement pour pourvoir être appliquées à des projets plus importants.

A l'inverse, les méthodes agiles se proposent de réserver un accueil favorable au changement : ce sont des méthodes itératives à planification souple qui leur permettent de s'adapter aux changements de contexte et de spécifications du projet. Nous aurons l'occasion de revenir au cours des différents articles sur ce principe fort qui ne va pas sans poser de problèmes dans une démarche de réalisation forfaitaire.

Les méthodes agiles sont orientées vers les personnes plutôt que vers les processus

Les méthodes agiles s'efforcent de travailler avec les spécificités de chacun plutôt que contre la nature des membres de l'équipe pour que le développement soit une activité plaisante ; chacun doit se voir confier une part de responsabilité.

Nées en réaction aux méthodes traditionnelles, il existe une grande diversité de méthodes auxquelles ont peut donner le qualificatif d'agiles. Avant de passer en revue les spécificités de chacune, essayons tout d'abord de dégager les principales caractéristiques communes à toutes ces méthodes. En d'autres termes, qu'est ce qui fait qu'une méthode de développement est dite agile ?

Dans les prochains numéros de DÉVELOPPEUR RÉFÉRENCE, nous étudierons les méthodologies suivantes :

- Extreme Programming
- Dynamic Software Development Method
- Adaptive Software Development
- Crystal
- Scrum
- Feature Driven Development

Quatre valeurs, douze principes

En février 2001, les instigateurs des principales méthodes agiles se sont réunis pour former l'Agile Alliance. Ensemble, ils ont pu dégager des principes fondamentaux sur lesquels s'appuyer pour que le développement puisse se faire rapidement et s'adapter au changement. Leur travail a abouti au "Manifesto for Agile Software Development" (Manifeste pour le Développement Agile d'Applications, voir figure 1). Examinons en détail les messages portés par ces quatre valeurs...

Figure 1. Le Manifeste pour le développement agile d'applications (http://www.agilealliance.org).

1. Priorité aux personnes et aux interactions sur les procédures et les outils

La meilleure garantie de succès réside selon l'Agile Alliance dans les personnes : même une bonne méthode, de bonnes procédures ne sauveront pas un projet de l'échec si l'équipe n'est pas constituée de personnes adéquates et ouvertes. Par contre, une mauvaise méthode peut conduire une équipe parfaite à l'échec. Pour constituer une équipe, on privilégiera des développeurs moyens capables de travailler en groupe et de communiquer à des génies individualistes.

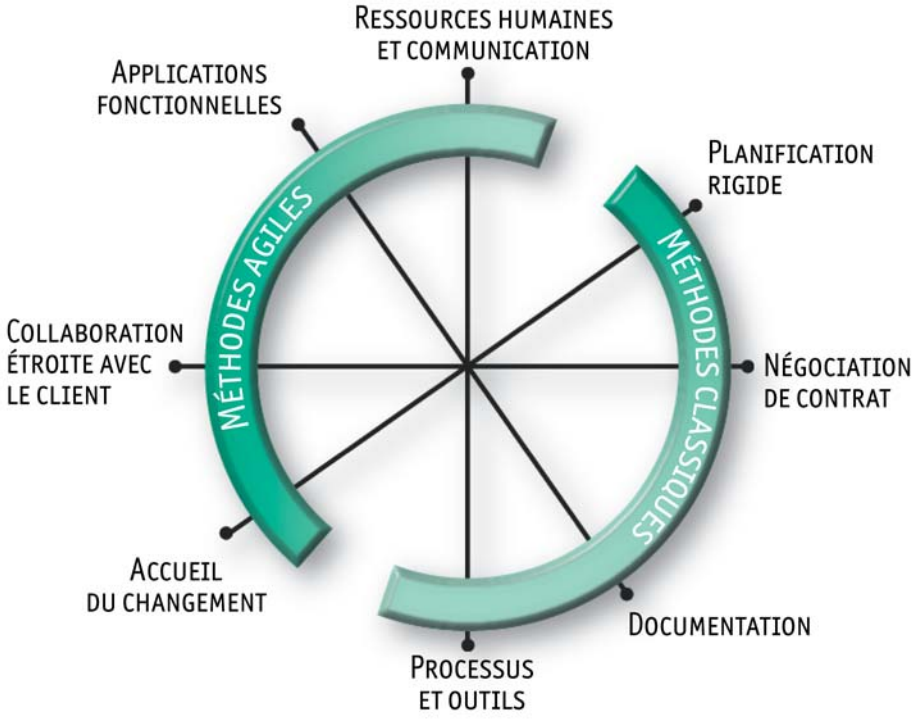
La gestion de projet a longtemps établi la "méthode" comme le vecteur essentiel de succès d'un projet. Cette méthode a souvent pris la forme de procédures. L'importance accordée aux outils supports de la gestion de projets s'est également renforcée au détriment de l'individu. Utiliser les outils appropriés (compilateurs, environnements de développement, outils de modélisation, etc.) peut être décisif. En revanche, l'utilisation d'outils surdimensionnés ou faisant de l'homme un simple instrument de la démarche est aussi néfaste que le manque d'outils appropriés.

Opposés à une approche trop outillée, les membres de l'Agile Alliance affirment que la meilleure garantie de succès réside dans les personnes.

2. Priorité aux applications fonctionnelles sur une documentation pléthorique

Les partisans des méthodes agiles affirment la légitimité d'une certaine forme de documentation mais ils dénoncent les effets pervers d'une documentation pléthorique. En effet, l'écriture et le maintien à niveau de la documentation sont extrêmement consommateurs de ressources. Un document qui n'est pas mis à jour régulièrement devient très rapidement inutile, voire même trompeur. Le manifeste est donc favorable aux documentations succinctes, ne décrivant que les grandes lignes de l'architecture du système mais régulièrement tenues à jour, et d'une documentation permanente du code lui-même.

Figure 2 - Positionnement des méthodes par rapport aux quatre critères agiles (source : Business Interactif).



Le meilleur transfert des connaissances sur le système s'effectue de toute manière par la participation au travail de l'équipe.

3. Priorité de la collaboration avec le client sur la négociation de contrat

Le succès d'un projet requiert un feedback régulier et fréquent de la part du client. Un contrat qui spécifie les exigences, le planning et le coût d'un projet a priori relève d'une vision utopique d'une projet informatique. Selon les partisans des méthodes agiles, la meilleure manière de procéder pour le client est de travailler en étroite collaboration avec l'équipe de développement, pour lui fournir un feedback continu qui assure un meilleur contrôle du projet. Ainsi, des modifications de spécifications peuvent intervenir très tard dans le cycle de développement du projet. C'est en définitive une solution répondant réellement aux attentes du client qui est réalisée et non une solution répondant aux exigences d'un contrat établi a priori. Nous le verrons en synthèse, ce point ne reste pas simple à mettre en œuvre. Cela nécessite bien sûr une grande maturité du client et du prestataire de service afin d'établir une réelle relation de confiance, ainsi qu'une bonne compréhension de la réalité opérationnelle du projet par les juristes en charge du dossier.

4. Priorité de l'acceptation du changement sur la planification

C'est la capacité à accepter le changement qui fait bien souvent la réussite ou l'échec d'un projet. Lors de la planification, il est donc nécessaire de veiller à ce que le planning soit flexible et adaptable aux changements qui peuvent intervenir dans le contexte, les technologies et les spécifications. Ce sont souvent les premières versions opérationnelles des applications qui vont déclencher chez le client les premières demandes de changement.

Les quatre valeurs ci-dessus sont en fait déclinées sur douze principes plus généraux qui caractérisent les méthodes agiles :

1. Notre priorité est de satisfaire le client en lui livrant très tôt et régulièrement des versions fonctionnelles de l'application source de valeur

Les méthodes agiles recommandent de livrer très tôt, dans les premières semaines si possible une version rudimentaire de l'application puis de livrer souvent des versions auxquelles les fonctionnalités s'ajoutent progressivement. De cette manière, le client peut décider à tout moment la mise en production de l'application, dès

qu'il la considère comme assez fonctionnelle. A chaque version (release), un feedback de la part du client est nécessaire pour permettre soit de continuer le développement comme prévu, soit d'opérer des changements. De cette manière, les changements dans les spécifications interviennent tôt dans le processus de développement et sont moins problématiques.

2. Accueillir le changement à bras ouverts, même tard dans le processus de développement. Les méthodologies agiles exploitent les changements pour apporter au client un avantage concurrentiel

Ceci est un état d'esprit nécessaire : tout changement des exigences doit être perçu comme une bonne chose car cela signifie que l'équipe a compris et appris comment satisfaire encore mieux la demande. Le but des méthodes agiles est de produire des systèmes très flexibles, de sorte que l'impact sur le système d'une évolution des spécification reste minimal.

3. Livrer le plus souvent possible des versions opérationnelles de l'application, avec une fréquence comprise entre deux semaines et deux mois

Ne pas se contenter de livrer des liasses de documents décrivant l'application : il faut constamment garder pour objectif de livrer le plus rapidement possible au client une solution qui satisfasse ses besoins.

4. Clients et développeurs doivent coopérer quotidiennement tout au long du projet

Pour qu'un projet puisse être considéré comme agile, il faut qu'il y ait une interaction permanente entre le client, les développeurs : c'est ce qui guide continuellement le projet.

5. Construire des projets autour d'individus motivés, leur donner l'environnement et le support dont ils ont besoin et leur faire confiance pour remplir leur mission

Dans un projet agile, les personnes sont considérées comme le facteur clé de succès. Tous les autres facteurs, processus, environnement, management sont susceptibles d'être changés s'ils s'avèrent être une entrave au bon fonctionnement de l'équipe.

6. La méthode la plus efficace de communiquer des informations à une équipe et à l'intérieur de celle-ci reste la conversation en face à face

Le mode de communication par défaut au sein d'une équipe agile est la conversation et non l'écrit. Des documents peuvent bien sûr être rédigés mais ils n'ont en aucun cas pour but de consigner par écrit la totalité des informations relatives au projet. Même les spécifications peuvent très bien ne pas être écrites de manière formelle.

7. Le fonctionnement de l'application est le premier indicateur d'avancement du projet

Contrairement à d'autres méthodes, l'avancement du projet ne se mesure pas à la quantité de documentation rédigée ni en terme de phase dans laquelle il se trouve mais bien en terme de pourcentage de fonctionnalités effectivement mises en place.

8. Les méthodes agiles recommandent que le projet avance à un rythme soutenable : développeurs et utilisateurs devraient pouvoir maintenir un rythme constant indéfiniment

Il ne s'agit pas de sprinter sur 100 mètres, mais plutôt de tenir la distance sur un marathon : l'équipe se doit donc d'adapter son rythme pour préserver la qualité de son travail sur toute la durée du projet.

9. Porter une attention continue à l'excellence technique et à la conception améliore l'agilité

La meilleure façon de développer rapidement est de maintenir le code source de l'application aussi propre et robuste que possible. Les membres de l'équipe doivent donc s'efforcer de produire le code le plus clair et le plus propre possible. Ils sont invités à nettoyer chaque jour le code qu'ils ont écrit.

10. La simplicité – art de maximiser la quantité de travail à ne pas faire – est essentielle

Rien ne sert d'essayer d'anticiper les besoins de demain. Au contraire, il faut construire le système le plus simple répondant aux besoins actuels pour que celui-ci soit facilement adaptable dans le futur.

11. Les meilleures architectures, spécifications et conceptions sont le fruit d'équipes qui s'auto organisent

Les responsabilités ne sont pas confiées à quelqu'un en particulier mais à l'équipe dans son intégralité. Les membres de l'équipe prennent ensuite leurs responsabilités et se partagent les tâches sur le principe du volontariat.

12. A intervalles de temps réguliers, l'ensemble de l'équipe s'interroge sur la manière de devenir encore plus efficace, puis ajuste son comportement en conséquence

Une équipe agile est consciente que son environnement est en perpétuelle évolution. C'est pourquoi elle ajuste continuellement son organisation, ses règles et son fonctionnement de manière à rester agile.

Les principales écoles

En s'appuyant sur ces principes, plusieurs écoles ont donné jour à des méthodes plus ou moins proches. Les principales, que nous étudierons dans les articles suivants, sont : Extreme Programming, Dynamic Software Development Method, Adaptive Software Development, Crystal, Scrum, Feature Driven Development.

Ces différentes écoles se positionnent plus ou moins radicalement par rapport aux méthodes traditionnelles. La plus connue reste probablement Extreme Programming (XP) : un nom porteur et une position plutôt radicale ont rapidement créé sa notoriété.

Pourtant, nous avons souhaité ajouté au panorama deux méthodes, souvent qualifiées de traditionnelles : RAD (Rapid Application Development) et UP (Unified Process). Nous verrons que même si ces méthodes ne respectent pas complètement les douze principes définis par l'Agile Alliance, certaines de leur caractéristiques s'inscrivent bel et bien dans une démarche agile.

L'agilité : le bon sens en action ?

Sans même entrer dans le détail de chacune des méthodes, la lecture des principes agiles fait souvent réagir : "Je ne vois rien de neuf", "C'est du bon sens, tout simplement", "Je mets déjà en œuvre une grande partie de ces principes".

Effectivement, la plupart de ces principes tombent sous le sens. Les fondateurs des méthodes agiles ne revendiquent d'ailleurs pas la propriété exclusive de ces règles. Néanmoins, force est de constater que si ces principes sont bien connus, ils ne sont pas forcément mis en œuvre. Du moins pas de manière systématique. Les méthodes agiles se proposent d'organiser le projet de telle sorte que ce "bon sens" puisse être mis en œuvre plus facilement.

La principale question que l'on puisse se poser est la suivante : ces principes de bon sens sont-ils tous adaptés à l'ensemble des cas ? Rien n'est moins sûr. La ligne dure des défenseurs d'Extreme Programming (XP) prône pourtant le fait que ne pas mettre en œuvre l'ensemble des principes d'XP revient à ne pas faire d'XP… Ce point de vue est loin d'être partagé par tous. Je reste personnellement persuadé que l'une des forces de l'agilité est la souplesse et l'adaptabilité à des contextes très différents. Enfermer la démarche dans une application systématique de l'ensemble des règles me semble quelque part aller à l'encontre même de l'origine du mouvement agile. L'un des facteurs clé de succès des méthodes agiles sera leur capacité à s'intégrer dans des méthodes traditionnelles pour les faire évoluer. Un point sur lequel nous aurons l'occasion de revenir dans les articles suivants…

Retrouvez et commentez cet article sur www.devreference.net/article.php3?id=55

Open Source

Eclipse/WSAD

Retour d'expérience

Didier Girard
est à la tête de la Direction Technique Technologies & Solutions d'Improve. Il est Docteur en Informatique de l'Ecole Normale Supérieure de Lyon, expert J2EE et XML créateur du site portail www.application-servers.com. Il est l'auteur du livre blanc « XML pour l'entreprise ».

« Je remercie sincèrement Bruno Paul pour l'aide qu'il m'a apporté lors de la rédaction de cet article ».

[Suite de la page 1]

est modifiable et ce code source peut être redistribué

Le fait qu'un logiciel soit Open Source ne signifie pas pour autant qu'il soit risqué de l'utiliser. La preuve : les plus grands éditeurs tels que Sun, IBM, Microsoft ou HP s'appuient sur des logiciels Open Source pour développer leur gamme de produits. Ils ont choisi de s'appuyer sur des logiciels libres car ils sont de grande qualité. En effet, les meneurs des projets Open Source phares sont toujours des personnes très compétentes qui s'investissent dans le logiciel libre par défi technique et pour la reconnaissance de leurs pairs.

Le marketing autour des projets Open Source laisse entendre que les logiciels Open Source sont plus stables, que le respect des standards et la qualité du support sont meilleurs. Ces avantages ont cependant un prix, la documentation associée aux projets Open Source est souvent très succincte et il est parfois difficile pour un débutant d'installer ces logiciels et de les utiliser. C'est pourquoi le marché de l'Open Source n'existe qu'associé à un marché de services. Bon nombre de sociétés ont compris cet enjeu et proposent des services permettant de faciliter l'utilisation des logiciels libres.

Le code source des logiciels Open Source étant disponible, un développeur peut comprendre le fonctionnement des logiciels, corriger des bugs, ou encore procéder à des évolutions. Si ces évolutions ne sont pas acceptées par le mainteneur du projet, il est possible de créer un projet concurrent (fork). Chaque développeur de la communauté décide alors de participer à l'un ou l'autre, ou aux deux projets à la fois. L'histoire de l'Open Source montre que les forks sont très rares.

La taille de la communauté permet une sélection darwinienne des projets : les plus valides techniquement survivent, les autres sont mis en sommeil, disparaissent ou fusionnent. Même dans des niches écologiques étroites, où il peut n'y avoir aucune concurrence, un projet ne peut persister très longtemps sans soutien de développeurs ou d'utilisateurs qui motivent le mainteneur à rajouter de nouvelles fonctionnalités.

IBM et l'Open Source

Le soutien du mode de développement Open Source par IBM ne date pas du 5 novembre dernier. Avant Eclipse, IBM maintenait un grand nombre de projets Open Source et participait très activement à des projets majeurs : Apache Jakarta, Mozilla, Apache Web Server, Apache XML ou encore Linux.

Les composantes que l'on retrouve le plus souvent dans les projets Open Source soutenus par IBM sont XML, Java, Linux et les services Web. Le point commun entre ces technologies est la portabilité :

- Java est un langage portable, ce qui permet de faire disparaître le coût du portage d'une application d'un système à un autre.
- Linux est un système "portable" qui permet de limiter les coûts de portage d'une application native d'une plate-forme à une autre.
- XML assure la portabilité des données.
- Les services Web assurent un accès portable aux applications.

Qu'IBM soutienne des projets qui vont lui permettre de maintenir la qualité des services offerts par ses plates-formes matérielles (iSeries, pSeries, xSeries et zSeries) tout en diminuant le coût de développement de ces services n'a donc rien d'étonnant.

L'Open Source ne représente pas une menace pour IBM : outre les indéniables qualités de l'Open Source que nous avons déjà énoncées, l'avantage pour IBM est qu'un logiciel libre n'appartient à aucune entreprise. C'est pourquoi le succès de l'Open Source ne devrait pas dresser un futur concurrent en face d'IBM.

Pourquoi Eclipse ?

Avoir une stratégie sur la portabilité des applications, c'est bien, avoir une communauté de développeurs pour développer ces applications portables, c'est mieux. Il était donc important, voire urgent, pour IBM de proposer à la communauté des développeurs un outil de développement. C'est l'objectif d'Eclipse.

Avec Eclipse, IBM propose une nouvelle étape dans la standardisation.

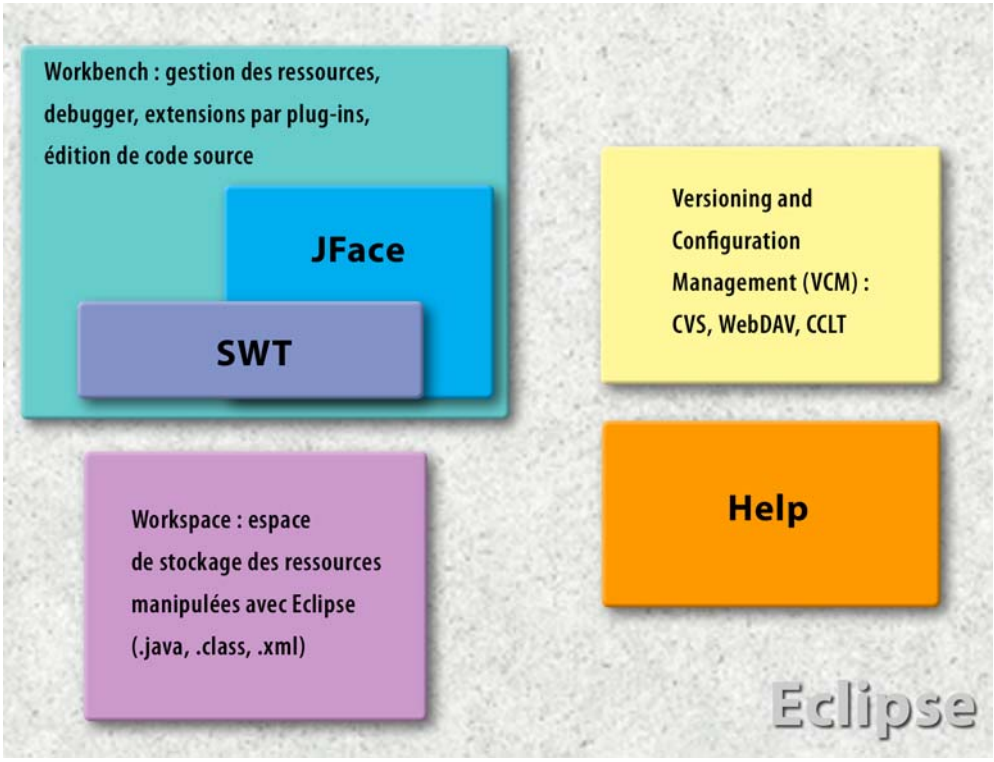


Figure 1. Architecture de la plate-forme Eclipse.

Pour contrer Microsoft et son outil VisualStudio .NET, il ouvre les spécifications d'une plate-forme de développement d'outils qui a nécessité deux ans d'effort. Plutôt que d'être le seul acteur à utiliser cette ambitieuse plate-forme de développement, IBM crée un nouveau marché. Grâce à la JVM, socle d'Eclipse, ce marché sera multi plates-formes matérielles (PC, serveurs, mainframes, assistants personnels…). Parions qu'IBM ne sera pas long à retirer des bénéfices de ce marché, puisqu'il se positionne déjà avec son produit WebSphere Studio Application Developer, construit sur Eclipse. La dynamique autour d'Eclipse est semblable à celle qui consiste à baisser le taux d'imposition, pour faire repartir la consommation et prélever au final plus d'impôts.

La création du projet Open Source Eclipse montre qu'IBM a parfaitement assimilé le nouveau principe de base de l'édition d'outils : celui qui gagne des parts de marché est celui qui a la plus grosse communauté de développeurs. Ce principe est apparu et s'est imposé avec le Web, médium de communication. Il existe plusieurs façons de créer une communauté : le marketing, la diffusion des outils (par exemple par le modèle Open Source), le leadership technologique ou une meilleure productivité apportée par l'ergonomie des outils.

Eclipse n'est plus IBM

IBM avait annoncé depuis juin dernier son souhait de donner Eclipse à la communauté Open Source. On l'attendait sur la licence et le mode de fonctionnement du projet. Le lancement officiel du projet Eclipse fin novembre a rassuré : la licence est la "Common Public License", reconnue par l'OSI.

Le projet est géré par un consortium, constitué aujourd'hui de dix entreprises avec des droits de vote équivalents : Borland, IBM, Merant, QNX, Rational Software, RedHat, SuSE, TogetherSoft et WebGain. L'accès de nouveaux participants à ce comité obéit à des règles habituelles dans les projets Open Source : le projet Eclipse est une "méritocratie".

L'architecture d'Eclipse

Eclipse a été conçu par Object Technology International (OTI, filiale d'IBM), qui a une très forte expérience du développement de produits : depuis 1988, cette équipe a produit VisualAge Smalltalk, VisualAge for Java et VisualAge Micro Edition. L'un des trois mainteneurs du projet Eclipse est Erich Gamma, co-auteur du célèbrissime "Design Patterns", bible de tous les développeurs soucieux d'architecture. Le projet Eclipse est déjà très avancé, puisque la version 1.0 est disponible et que la version 2.0 en développement est prévue pour avril 2002.

Le projet Eclipse est constitué de trois sous projets : la plate-forme, le Java Development Tools (JDT) et le Plug-in Development Environment (PDE).

La plate-forme

Elle définie un ensemble de frameworks et de services communs pour assurer une interopérabilité des outils qui se rajoutent par dessus (plug-ins). C'est l'équivalent d'un kernel pour un OS. Elle comprend un système de gestion concurrente de ressources distribuées VCM (Versioning and Configuration Management), une gestion de modèle de projets, une gestion automatique des deltas entre versions pour les compilations incrémentales, une infrastructure de debug indépendante des langages de développement, un framework d'extension des fonctionnalités par plug-ins, une bibliothèque graphique portable SWT (Standard Widget Toolkit), un framework pour une interface graphique portable (JFace), un système générique d'aide, de recherche, de comparaison, de mise à jour des plug-ins, des parseurs, l'intégration de Jakarta Ant, le support de scripts, etc.

Comme tout le reste d'Eclipse, la plate-forme est intégralement codée en Java, à l'exception d'une unique DLL de 232Ko en C. La JVM utilisée est interchangeable à souhait, Eclipse supporte la JVM 1.3 aux dernières bêtas de la JVM 1.4. Des JVM antérieures peuvent être utilisées mais un certain nombre de fonctionnalités seront inopérantes.

La DLL mentionnée est une partie du Standard Widget Toolkit. Cette nouvelle bibliothèque graphique a surpris toute la communauté Java. SWT/JFace se positionne comme un concurrent de Swing. Toute l'interface graphique d'Eclipse est bâtie sur SWT. Swing n'est pas pour autant passé à la trappe, on peut toujours développer et tester des composants AWT ou Swing avec Eclipse, mais il n'existe pas pour l'instant d'environnement de programmation visuelle.

IBM est le concepteur de SWT, dont la première version de production date de mars 2000. IBM utilise déjà cette bibliothèque dans des produits commerciaux, en témoigne l'interface utilisateur de VisualAge for Java Micro Edition 1.1. De manière générale, IBM va tendre à utiliser la plate-forme Eclipse pour migrer le maximum de ses produits logiciels. Pour autant, IBM a affirmé ne pas vouloir imposer SWT comme standard. SWT est à la disposition de la communauté Java, c'est elle qui devra en assurer la promotion si elle l'estime nécessaire.

Quels sont donc les avantages de SWT par rapport à Swing ?

- SWT est moins complexe : environ 200 classes pour SWT contre plus de 1000 pour Swing.
- SWT est plus flexible : il permet des appels directs à l'API graphique de l'OS. On peut donc coder des appels à des ActiveX sur Win32 (mais au détriment de la portabilité).
- SWT est aussi rapide qu'une bibliothèque native, notamment pour le rafraichissement des composants.
- Les applications SWT sont mieux intégrées : elles ressemblent vraiment

à des applications natives, les primitives graphiques utilisées sont directement celles de l'OS.
- SWT est plus facilement portable : 232 Ko de code natif pour SWT dans le JDK 1.3, contre 960 Ko pour Swing.

Il existe aujourd'hui des implémentations complètes de SWT sur Win32 et Motif (Linux, AIX, Solaris). Un développement sur GTK/Linux, Motif/HP-UX et Photon/QNX est en cours. L'utilisation de GTK permettrait l'appel à des composants Bonobo, là aussi au détriment de la portabilité. Pour assurer une reprise de l'existant, des ponts AWT-SWT et Swing-SWT sont en développement, mais le problème est ardu.

En cassant le paradigme de la portabilité totale des Swing, SWT laisse une plus grande liberté au programmeur. En choisissant ou pas d'intégrer des composants spécifiques de l'OS où il se trouve, il peut choisir entre portabilité et rapidité du développement. Et surtout, SWT résout de manière élégante les problèmes de performance des grosses applications Swing. Par contre, l'application doit libérer explicitement les ressources de l'OS allouées par SWT. Elle ne peut pas se reposer sur le ramasse-miettes pour cela.

Si SWT reçoit un fort soutien de la communauté, le renouveau des développements Java côté client est assuré.

Les plug-ins : ne réinventez pas la roue !

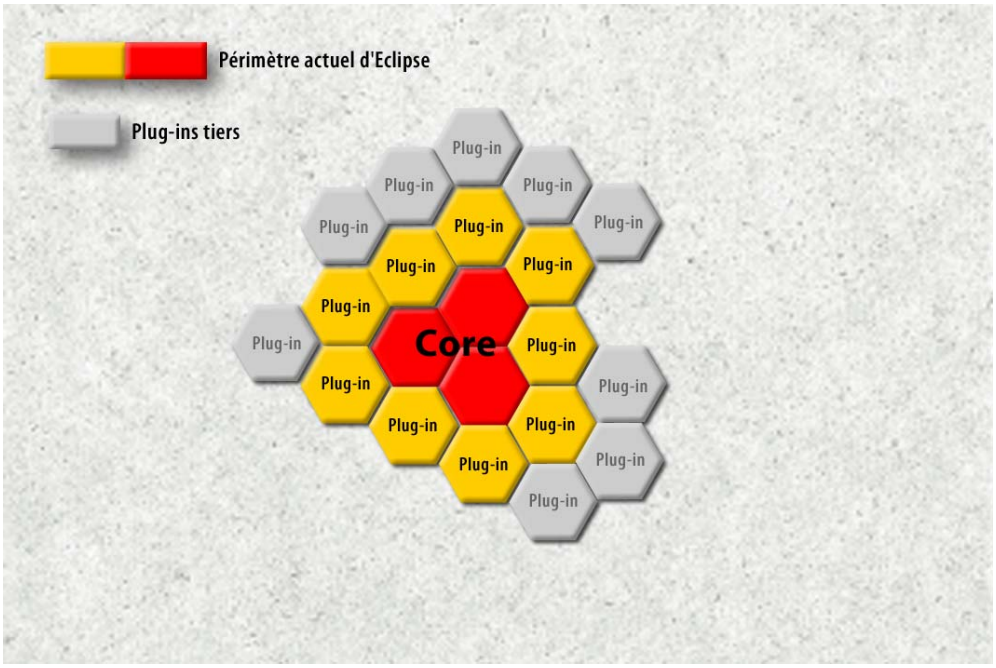
Un autre intérêt majeur de la plateforme réside dans son architecture de plug-ins. Pour mieux comprendre sa maturité, il faut détailler les différents niveaux d'intégration offerts pour un plug-in :

1. Invocation : un double-clic appelle la cible.
2. Intégration des données : permet à des plug-ins d'accéder à des données qui viennent d'être créées par d'autres plug-ins.
3. API : l'ajout d'un plug-in sur le système rajoute des nouveaux comportements à des plug-ins déjà installés. On n'accède plus à des données comme précédemment, on réutilise un comportement.
4. Framework d'interface graphique : permet de construire une interface utilisateur.

La découverte et la mise à jour des nouveaux plug-ins est dynamique. Mieux, les plug-ins peuvent eux-mêmes comporter des points d'extensions, et supporter leurs propres plug-ins, lesquels peuvent utiliser tous les services de la plate-forme.

La prolifération de ces plug-ins n'alourdit pas inutilement l'application. En effet, bien que leurs présences et leurs fonctionnalités soient reconnues dès l'installation, ils ne sont pas chargés en mémoire avant leur utilisation réelle. Il existe aussi un mécanisme de déchargement des plug-ins de la mémoire, quand l'utilisateur ferme un outil par exemple. L'empreinte mémoire de l'application est ainsi fortement réduite. Je recommande d'utiliser Eclipse sur une station avec 180 Mo de RAM (avec Windows NT4). Il est possible d'alléger très simplement l'application en retirant les plugins inutilisés et qui ne font pas partie du Core (composé uniquement de org.eclipse.core.boot, org.eclipse.core.runtime, org.apache.xerces). La plateforme est ainsi customisable et facilement déployable dans des contextes très divers.

Figure 2. Architecture des plug-ins.



Le Java Development Tools

Le JDT (Java Development Tools) est un environnement de développement intégré pour Java, très complet, qui utilise les services de la plate-forme : éditeur de code, complétion contextuelle, debugger, JDK adaptable pour chaque projet, refactoring, gestion des versions, comparaison, recherche, fusion de versions, compilation incrémentale, build automatique ou manuel, assistants de création.

C'est la partie la plus visible du projet Eclipse, car c'est l'environnement qui s'ouvre quand on lance Eclipse. Le JDT distingue ce projet Open Source des autres : il propose un IDE déjà parfaitement utilisable pour du développement Java professionnel, avec une documentation complète. Le JDT est utilisé par OTI depuis plusieurs mois pour développer les nouvelles fonctionnalités d'Eclipse (statut dog food). Le mainteneur du projet JDT est Erich Gamma. Il participe aux débats sur le newsgroup et sur les listes de diffusion du projet Eclipse. Il se voit assigner dans Bugzilla la résolution d'un nombre important de problèmes.

Le JDT reprend une vue par fichier d'un projet Java. Cependant on a accès à l'arborescence des méthodes dans le panel navigateur, comme dans VisualAge for Java. Par défaut , on édite l'ensemble du source du fichier, bien qu'une option permette de n'éditer que la méthode sélectionnée. Tous les développeurs Java trouveront leur bonheur. Nouveauté : le javadoc peut être consulté en survolant la déclaration avec la souris.

Malgré sa maturité, le projet JDT n'est pas terminé pour autant. Il manque par exemple une génération de javadoc, un support aisé de l'internationalisation d'applications comme dans VisualAge for Java, des raccourcis clavier personnalisables, et un environnement de composition visuelle pour SWT, AWT ou Swing. L'interface utilisateur est seulement disponible aujourd'hui en anglais, mais l'architecture supportant l'internationalisation de l'interface est en place. Très bientôt l'allemand et le japonais seront supportés, suivis par l'hébreu et l'arabe.

Le Plug-in Development Environment

Le PDE (Plug-in Development Environment) propose un ensemble d'outils pour faciliter le développement de plug-ins pour Eclipse. On pourra créer facilement le fichier manifest XML du plug-in, indiquer les ressources nécessaires à l'exécution, définir les points d'extension, associer des fichiers XML Schema aux plug-ins pour permettre une validation des ressources spécifiées (on pense notamment à la notion de version, ce qui règle de façon élégante le DLL HELL classique du développement dans un environnement Windows).

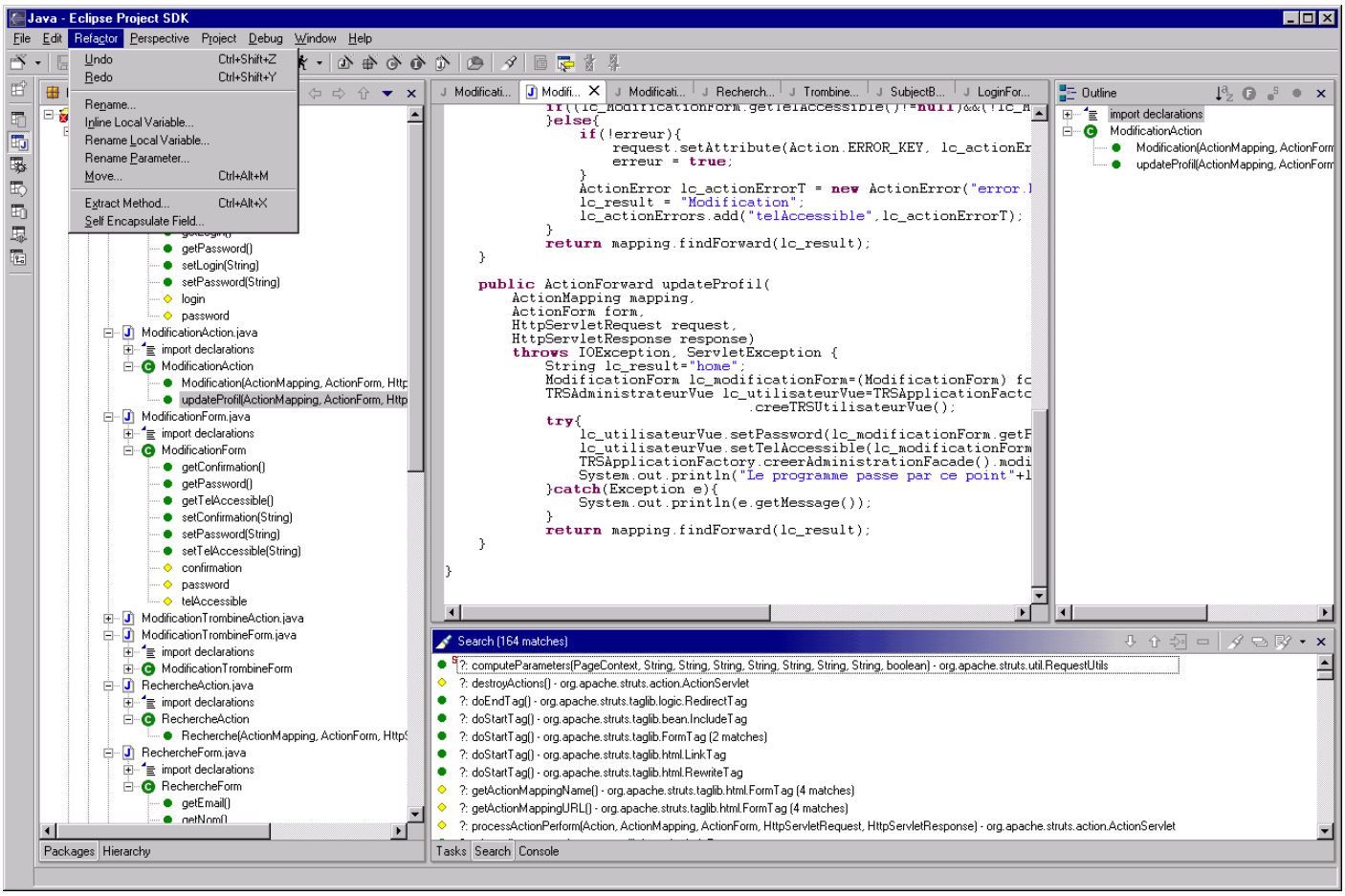


Figure 3. La perspective Java avec l'organisation par défaut.

Prise en main : les perspectives

L'ergonomie générale d'Eclipse est organisée en perspectives. Il s'agit de fenêtres rassemblant des options autour d'un thème commun. La navigation dans Eclipse est complètement customisable : emplacement, présence, taille, empilement des panels, options présentes. Cette versatilité peut dérouter au début, mais permet à tous les développeurs, quelles que soient leurs habitudes, de construire une ergonomie qui leur convienne. Il existe sept perspectives par défaut, nous allons aborder les principales.

Perspective Java

Le JDT est bien sûr constitué d'un ensemble de plug-ins sur la plate-forme. Il ne faut pas restreindre Eclipse à un IDE Java, l'ajout de futurs plug-ins permettra de développer dans d'autres langages. Il existe déjà un plug-in Open Source pour le développement C/C++.

Sans surprise, la perspective Java rassemble tous les outils nécessaires pour développer des classes. Par contre, les habitués de VisualAge for Java seront surpris par la nécessité d'enregistrer systématiquement les changements dans tous les fichiers ouverts avant de changer de perspective. En effet, un fichier ouvert dans une perspective n'est qu'une vue, et ouvrir le même fichier dans une autre perspective ouvre une autre vue, qui ne sont pas forcément toujours synchronisées !

Refactoring et compilation incrémentale

Le JDT intègre des outils de refactoring, prisés par la méthodologie Extreme Programming. Il s'agit de transformer la structure du code sans modifier sa logique. On peut par exemple :

- renommer un objet : toutes les références à cet objet sont automatiquement modifiées ;
- extraire une méthode : la sélection de portion de code peut être transformée en méthode appelée avec un clic ;
- déplacer du code.

Toutes ces actions supportent le retour à l'état initial.

Les fonctionnalités que nous venons de citer sont encore loin de celles offertes par un outil comme IDEA, mais gageons que la communauté Eclipse va se charger de les enrichir régulièrement.

Il est aussi important de signaler que la compilation incrémentale est déjà intégrée dans Eclipse. Cette fonctionnalité permet de recompiler uniquement le code modifié ainsi que ses dépendances.

Perspective debugger

Elle propose des services de debug très classiques. C'est l'un des points en retrait par rapport à VisualAge for Java :

- pas de point d'arrêt conditionnel ;
- pas de remplacement de code à chaud (HCR) ;
- pas de passage méthode par méthode quand elles sont chaînées au sein d'une ligne ;
- pas de reprise du debuggage en amont dans le code (drop to frame).

Ces options très avancées étaient rendues possibles dans VisualAge for Java par une JVM supportant une API propriétaire IBM, et un compilateur qui permettait une instrumentation plus fine du code. Une partie de ces fonctionnalités (dont le magique HCR) ont été standardisées sous l'impulsion d'IBM, et seront présentes dans la JVM1.4. Le HCR commence à apparaitre dans les derniers builds de développement d'Eclipse 2. WSAD 4.0 étant basé sur une branche Eclipse1.0, le HCR est absent.

Le débuggage sur serveur distant (remote debugging) est possible avec l'IBM Distributed Debugger, téléchargeable gratuitement (sous une licence propriétaire).

Travail en équipe

Dans ce domaine critique, IBM a résolument joué la carte des standards.

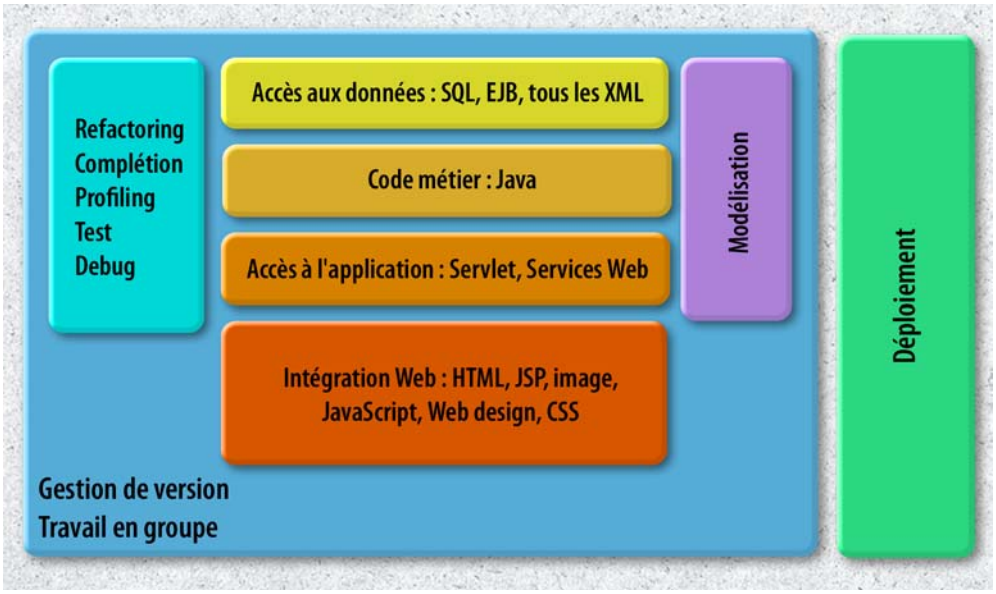


Figure 4. Besoins sur un projet J2EE : WSAD permet d'aborder cette complexité avec un seul outil. La modélisation ne fait pas partie des outils livrés avec WSAD. Elle sera par contre bientôt accessible puisque des éditeurs comme Rational et TogetherSoft annoncent l'intégration de leurs outils dans Eclipse.

Le Versionning et Control Management d'Eclipse est livré avec des clients CVS et WebDAV. La connexion sur un serveur CVS/Unix est supportée avec les protocoles pserver ou extssh (plus sécurisé). La connexion vers CVSNT/Windows NT-2000 n'est recommandée qu'avec pserver, car le mode sécurisé n'est qu'expérimental avec CVSNT.

Si l'on travaille seul, l'installation d'un serveur CVS pour gérer les versions n'est pas indispensable : il existe un historique local de taille paramétrable, mais qui n'assure pas de protection contre la perte de code en cas de crash.

Pari gagné pour IBM ?

Comme tout développement logiciel, Eclipse n'est pas exempt de bugs, mais à l'utilisation ils sont peu gênants. Comme tout projet Open Source, une gestion des problèmes connus est consultable.

Outre ses atouts techniques indéniables, le succès d'Eclipse résidera d'abord dans l'attraction d'une communauté de développeurs, et d'éditeurs tiers qui s'engagent sur cette plate-forme de développement. L'avenir nous dira si Eclipse est un succès.

Le JDT est un superbe produit d'appel pour la plate-forme Eclipse. Il est cantonné en standard à du développement J2SE, mais on peut importer Tomcat et bénéficier d'un environnement de développement Servlet/JSP solide entièrement Open Source. Les développeurs désirant un niveau de productivité supplémentaire se tourneront vers WSAD.

WSAD

Durant la phase de réalisation d'un projet J2EE de nombreuses technologies sont manipulées par les développeurs (voir figure 4). IBM positionne WebSphere Studio Application Developer (WSAD) comme l'environnement de développement qui va permettre de gérer toute la complexité technologique d'un projet J2EE.

Un environnement de développement J2EE 1.2

WSAD 4.0 est un environnement de développement intégrant les fonctionnalités de VisualAge for Java et de WebSphere Studio. Il est composé d'un workbench basé sur Eclipse 1.0 et d'un ensemble de plug-ins permettant de développer, tester et déployer des applications J2EE 1.2. Si Eclipse n'appartient pas à IBM, WSAD fait lui bien partie de l'offre IBM, et à ce titre il est payant.

Les plug-ins WSAD sont rassemblés sous forme de perspectives, une perspective étant un point de vue sur le projet. WSAD 4.0 comporte neuf perspectives majeures (voir figure 5).

Les perspectives Java, Debug et Aide proviennent directement d'Eclipse. Les perspectives Web, J2EE, Serveur, Profiling, Data et XML représentent les ajouts de WSAD. Ces ajouts seront détaillés par la suite.

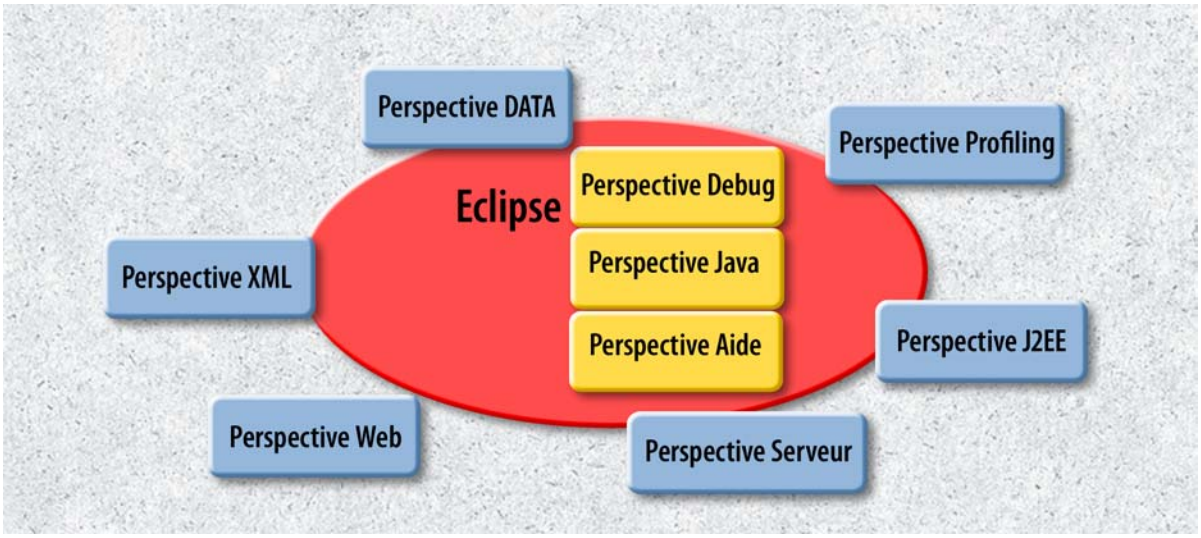
Aujourd'hui, WSAD4.0 n'existe que sur Win32. Une version pour Linux est en développement et sortira au premier semestre 2002. Elle pourrait être basée sur Eclipse 2.0.

Perspective Web

La perspective Web intègre un environnement complet de développement Web. Cet environnement fournit les outils nécessaires au développement d'applications Web, cela inclut : les pages Web statiques, les JSP, les Servlets ainsi que le descripteur XML de déploiement.

Tous ces aspects sont réunis dans un environnement de travail unique permettant à tous les intervenants d'un projet Web (auteurs de contenu, infographistes, développeurs et webmasters) de travailler ensemble avec le même outil.

Figure 5 : WSAD apporte un ensemble de perspectives qui vont permettre de traiter la complexité d'un projet J2EE.



Parmi les fonctionnalités offertes, citons :
- Création, validation, édition de fichiers, JSP et HTML
- Edition de graphismes et d'animations
- Support de l'édition de feuilles de style en cascade (CSS)
- Moteur JavaScript (Rhino)
- Support de DHTML
- Importation HTTP et FTP
- Exportation vers un serveur via FTP
- Import/export de fichiers WAR
- Visualisation des liens
- Gestion et parsing des liens
- Edition graphique du fichier web.xml

WSAD se veut l'outil collaboratif idéal pour la création, l'assemblage, la publication, le déploiement et la maintenance dynamique d'une application Web.

Il semble évident qu'au moins une ligne du cahier des charges de WSAD fixait pour objectif de faire aussi bien, voire mieux, que Dreamweaver et FrontPage pour la partie code client, et aussi bien que FireWorks pour la partie Web design, à la fois sur le fond et sur la forme. Sur la forme, l'objectif est atteint avec brio, l'ergonomie est familière, la prise en main immédiate. Sur le fond, si l'on fait une synthèse des points forts et des points faibles de l'application, on peut en déduire que la boîte à outil Web de WSAD fait le poids face aux leaders du marché. Un intégrateur et un designer peuvent travailler avec cet outil sans trop perdre en productivité, quitte à utiliser de manière anecdotique quelques outils externes.

Une partie du plug-in Web Art Designer n'est pas encore codée en Java, mais sous forme d'exécutable Windows. Elle est donc absente de la version Linux de WSAD 4.0, actuellement en bêta.

Perspective J2EE

La perspective J2EE donne accès aux informations nécessaires pour créer et configurer une application d'entreprise. Depuis cette perspective, il est donc possible de manipuler les composants Web mais aussi de créer des EJB 1.1 et de paramétrer le serveur d'applications. Depuis cette perspective, des éditeurs graphiques permettent de manipuler les fichiers web.xml, ejb-jar.xml ou application.xml, autant de fichiers à utiliser pour le déploiement de l'application.

Pour les EJB, toutes les techniques de développement sont supportées : top-down (génération du schéma de la base de donnée à partir de l'EJB), bottom-up (génération du Bean à partir du schéma) et in-the-middle (la plus flexible). Bien qu'utilisable, c'est le module qui nous a semblé le moins performant.

Perspective XML

Cette perspective donne accès à des outils utiles lors du développement d'applications XML :
- Editeur XML : avec cet outil il est possible de créer et de valider des documents XML. Il comprend une vue de conception et une vue source.
- Editeur de DTD : cet éditeur autorise la création et la validation de DTD. Les DTD peuvent être converties en XML Schema et vice-versa.
- Editeur de XML Schema : cet éditeur permet de créer et de valider des documents XSD (XML Schema Definition).
- Mapping XML vers XML : cet outil propose une interface graphique qui permet de créer des feuilles de style XSLT de mapping XML vers XML.
- Traceur XSLT : cet outil permet d'exécuter pas à pas une transformation XSLT.
- Validateur de documents XML : vérifie le respect de la DTD ou du XSD.
- Mapping RDB vers XML: cet outil convertit automatiquement les résultats de requêtes SQL en un fichier XML avec son XSD associé.

On retrouve dans cette perspective un bon noyau d'outils pour les développements XML. Ils sont encore de qualité inégale : les éditeurs de document XML, DTD et XML Schema sont agréables à utiliser. Par contre les outils tournant autour de XSLT manquent encore de maturité.

Perspective Serveur

La perspective serveur permet de visualiser et de modifier la configuration des serveurs en charge d'exécuter les applications Web et les EJB. Il est possible de configurer un déploiement sur un serveur distant. WSAD 4.0 est livré en standard avec un plug-in WAS4 AEs, et d'autres connecteurs pour Tomcat 3.2 et 4.0.

Perspective Profiling

Le développement d'applications performantes est souvent une priorité pour les développeurs. Les outils de profiling permettent de mesurer les performances d'applications, et de localiser ainsi dès le stade du développement les sources d'éventuels problèmes de performances. IBM a réalisé l'intégration de JInsight, un outil de profiling présent à son catalogue depuis quelques années. Pour de meilleures performances, il est essentiel de bien paramétrer les filtres, pour restreindre le profiling aux seules classes qui sont intéressantes.

Cet outil est moins complet que ceux des leaders du marché, qui ont déjà annoncé le portage de leurs produits sur Eclipse. Il apportera néanmoins de grands services à ceux qui n'en disposent pas.

Perspective Data

Cette perspective est destinée aux administrateurs de données et aux développeurs d'EJB. Elle permet de créer, importer ou modifier le schéma d'une base relationnelle. A partir de ce schéma il est possible d'interroger ou de modifier

Services Web : ne parlez plus de SOAP ni de WSDL !

Avec WSAD, rendre le métier d'une entreprise accessible via les services Web est aussi simple que d'installer WinZip ! Pas de code à développer, juste un assistant qui vous guide dans votre développement.

Grâce au niveau d'intégration atteint par un outil comme WSAD, l'enjeu des services Web n'est désormais plus situé sur le développement, c'est uniquement une problématique de métier et d'architecture :
1. L'entreprise doit exposer un service à valeur ajoutée.
2. Le concepteur doit répondre aux bonnes questions, c'est-à-dire construire une architecture qui va garantir la qualité et la sécurité des services proposés.

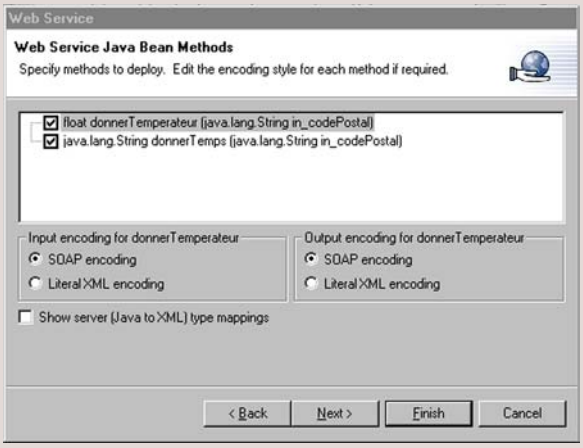


Figure 6. Pour développer un service Web avec WSAD, il n'y a pas de code à produire, il suffit d'utiliser un assistant !

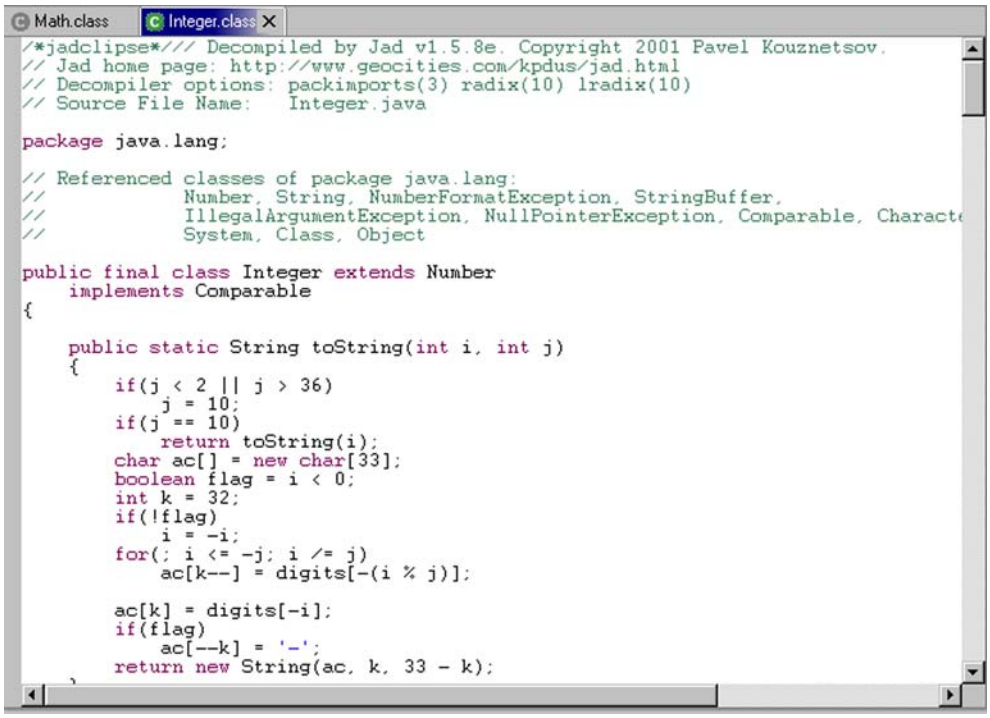


Figure 7. jadclipse est un plug-in qui permet de décompiler à la volée les fichiers .class. Dans cette écran, nous voyons la décompilation de la classe java.lang.Integer par jadclipse.

la base, qui peut bien sûr être distante. Des drivers jdbc pour toutes les grandes bases sont fournis.

Plug-in commerciaux

D'après IBM, 150 sociétés ont rejoint la plate-forme Eclipse. Les offres de ces éditeurs viennent agréablement compléter les outils disponibles. Le principe d'intégration au cœur d'Eclipse prend tout son sens.

Si la plupart des plug-ins tiers ne seront pas prêts avant mi 2002, Rational propose déjà un plug-in ClearCase LT en bundle avec WSAD 4.0 pour Windows. Il ne s'agit que du premier pas de cet éditeur phare, qui s'est engagé à proposer l'intégralité de son offre intégré sur Eclipse : ClearQuest, ClearCase Entreprise et Rational Suite. L'offre ClearCase est concurrencée par le client CVS présent en standard. Cependant ces deux produits existaient déjà indépendamment d'Eclipse, et chacun avait ses utilisateurs. Il en va de même pour les futures offres Merant (PVCS), StarBase (StarTeam), ou Telelogic (CM Synergy), toutes situées sur le créneau de la gestion de configuration.

Sitraka (JProbe), un des leaders sur le marché des outils d'optimisation, est concurrencé par l'outil intégré en standard dans WSAD 4.0. Son intérêt réside surtout dans une offre moins onéreuse que WSAD si l'on n´a besoin que de l'outil d'optimisation. C'est aussi le cas pour Macromedia (Dreamweaver, UltraDev, Flash), en concurrence avec les outils de WSAD.

Rational complètera avec un plug-in Rose le module modélisation absent dans WSAD 4.0. Il se retrouvera en concurrence sur Eclipse avec l'atelier de modélisation de TogetherSoft, particulièrement performant pour le développement Java.

On retrouve comme partenaires de la première heure sur Eclipse les sociétés Instantiations et Versata. La première proposait déjà des outils d'amélioration de la productivité sur VisualAge for Java, et la deuxième offre un environnement de développement RAD.

On peut aussi citer des éditeurs leaders sur leurs marchés, qui apporteront une plus-value exceptionnelle à la plate-forme Eclipse : Interwoven (gestion de la relation client), Mercury Interactive (injecteur), Bowstreet (Bowstreet Business Web Factory), Versant (cache de donnée enJin), WebGain (outil de mapping TopLink). Beaucoup d'autres besoins seront couverts ! Citons par exemple le framework de developpement d'applications d'Altoweb, concurencé par le framework Open Source Jakarta Struts.

Le nombre de sociétés annonçant leur support rend crédible l'ambition d'un nouveau marché de composants logiciels interopérables que permet l'architecture ouverte d'Eclipse.

Plug-in libres

Si l'architecture d'Eclipse va dynamiser le marché des plug-ins commerciaux, elle va aussi stimuler les projets Open Source. Pour preuve, plusieurs projets Open Source sont déjà sur les rails : jadclipse (voir figure 7), Poseidon et struts-console.

D'un modèle de développement pour hackers éclairés, l'Open Source est désormais devenu un enjeu économique majeur. Eclipse possède tous les atouts pour devenir rapidement la plate-forme de développement universelle. L'ignorer, c'est passer à côté d'une révolution en marche.

WSAD 4.0 est une offre qui utilise parfaitement l'intégration rendue possible par Eclipse, et s'impose aujourd'hui comme un environnement complet de développement J2EE 1.2. Bonne aventure technologique ! ●

ALLER PLUS LOIN

L'Open source :
http://www.opensource.org

IBM et l'Open Source :
http://www-124.ibm.com/developerworks/oss

Le projet Eclipse :
http://www.eclipse.org

WebSphere Studio Application Developer :
http://www-4.ibm.com/software/ad/studioappdev/

Plug-in Open Source pour le développement C/C++ :
http://www.alphaworks.ibm.com/tech/eclipse4c

Du XML dans vos applications J2EE

Eoin Lane
a travaillé comme consultant senior chez Valtech depuis le début des années 2000. Il s'est consacré ces trois dernières années à Linux, la conception objet, Java, J2EE, EJB et XML, avec un intérêt tout particulier pour les technologies Open Source.

A travers une étude de cas, nous montrerons commment intégrer une couche de présentation XML dans une architecture à couches J2EE. Notre solution assure une indépendance vis à vis de la cible en produisant dynamiquement des données XML et s'appuie sur des technologies Open Source Cocoon, JBoss et Tomcat.

De nos jours, la mutation rapide des entreprises vers l'e-business crée de nouveaux besoins en matière de flexibilité des architectures. Elles doivent être puissantes, robustes, supporter la montée en charge, et surtout être capables de se plier aux nouveaux impératifs du marché. De plus en plus d'entreprises adoptent une architecture multitiers basée sur la plate-forme J2EE pour leurs applications distribuées. Mais ces systèmes reposent encore sur HTML pour la couche de présentation. Cela les empêchent de bénéficier pleinement de la flexibilité qu'un framework multitiers peut offrir. La publication Web au format XML est essentielle à la résolution de ce problème d'architecture.

Une couche de présentation XML

Cet article propose de remplacer l'actuelle couche de présentation J2EE basée sur HTML par une couche de contenu XML dynamique. Au travers de notre étude de cas, nous allons découvrir un serveur d'application XML Open Source avec une solution de bout en bout complètement opérationnelle. Au cœur de cette solution réside Cocoon, une servlet Apache qui fournit un framework de présentation Java pluggable utilisé pour présenter du contenu XML dans une grande variété de formats, à la demande.

J2EE définit un standard pour développer des applications d'entreprise multitiers. J2EE simplifie ces applications en se basant sur des composants modulaires et standardisés, en offrant un ensemble complet de services pour ces composants (persistance, sécurité, transactions, etc.) et en prenant en charge un grand nombre de détails dans le comportement de l'application, sans requérir de programmation complexe. Cependant, la couche de présentation de l'architecture est basée en grande partie sur HTML, limitant d'autant l'audience aux seuls navigateurs Web. Cette audience demain sera bien plus large. Il faudra compter avec de nouveaux périphériques censés révolutionner notre manière d'envoyer et de recevoir l'information. De ce fait, il est impératif de découpler le contenu de la logique de présentation, et de devenir indépendant des périphériques cibles. XML est la solution.

La publication de documents XML pour le Web est possible en utilisant XSLT pour transformer des documents XML en documents textes. Ces documents cibles sont souvent eux-mêmes des langages basés sur XML (HTML étant l'exception qui confirme la règle, il est basé sur SGML). En tant que métalangage, XML peut également produire d'autres langages Web comme XHTML (pour les navigateurs Web, les WebTV et les périphériques WAP), WML (un langage pour décrire des graphiques en deux dimensions au travers d'XML), SVG, MathML, VoxML (pour la voix)…

XSLT peut transformer un document XML dans n'importe lequel de ces langages cibles. Une fois que nous avons exprimé la solution en XML, on peut l'exporter à la demande pour produire une puissante application Web qui restera indépendantes des périphériques ciblés.

Étude de cas

Dans cette étude de cas, nous allons développer une application pour informatiser les différentes tâches d'une société de location de véhicule (réserver, louer et rendre un véhicule). Nos objectifs : analyse, conception, et implémentation d'un système de gestion de location de véhicule, et création d'un système évolutif capable de cibler de multiples périphériques (navigateurs Web, téléphones WAP, Web TV, etc.). Par souci de simplicité, nous nous contenterons d'implémenter un seul cas : la réservation de véhicule. Les autres cas, comme la modification/annulation d'une réservation, le retour, la location et la mise en maintenance de véhicule pourront être implémentés de la même manière.

Dans le cas d'une réservation de véhicule, le client doit tout d'abord contacter le service des réservations pour réserver une location. L'agent de réservation de location prend le nom du client, et interroge le système pour savoir s'il est déjà inscrit. Si le client est nouveau, l'agent saisit les coordonnées du client.

Ensuite, l'agent demande au client les informations concernant sa réservation, à savoir la date de début, la durée de la location et le type de véhicule qu'il souhaite louer. Une fois ces données entrées dans le système, l'agent vérifie si un véhicule du type demandé par le client est disponible pour la période désirée. Gardons à l'esprit que d'autres requêtes pour des informations supplémentaires sur les réservations (overbooking, disponibilité d'un véhicule) pourront être rajoutés par la suite.

Après avoir donné au client un numéro de confirmation pour sa réservation, utile pour toute future communication, l'agent calcule le montant total de la réservation. A tout moment durant la conversation, le client peut choisir d'arrêter la transaction. Dans ce cas, la réservation est annulée, mais les coordonnées du client sont gardées au cas où il rappellerait.

L'environnement container

Ce projet utilise deux containers Open Source : un container EJB appelé jBoss, et le container de servlet Tomcat de l'Apache Group. Tomcat inclut la servlet Cocoon pour le rendre compatible XML. Les instructions relatives à l'installation d'Apache, Tomcat et Cocoon peuvent être trouvées sur le site de Cocoon. Sont détaillées les installations Unix et Windows NT/2000. jBoss est désormais fourni avec Tomcat, il peut être obtenu sur son site Web (voir les liens en annexe).

Le container de servlet Tomcat/Cocoon

Nous utiliserons Tomcat comme container de servlet pour ce projet. Tomcat, développé par l'Apache Software Foundation, sert de référence officielle pour l'implémentation des technologies Java Servlet 2.2 et des JSP 1.1. Dans cet article, nous utiliserons la dernière version stable de Tomcat 3.2. Sorti récemment, Tomcat 4.0 est basé sur les servlet 2.3 et JSP 1.2.

Comme nous nous basons sur l'API Servlet 2.2, le déploiement se fait au travers de fichiers WAR, un fichier archive Web (similaire au format ZIP ou JAR) utilisé pour regrouper ensemble les pages JSP, les classes JavaBean associées, les servlets, les fichiers XML, les images, et les fichiers XSLT. Comme Tomcat couplé avec jBoss est par essence un serveur d'applications, il est judicieux, dans le cas où ils tourneraient sur la même machine, de les faire fonctionner dans la même machine virtuelle (JVM), cela peut améliorer les temps de réponse d'un facteur de 30.

Cocoon, un framework de publication 100% Java également développé par Apache, se base sur les nouvelles technologies du W3C (tel que DOM, XML et XSL) pour servir le contenu Web. Bien que Cocoon ne soit qu'une servlet déployée au sein de Tomcat, lui apportant la gestion d' XML, il représente l'épine dorsale de notre projet. Cocoon permet la séparation claire entre le contenu et l'utilisation des styles en utilisant le paradigme XML/XSLT.

Le container d'EJB jBoss

jBoss, un serveur d'application EJB 1.1 (avec un support EJB 2.0 partiel) Open Source, incorpore la plupart des fonctionnalités que l'on trouve dans des serveurs d'applications haut de gamme. Comme jBoss utilise le déploiement à chaud pour les fichiers JAR EJB, le déploiement se résume au dépôt d'un fichier JAR EJB dans le répertoire de déploiement. jBoss améliore aussi les fonctionnalités de proxy dynamique de Java 1.3 pour offrir la compilation dynamique des subs et des squelettes. Les prochaines versions devraient intégrer des possibilités de clustering en utilisant la technologie JINI.

Pour notre projet, nous utiliserons jBoss 2.0 (intégrant Tomcat). La version 2.0 inclus l'intégration avec JMS (Java Messaging Service) en utilisant SpyderMQ. JAWS (Just Another Web Storage) délivre la persistance des beans entité, et Minerva gère les connections aux bases de données.

L'architecture en couches

L'architecture en couches J2EE, illustrée sur la figure 1, permet de minimiser le coût d'évolution des fonctionnalités d'une application et des mises à jour des technologies d'implémentations. Ces couches peuvent être schématiquement catégorisée en couche de présentation, couche application, couche de service, couche entreprise, et couche de persistance.

La couche de présentation

Pour nos besoins, nous nous attarderons surtout sur la couche de présentation. Celle-ci est confiée au sein de l'architecture J2EE à HTML/JSP (avec JavaScript pour les validations coté client), ce qui implique dès le départ que notre audience sera limitée aux navigateurs Web. Une fois la logique de présentation séparée des données, la présentation peut être réalisée en utilisant XSLT pour le texte (HTML et XHTML), XSLFO pour les fichiers binaires PDF et SVG pour les images.

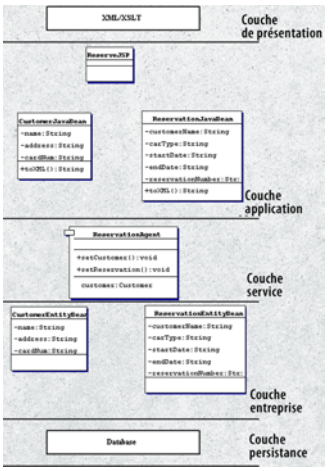


Figure 1. L'architecture en couches J2EE.

Notre couche de présentation n'utilise ni HTML, ni les fichiers de présentation JSP associés. A la place, elle utilise des fichiers XML bruts avec d'autres fichiers XML créés dynamiquement à la volée par la couche application. Ensuite, en fonction du périphérique qui a effectué la demande, un fichier XSLT délivre la présentation. Le moteur de publication va déterminer le type de périphérique et choisir en fonction la feuille de style XSLT appropriée. Cela nous permet de séparer la présentation du contenu et de définir clairement la division des responsabilités.

En termes de modèle-vue-contrôleur (MVC), le document XML joue le rôle du modèle, le JSP dans la couche applicatiion celui du contrôleur, et le fichier XSLT est la vue. En changeant simplement de fichier XSLT, différentes vues peuvent être ainsi réalisées.

La couche applicative

La réservation d'un véhicule peut être divisée en trois parties : ajouter un client, mettre à jour client et effectuer une réservation. Un contrôle JSP (reserve.jsp) défini la couche applicative et délègue les responsabilités aux JavaBeans. Notons que ici, le JSP doit être considéré comme un servlet, il ne contient aucune information de présentation. Pour le code source du fichier reserve.jsp, voir listing 1 en annexe.

Comme nous l'avons vu dans le code précédent, le JSP de contrôle utilise deux JavaBeans : un bean client (Customer), et un bean réservation (Reservation). Cela simplifie le JSP en faisant une abstraction totale de tous les problèmes complexes d'invocation de méthodes, et délègue les responsabilités aux JavaBeans. De ce fait, le JSP ne s'occupe que de la logique simple. Il route ensuite les contrôles vers différentes pages en fonction de cette logique. Cela simplifie également le test du JSP, on peut déjà écrire ce test pour chaque bean, et le tester sans la complexité à avoir à déployer le bean dans le container de servlet.

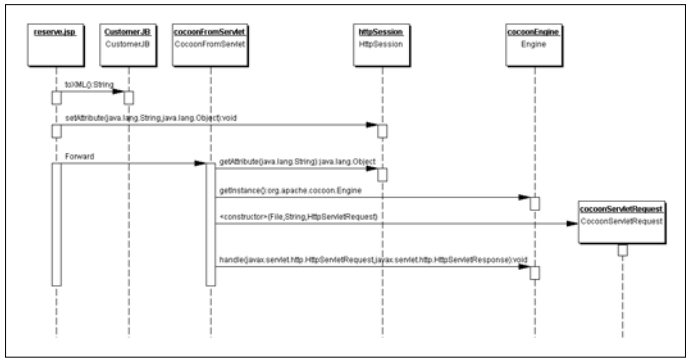
Au bout de la chaîne, le bean produit une représentation XML de lui-même. Chaque bean possède une méthode toXML() qui retourne une chaîne xmlisée des données représentées dans le bean. Actuellement, ces fonctions n'effectuent que des concaténations de chaînes des attributs du bean, mais l'on pourrait réaliser une représentation plus complexe basée sur DOM ou JDOM.

Le bean Customer

Le bean Customer récupère et défini un client. Le diagramme de séquence montre clairement que suite à un POST d'un nom de client par l'agent, le contrôle reserve.jsp invoque getCustomer() dans le JavaBean. Le bean à son tour fait appel au middleware vers une session EJB sans état appelée ReservationAgent. Si le client existe dans la base de données, alors le JavaBean se remplit des informations concernant le client. La page reserve.jsp appelle ensuite la méthode toXML() sur le bean Customer et renvoie la chaîne résultante représentant le document XML à Cocoon pour en assurer le rendu avec la feuille de style XSLT correspondante. Pour le code du bean Customer, voir listing 2.

Cette procédure simple se répète pour tout les autres cas d'utilisation, mais il est bon d'explorer plus en détail ce qui arrive lorsqu'une chaîne XML, représentant le bean Customer, est envoyée vers Cocoon. Une limitation de la spécification actuelle des servlets est que la sortie d'une servlet ne peut être redirigée vers l'entrée d'une autre. La figure 2 montre comment contourner cette limitation. Le JSP place la chaîne, qui contient la représentation XML du JavaBean, dans le contexte de session. Elle est ensuite envoyée

Figure 2 : Envoi des données vers la servlet Cocoon.



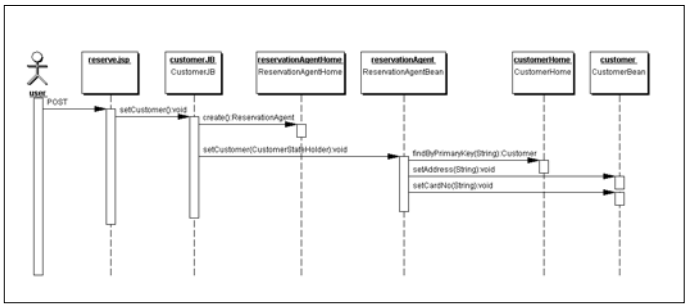


Figure 3 : Mise à jour d'un client.

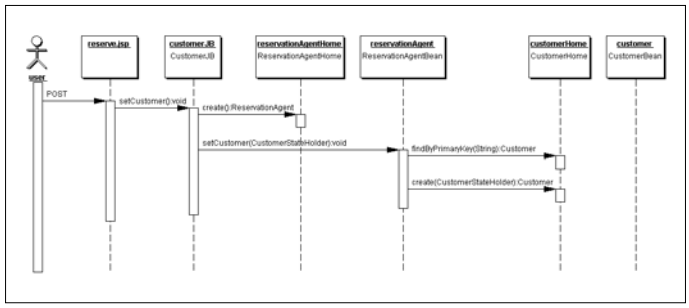


Figure 4 : Créer un nouvel utilisateur.

vers une autre servlet CocoonFrom Servlet, qui récupère l'objet String à partir du contexte de session pour encapsuler la chaîne XML dans un objet requête HTTP. Il lance enfin Cocoon et envoi cette requête HTTP pour qu'elle soit traitée. Il est à noter que ce contournement n'a été testé en profondeur qu'avec Cocoon version 1.7.4.

Que le client existe ou non, l'utilisateur est ensuite dirigé vers une page pour la saisie de la fiche client. Si le client existe, les champs sont déjà remplis et peuvent être modifiés au besoin. La figure 3 illustre le diagramme de séquence pour la mise à jour d'une fiche client. La figure 4 montre le diagramme de séquence pour un nouvel utilisateur. Si le client n'existe pas, seul le champ du nom (qui à été rentré préalablement) est renseigné.

A la suite du retour de ce formulaire au contrôle JSP, la page invoque la méthode setCustomer() du bean Customer qui, encore une fois, fait appel au middleware pour mettre à jour les données. Le client existe maintenant de manière persistante dans le système de stockage, et le contrôle JSP stocke l'objet string dans le contexte de session. Cet objet est interrogé chaque fois que la page de contrôle est appelée. Lors des prochains appels à ce JSP lors de cette session, le contrôle saura, par l'existence de cet objet, qu'il traite un client déjà stocké.

Le bean Reservation

Le bean Reservation récupère et stocke les réservations. Une fois les données clients stockées, la figure 5 décrit comment le JSP réalise une réservation. Pour accomplir cela, le contrôle JSP utilise un deuxième bean, Reservation.

Le nom du client actuellement traité est passé à ce bean, le JSP appelle la méthode toXML() puis effectue le rendu avant de l'envoyer à Cocoon (listing 3 en annexe).

Une fois les détails de la réservation remplis, l'agent renvoie les données à nouveau vers le JSP, qui invoque la fonction setReservation() du bean de réservation. Le bean fait encore une fois appel au middleware afin de rendre les données persistantes et de retourner un numéro de réservation unique.

La couche de service

La couche de service contient des contrôleurs. Dans l'architecture à couches J2EE, ce sont les beans de session qui représentent ces contrôleurs. Les beans de session représentent les cas d'usage, de fait ils délimitent les bornes transactionnelles. Ce positionnement objet nous dirige vers une architecture basée sur les services, où les beans de session qui représentent les flux de données, contrôlent les beans entité. On peut considérer que les beans de session sont une extension du client, côté serveur.

Les beans de session sont de deux types : avec et sans maintien des états. Dans le premiers cas, le bean ne maintient pas les états entre les différents appels de méthodes. Ce type de session, plus complexe à mettre en place, requiert typiquement un grand nombre de paramètres qui doivent être passés aux méthodes, ce qui provoque un fort trafic. A l'opposé, les beans avec maintien conservent les états du client. Elles sont plus simples à créer, et requièrent moins de paramètres à passer de la part du client.

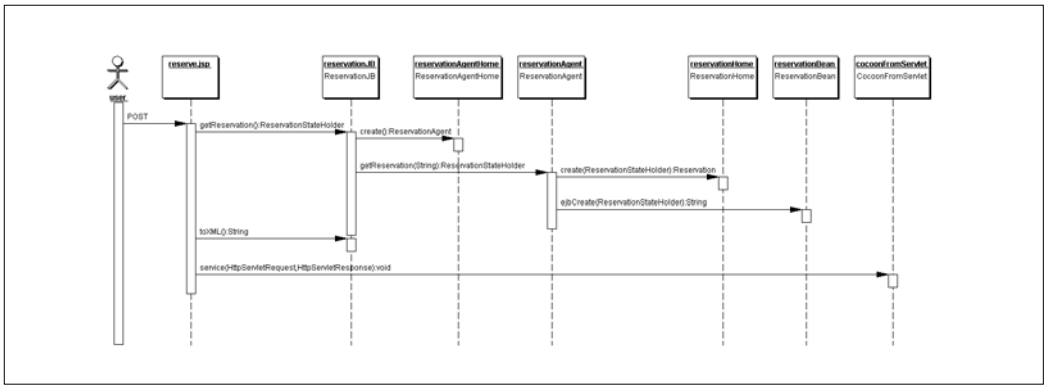
Les sessions sans maintien d'état permettent une grande scalabilité, elles ne sont liées à aucun client particulier. Une de ces sessions peut prendre en charge de multiples clients, quelques centaines de ces sessions peuvent prendre en charge des milliers de clients simultanés. Par leur nature, les sessions avec maintien d'état sont liées à un client sur une base d'une session pour chaque client. Bien qu'ils n'y ait pas de règles fixes, il est conseillé pour les EJB créés pour des systèmes susceptibles de prendre en charge des milliers de clients simultanés d'utiliser des sessions sans maintien d'état.

Dans notre cas, nous représentons le cas d'usage de réservation de véhicule comme une session sans maintien d'état appelée Reservation Agent, composée de quatre méthodes : getCustomer(), setCustomer(), getReservation() et setReservation(). C'est ReservationAgent qui contrôle les flux de données et manipule les entités des beans Customer et Reservation (listing 4).

La méthode getCustomer() effectue une recherche en appelant findByPrimaryKey() sur l'objet distant du bean entité et invoque getCustomerStateHolder() sur le bean entité. Un objet value permet de transférer de grandes quantités de données liées à un objet particulier au travers du middleware, en un seul objet sérialisable. Cela réduit le trafic réseau. Il faudra être prudent en utilisant ce type de conception, des doubles des données sont maintenant présents dans le modèle. De manière idéale, il ne faut les utiliser que si les données sont en lecture seule, ou que si un blocage peut être effectué sur les données coté backend lors de leur extraction.

La méthode setCusomer() recherche si un client existe dans le bean Customer en appelant findByPrimaryKey() sur l'objet distant. Si le client n'existe pas, une nouvelle entité est créée pour le représenter. S'il existe, les informations sont mises à jour dans son bean entité. Les méthodes getReservation() et setReservation() manipulent de manière similaire les beans Reservation.

Figure 5. Effectuer une réservation.



La couche entreprise

La couche entreprise d'une application EJB contient les objets partagés, généralement entre plusieurs clients. En première approche, une bean entité représente une ligne de la base de données et représente cette ligne dans un objet Java (un bean entité) ce qui permet de la manipuler avec ses méthodes accesseurs publiques.

Les beans entité sont de deux types : persistance gérée par le container (CMP), et persistance gérée par le bean (BMP). Dans le premier cas, les opérations de base de données

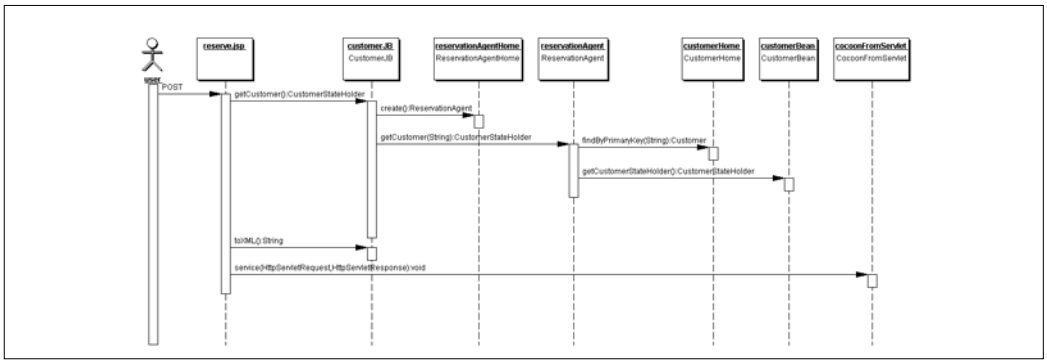


Figure 6. Récupérer un client.

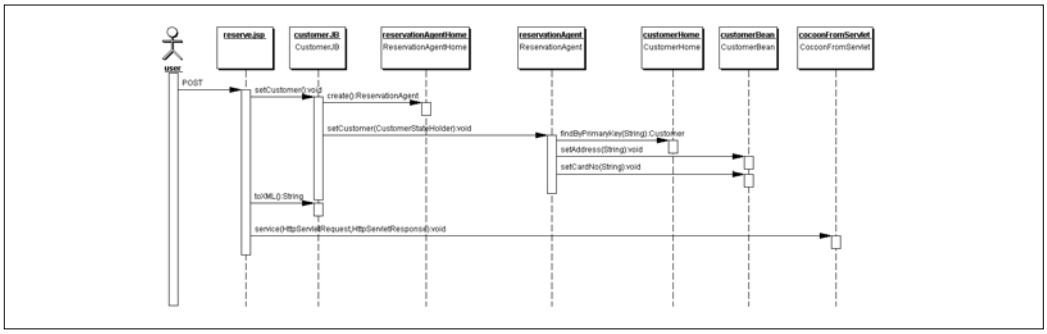


Figure 7. Stocker un client.

(création, lecture, mise à jour et suppression) sont déléguées au container. Dans le second, c'est au développeur d'écrire le code SQL pour les interactions avec la base de données. Pour l'intégration avec des systèmes anciens, ou des structures de bases de données complexes, le mode BMP est la seule option viable. On peut cependant se féliciter que la spécification EJB 2.0 améliore sensiblement les fonctionnalités des beans de type CMP.

Par souci de simplicité, ces beans seront modélisés comme persistantes gérées par le container, ce qui nécessite de préciser quels attributs des beans seront persistants. On déclare ces champs persistants dans le fichier XML de déploiement, le fichier ejb-jar.xml (listing 5).

Cette couche contient deux beans CMP, l'entité CustomerBean représente un client, et l'entité ReservationBean représente une réservation. Les deux étant modélisés comme des objets partagés persistants, ils sont modélisés comme des beans entités.

Le bean entité Customer

Le bean customer comporte trois champs persistants : name, address et cardNo (pour le numéro de carte de crédit). Par souci de simplicité, ces trois champs sont modélisés par des chaînes de caractères. La clé primaire est le champ nom (name). Le container jBoss va automatiquement mapper ces champs vers une table Customer, et effectuer automatiquement les conversions Java/SQL. Pour cela, il est nécessaire que cette fonction implémente getStateHolder() et setStateHolder().

Ce bean implémente également ejbFindByPrimaryKey() et l'optionnel ejbCreate(). Cette fonction crée une bean entité Customer qui ajoute une ligne client à la base de données (listing 6).

Le bean entité Reservation

Le bean Reservation comporte cinq champs persistants : reservationNo, startDate, endDate, carType et customerName.

La clé primaire est reservationNo, customerName étant une clé étrangère. Tout comme pour le bean customer, les fonctions setStateHolder(), getStateHolder(), ejbFindByPrimaryKey et ejbCreate sont toutes implémentées (listing 7).

La couche persistance

Dans cet article, nous utiliserons la base de données Open Source fournie avec jBoss, Hypersonic. Pour 100 Ko, Hypersonic est un logiciel admirable. Cependant, pour des utilisations plus sérieuses, on pourra se reporter vers d'autres bases de données Open Source plus matures et supportant l'ACIDité.

Les résultats

Remplacer la couche de présentation par une couche XML nous donne maintenant de la flexibilité pour cibler divers types de périphériques. Une feuille de style différente devra être écrite pour chaque périphérique. Cet exemple inclus une feuille de style pour téléphone WAP (testé sur un émulateur WAP Nokia), et une pour navigateur Web (testée avec IE5, Netscape 4.7 et Mozilla).

Chaque feuille de style doit être listée dans le fichier XML, de manière statique ou dynamique, avec un marqueur supplémentaire pour identifier le type de périphérique ciblé. Cocoon identifie alors le type de chaque périphérique à partir de l'entête et utilise la feuille de style appropriée pour transformer le XML dans le langage compris par le périphérique.

Vous pouvez télécharger le code source complet de cet article dans la section Annexe. Il a été développé dans un environnement Extreme Programming, en utilisant les techniques de développement itératif, de test unitaire, de maîtrise du risque et de programmation par paire. Les tests ont été réalisés en utilisant le framework de test JUnit. Le projet entier peut être compilé avec Ant. Le résultat est un fichier ear qui peut être déplacé vers le répertoire deploy de jBoss.

Le code source est également distribué en tant que projet Open Source sur SourceForge, que l'on retrouve sur jaba.sourceforge.net. Vous y trouverez des mises à jour du code, et un how to sur l'installation de JBoss/Tomcat. Toute personne voulant se joindre au projet étant bien évidemment la bienvenue.

Dans cet article, nous avons vu comment remplacer la couche de présentation de J2EE, orientée vers les navigateurs Web par une couche XML plus générique, qui nous permet en prime d'être indépendant du type de périphérique en séparant clairement les données (XML) de la présentation (XSLT). On peut maintenant effectuer des représentations différentes des mêmes données en utilisant des feuilles de styles XSLT. Créer un serveur d'applications Open Source XML a rendu tout cela possible.

XML s'impose clairement comme le meilleur choix tant les méthodes de diffusion de l'information se multiplient au fur et à mesure que la révolution de l'information avance. XML nous permet en tant que programmeurs de nous concentrer uniquement sur les données et de déléguer les responsabilités de sa présentation ailleurs. De plus, XML nous apporte une constance dans un monde où l'on ne sait jamais vraiment comment les données vont être présentées. Finalement, XML ouvre un monde de possibilités où l'ubiquité de l'information peut enfin se réaliser. ●

Traduction Guillaume Louel. "Add XML to your J2EE applications" par Eoine Lane, publié par JavaWorld (www.javaworld.com), copyright IDG.net, 2001. Traduit et republié avec permission. http://www.javaworld.com/javaworld/jw-02-2001/jw-0209-xmlj2ee.html?

ALLER PLUS LOIN

Retrouvez le code source complet de cet article :
http://www.devreference.net/magazine/v205/jw-0209-xmlj2ee.zip

Le site dédié au projet présenté dans cet article :
http://jaba.sourceforge.net/

Tout savoir sur les feuilles de style XSLT :
http://www.w3.org/Style/XSL

XHTML, réécriture XML de HTML :
http://www.w3.org/MarkUp

Le site de Cocoon :
http://xml.apache.org/cocoon/index.html

Le site de jBoss :
http://www.jboss.org/

Ant, un outil d'aide à la compilation :
http://jakarta.apache.org/ant/index.html

Listing 1 : Reserve.jsp.

```
// Reserve.jsp page

<jsp:useBean id="customer" scope="request"
class="com.valtech.bootcamp.carRental.web.customer.CustomerJB"/>
<jsp:useBean id="reservation" scope="request"class=
"com.valtech.bootcamp.carRental.web.reservation.ReservationJB"/>

<%
  if( request.getParameter("btnReset") != null ) {
    session.invalidate();
    session = request.getSession();
   } %>

<% String client = (String)session.getAttribute("USER"); %>

// Checks if the client exists
<% if(client == null){ %>

  // Checks if the name field has been filled in
  // If the name field is null,
  // then the user is presented with a login file
  <% if (request.getParameter("name") == null)
  { %>
     <jsp:forward page="login.xml" />

   // if the address field is null, then the user is presented
   // with a customer details form
   <% } else if (request.getParameter("address") == null)
       { %>
         <jsp:setProperty name="customer" property="*"/>
       <% customer.getCustomer(); %>
       <% String customerXML = customer.toXML(); %>
       <% session.setAttribute("CURRENTDOC", customerXML); %>
       <jsp:forward page="/servlet/com.valtech.bootcamp.
         carRental.web.cocoon.CocoonFromServlet" />

      <% } else { %>

  // The user has completed the form, and the customer's
  // data has been persisted
  // A String object is also saved to the session context
  // to show that the customer has been
  // validated for this session
       <jsp:setProperty name="customer" property="*"/>
       <% customer.setCustomer(); %>
       <% session.setAttribute("USER", "true"); %>
       <jsp:forward page="reserve.jsp" />
      <% } %>
<% } else { %>

   // Checks if the stateDate field has been filled in
   // If the field is null, the user is presented with a
   // reservation page with the name of the customer filled in
  <% if (request.getParameter("startDate") == null) { %>
    <% reservation.setCustomerName(customer.getName()); %>
    <% String reservationXML = reservation.toXML(); %>
    <% session.setAttribute("CURRENTDOC", reservationXML); %>
    <jsp:forward page="/servlet/com.valtech.bootcamp.carRental.
      web.cocoon.CocoonFromServlet" />
     <!-- <jsp:forward page="reservation.xml" /> -->

  // If the startDate field has been filled in, the full reservation
  // details are displayed to the customer
  <% } else { %>
      <jsp:setProperty name="reservation" property="*"/>
      <% reservation.setReservation(); %>
    <% String reservationXML = reservation.toXML(); %>
    <% session.setAttribute("CURRENTDOC", reservationXML); %>
    <jsp:forward page="/servlet/com.valtech.bootcamp.carRental.web.
      cocoon.CocoonFromServlet" />
  <% } %>
<% } %>
```

Listing 2. Le bean Customer.

```
/**
* Client side bean for the Customer
* @author Valtech UK
*/

public class CustomerJB {
/**
* the customer name
*/
private String name = "";

/**
* the customer address
*/
private String address = "";

/**
* the card number of the customer
*/
private String cardNum = "";

/**
* the state holder object
*/
private CustomerStateHolder csh;

/**
* @param name the name of the customer
*/
public CustomerJB(String name)
{
    this.name = name;
}

public CustomerJB(){}

/**
* Full arguments constructor.
* @param name the name of the customer
* @param address the address of the customer
* @param cardNum the card number of the customer
*/
public CustomerJB(String name, String address, String cardNum)
{
    this.name = name;
    this.address = address;
    this.cardNum = cardNum;
}

/**
* gets the name of the customer
*/
public String getName()
{
    return name;
}

/**
  * sets the customer name
  */
public void setName(String name)
{
    this.name = name;
}

/**
* gets the address of the customer
*/
public String getAddress()
{
   return address;
}

/**
* sets the address of the customer
*/
public void setAddress(String address)
{
   this.address=address;
}

/**
* gets the card number
*/
public String getCardNum()
{
   return cardNum;
}

/**
* sets the card number
*/
public void setCardNum(String cardNum)
{
   this.cardNum=cardNum;
}

/**
  * Calls getCustomer() on the ReservationAgent facade and
  * extracts this state holder into the bean attribute fields.
  */
public CustomerStateHolder getCustomer() {
   CustomerStateHolder csh = null;
   System.setProperty("java.naming.factory.initial",
   "org.jnp.interfaces.NamingContextFactory");
   System.setProperty("java.naming.provider.url",
   "localhost:1099");
   try
   {
      Context initial = new InitialContext();
      ReservationAgentHome resAgentHome = (ReservationAgentHome)
         PortableRemoteObject.narrow
         (initial.lookup("ReservationAgentBean"),
         ReservationAgentHome.class);
         ReservationAgent reservationAgent = resAgentHome.create();
         csh = reservationAgent.getCustomer(this.name);
   }
   catch(Exception e)
   {
      System.out.print("Test Error");
      //e.printStackTrace();
   }
   if (csh != null)
   {
      this.address = csh.getAddress();
      this.cardNum = csh.getCardNo();
```

```
        this.name = csh.getName();
    }
    return csh;
}

/**
*  calls setCustomer() on the ReservationAgent facade.
*/
public void setCustomer() {

    System.setProperty("java.naming.factory.initial",
        "org.jnp.interfaces.NamingContextFactory");
    System.setProperty("java.naming.provider.url","localhost:1099");

    try
    {
        Context initial = new InitialContext();
        ReservationAgentHome resAgentHome = (ReservationAgentHome)
            PortableRemoteObject.narrow(initial.lookup
            ("ReservationAgentBean"), ReservationAgentHome.class);
            ReservationAgent reservationAgent = resAgentHome.create();
            reservationAgent.setCustomer(
                this.createCustomerStateHolder());
    }
    catch(Exception e)
    {
        System.out.print("Test Error");
        e.printStackTrace();
    }
}

/*
* places bean attributes into a CustomerStateHolder.
*/
public CustomerStateHolder createCustomerStateHolder()
{
    return new CustomerStateHolder(this.name, this.address,
                                   this.cardNum);
}

/*
* removes the occurrence of a customer by calling
* removeCustomer() on the ReservationAgent bean
*/
public void removeCustomer(CustomerStateHolder csh)
{
    System.setProperty("java.naming.factory.initial",
        "org.jnp.interfaces.NamingCon textFactory");
    System.setProperty("java.naming.provider.url","localhost:1099");

    try
    {
        Context initialb = new InitialContext();
        ReservationAgentHome raHome =
            (ReservationAgentHome)PortableRemoteObject.narrow(
                initialb.lookup("ReservationAgentBean"),
                ReservationAgentHome.class);
        ReservationAgent ra=raHome.create();
        ra.removeCustomer(csh.getName());
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

/*
* Generates a String of XML for the customer details entry form.
* Note that it is planned to encapsulate this detail in another
object.
*/
public String toXML()
{
    return new String( "<?xml version=\"1.0\"?>" +
        "<?cocoon-process type=\"xslt\"?>" +
        "<?xml-stylesheet href=\"/stylesheet/form-html.xsl\"
            type=\"text/xsl\"?>"  +
        "<?xml-stylesheet href=\"/stylesheet/form-wml.xsl\"
            type=\"text/xsl\" media=\"wap\"?>"  +
        "<form action=\"reserve.jsp\">" +
        "<head>" +
        "<title>ABC Car Rental</title>" +
        "</head>" +
        "<input type=\"text\" name=\"name\" value=\"" +
            this.getName() + "\" size=\"11\"/>" +
        "<input type=\"text\" name=\"address\" value=\""
            + this.getAddress() + "\" size=\"11\"/>" +
        "<input type=\"text\" name=\"cardNum\" value=\""
            + this.getCardNum() + "\" size=\"11\"/>" +
        "</form>" );
}
}
```

Listing 3. Le bean Reservation.

```
/**
* Client side java bean for the Reservation object.
* @author Valtech UK
* @version 1.0
*/
public class ReservationJB {

/**
* the car type of the reservation
*/
private String carType = "";

/**
* the name of the customer
*/
private String customerName = "";

/**
* the end date of the reservation
*/
private String endDate = "";

/*
* the reservation number of the reservation
*/
private String reservationNumber = "";

/**
* the start date of the reservation
*/
private String startDate = "";

/**
* @param reservationNumber the unique reservation number of the
reservation.
* @param startDate the start date of the reservation
* @param endDate the end date of the reservation
* @param carType the type of car specified in the reservation.
* @param customerName the name of the customer making the reservation
*/
public ReservationJB(String reservationNumber, String startDate,
      String endDate, String carType, String customerName)
{
    this.carType = carType;
    this.customerName = customerName;
    this.endDate = endDate;
    this.startDate = startDate;

    /*
     * note the reservation number is set to a default
     * here as the value gets reset.
     */
    this.reservationNumber = "555";
}

/**
* populates the state holder attributes into the bean
* @param rsh the state holder containing details of the reservation
*/
public ReservationJB(ReservationStateHolder rsh)
{
    this.carType = rsh.getCarType();
    this.customerName = rsh.getCustomerName();
    this.endDate = rsh.getEndDate();
    this.startDate = rsh.getStartDate();
    this.reservationNumber = rsh.getReservationNumber();
}

/**
  * overridden for the JUnit testing framework
  * @param rjb the ReservationJB that is being used in
  * the comparison to the calling object
  */
public boolean equals( ReservationJB rjb)
{
    return (rjb.startDate.equals(this.startDate) &&
        rjb.endDate.equals(this.endDate) &&
        rjb.carType.equals(this.carType) &&
        rjb.customerName.equals(this.customerName));
}

/**
* Default constructor
**/
public ReservationJB(){}

/**
* gets the car type
*/
public String getCarType(){ return carType; }

/**
* sets the car type
*/
public void setCarType(String carType){ this.carType = carType; }

/**
* gets the customer name
*/
public String getCustomerName(){ return customerName; }

/**
* sets the customer name
*/
public void setCustomerName(String customerName){ this.customerName =
customerName; }

/**
* gets the end date
*/
```

```
public String getEndDate(){ return endDate; }

/**
* sets the end date
*/
public void setEndDate(String endDate){ this.endDate = endDate; }

/**
* gets the reservation number
*/
public String getReservationNumber(){ return reservationNumber; }

/**
* sets the reservation number
*/
public void setReservationNumber(String reservationNumber){
this.reservationNumber = reservationNumber; }
/**
* gets the start date
*/
public String getStartDate(){ return startDate; }

/**
* sets the start date
*/
public void setStartDate(String startDate){ this.startDate =
startDate; }

/**
* calls setReservation() on the ReservationAgent facade.
*/
public void setReservation()
{
   System.setProperty("java.naming.factory.initial",
      "org.jnp.interfaces.NamingContextFactory");
   System.setProperty("java.naming.provider.url","localhost:1099");
   try
   {
      Context initial = new InitialContext();
      ReservationAgentHome resAgentHome =
         (ReservationAgentHome)PortableRemoteObject.narrow(
            initial.lookup("ReservationAgentBean"),
      ReservationAgentHome.class);
      ReservationAgent reservationAgent = resAgentHome.create();
      String resCreate =
      reservationAgent.setReservation(getReservationStateHolder());
      System.out.println(resCreate);
      setReservationNumber(resCreate);
   }
   catch(Exception e)
   {
      System.out.print("Test Error");
      e.printStackTrace();
   }
}

/**
* returns a state holder of the reservation.
*/
public ReservationStateHolder getReservation()
{
   ReservationStateHolder rsh = null;
   System.setProperty("java.naming.factory.initial",
      "org.jnp.interfaces.NamingContextFactory");
   System.setProperty("java.naming.provider.url",
      "localhost:1099");
   try
   {
      Context initial = new InitialContext();
      ReservationAgentHome resAgentHome =
         (ReservationAgentHome)PortableRemoteObject.
         narrow(initial.lookup("ReservationAgentBean"),
         ReservationAgentHome.class);
      ReservationAgent reservationAgent = resAgentHome.create();
      rsh = reservationAgent.getReservation(this.reservationNumber);
   }
   catch(Exception e)
   {
      System.out.print("Test Error");
      e.printStackTrace();
   }
   return rsh;
}

/**
* returns a state holder constructed from the bean attributes
*/
public ReservationStateHolder getReservationStateHolder()
{
   return new  ReservationStateHolder(this.reservationNumber,
   this.startDate, this.endDate, this.carType, this.customerName);
}

public void getRegistration()
{
}

/**
* generates an XML document for the reservation entry form.
*/
public String toXML()
{
   return new String( "<?xml version=\"1.0\"?>" +
      "<?cocoon-process type=\"xslt\"?>" +
      "<?xml-stylesheet href=\"/stylesheet/
         form-html.xsl\" type=\"text/xsl\"?>"  +
      "<?xml-stylesheet href=\"/stylesheet/
         form-wml.xsl\" type=\"text/xsl\" media=\"wap\"?>"  +
      "<form action=\"reserve.jsp\">" +
      "<head>" +
      "<title>ABC Car Rental</title>" +
      "</head>" +
      "<input type=\"text\" name=\"customerName\"
         value=\"" + this.getCustomerName() + "\" size=\"11\
         "/>" +
      "<input type=\"text\" name=\"carType\" value=
         \"" + this.getCarType() + "\" size=\"11\"/>" +
      "<input type=\"text\" name=\"startDate\" value=\""
         + this.getStartDate() + "\" size=\"11\"/>" +
      "<input type=\"text\" name=\"endDate\" value=
         \"" + this.getEndDate() + "\" size=\"11\"/>" +
      "<input type=\"text\" name=\"reservationNo\"
         value=\"" + this.getReservationNumber() +
         "\" size=\"11\"/>" +
      "</form>" );
}
}
```

Listing 4. ReservationAgent.

```
/**
* Remote interface for ReservationAgent session EJB


* @author Valtech UK
*/
public interface ReservationAgent extends EJBObject
{
    /**
     * gets the customer
     */
    CustomerStateHolder getCustomer(String name) throws
RemoteException;

    /**
     * sets the customer
     */
    void setCustomer(CustomerStateHolder customer) throws
RemoteException;

    /**
     * sets the reservation
     */
    String setReservation(ReservationStateHolder reservation) throws
    RemoteException;

    /**
     * gets the reservation
     */
    ReservationStateHolder getReservation(String reservationNumber)
throws
    RemoteException;

    /**
     * removes the customer
     */
    void removeCustomer(String name) throws RemoteException;
    /** @link dependency */
    /*#ReservationAgentBean lnkSession1Bean;*/
}
```

Listing 5. Le fichier XML de déploiement ejb-jar.xm.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Enterprise
JavaBeans
1.1//EN" "http://java.sun.com/j2ee/dtds/ejb-jar_1_1.dtd">

<ejb-jar>
<description>This contains the Service elements of
  the CarRental System< /description>
    <display-name>ServiceEJB</display-name>
    <enterprise-beans>
      <session>
   <ejb-name>ReservationAgentBean</ejb-name>
    <home>
com.valtech.bootcamp.carRental.service.
reservationAgent.ReservationAgentHome< /home>
    <remote>
com.valtech.bootcamp.carRental.service.
reservationAgent.ReservationAgent< /remote>
    <ejb-class>
com.valtech.bootcamp.carRental.service.
reservationAgent.impl.ReservationAgentBean</ejb-class>
   <session-type>Stateless</session-type>
   <transaction-type>Container</transaction-type>
    <ejb-ref>
         <ejb-ref-name>ReservationBean</ejb-ref-name>
         <ejb-ref-type>Entity</ejb-ref-type>
           <home>
com.valtech.bootcamp.carRental.business.reservation.ReservationHome</
home>
          <remote>
```

```
com.valtech.bootcamp.carRental.business.reservation.Reservation</
remote>
        <ejb-link>ReservationBean</ejb-link>
       </ejb-ref>
       <ejb-ref>
        <ejb-ref-name>CustomerBean</ejb-ref-name>
        <ejb-ref-type>Entity</ejb-ref-type>
          <home>
com.valtech.bootcamp.carRental.business.customer.
CustomerHome</home>
         <remote>com.valtech.bootcamp.
         carRental.business.customer.Customer<
/remote>
        <ejb-link>CustomerBean</ejb-link>
       </ejb-ref>
       </session>
  <entity>
   <ejb-name>CustomerBean</ejb-name>
   <home>com.valtech.bootcamp.carRental.business.
   customer.CustomerHome</home>
   <remote>com.valtech.bootcamp.carRental.business.
      customer.Customer</remote>
   <ejb-class>
com.valtech.bootcamp.carRental.business.customer.
  impl.CustomerBean</ejb-class>
   <persistence-type>Container</persistence-type>
   <prim-key-class>java.lang.String</prim-key-class>
   <primkey-field>name</primkey-field>
   <reentrant>False</reentrant>
       <transaction-type>Bean</transaction-type>
   <cmp-field><field-name>name</field-name></cmp-field>
   <cmp-field><field-name>address</field-name
      ></cmp-field>
   <cmp-field><field-name>cardNo</field-name
       ></cmp-field>
  </entity>
  <entity>
      <ejb-name>ReservationBean</ejb-name>
   <home>com.valtech.bootcamp.carRental.business.
reservation.ReservationHome<
/home>
   <remote>com.valtech.bootcamp.carRental.business.
reservation.Reservation<
/remote>
   <ejb-class>
com.valtech.bootcamp.carRental.business.reservation.impl.
ReservationBean<
/ejb-class>
   <persistence-type>Container</persistence-type>
   <prim-key-class>java.lang.String</prim-key-class>
   <primkey-field>reservationNumber</primkey-field>
   <reentrant>False</reentrant>
       <transaction-type>Bean</transaction-type>
   <cmp-field><field-name>reservationNumber
      </field-name></cmp-field>
   <cmp-field><field-name>startDate</field-name>
      </cmp-field>
   <cmp-field><field-name>endDate</field-name>
      </cmp-field>
   <cmp-field><field-name>carType</field-name>
      </cmp-field>
   <cmp-field><field-name>customerName</field-name>
      </cmp-field>
  </entity>
    </enterprise-beans>
  </ejb-jar>
```

Listing 6. Le bean entité Customer.

```
/**
* Remote interface for Customer EJB
* @author Valtech UK
*/

public interface Customer extends EJBObject
{
  /**
   * Returns CustomerStateHolder containing parameters
   * mapping to Entity CMP fields
    */
  CustomerStateHolder getCustomerStateHolder() throws
RemoteException;

  /**
   * sets CustomerStateHolder
   * @param csh The CustomerStateHolder
    */
  void setCustomerStateHolder(CustomerStateHolder csh) throws
RemoteException;

  /**
   * sets the address
    */
  void setAddress(String address) throws RemoteException;

  /**
   * sets the card number
    */
  void setCardNo(String cardNo) throws RemoteException;

}
```

Listing 7. Le bean entité Reservation.

```
/**
* Remote interface for ReservatEion EJB
* @author Valtech UK
*/

public interface Reservation extends EJBObject {


    /**
     * gets the reservation number
      */
   public String getReservationNumber() throws RemoteException;

    /**
     * sets the reservation number
      */
   public void setReservationNumber(String resnum) throws
RemoteException;

    /**
     * gets the start date of the reservation
      */
   public String getStartDate() throws RemoteException;

    /**
     * sets the start date of the reservation
      */
   public void setStartDate(String newStartDate) throws
RemoteException;

    /**
     * gets the end date
      */
   public String getEndDate() throws RemoteException;

    /**
     * sets the end date
      */
   public void setEndDate(String newStartDate) throws
RemoteException;

    /**
     * sets the car type
      */
   public void setCarType(String newCarType) throws RemoteException;

    /**
     * gets the car type
      */
   public String getCarType()  throws RemoteException;

    /**
     * sets the customer name
      */
   public void setCustomerName(String newCustomerName) throws
RemoteException;

    /**
     * gets the customer name
      */
   public String getCustomerName() throws RemoteException;

    /**
     * gets the resrvation state holder
      */
   public ReservationStateHolder getReservationStateHolder() throws
RemoteException;

    /**
     * sets the reservation state holder
      */
   public void setReservationStateHolder(ReservationStateHolder rsh)
throws
RemoteException;

}//end of class
```

XML dans le .NET Framework

Dans une premier temps, nous allons nous attacher à comprendre comment le .NET Framework utilise XML en interne. En second lieu, nous présenterons les outils mis à disposition des développeurs pour manipuler des documents XML.

Médéric Morel, *est le directeur technique de Neoxia. À ce titre, il intervient dans le cadre de missions de qualification de technologie, de conception d'architecture et d'audit technique.*

XML est partout. Initialement conçu pour l'échange de données, il est dorénavant utilisé pour bien d'autres usages. On le retrouve aussi bien dans la description d'interfaces, dans la description de méta-données, dans les protocoles de communication comme SOAP que dans la plupart des bases de données actuelles. Bref, XML s'est rapidement imposé comme le format de données incontournable.

Microsoft a rapidement fourni les outils nécessaires à la génération et au traitement de documents XML. Le plus connu est le composant MSXML qui inclut le parseur mais également un transformateur XSL-T. MSXML existe aujourd'hui dans sa version 4.0.

Microsoft .NET Framework sortira dans sa version finale au mois de janvier 2002. Le .NET Framework fournit bien plus qu'une palette d'outils complète pour la manipulation du XML. Il est entièrement basé sur ce format. Le .NET Framework utilise XML en interne pour représenter ses données mais fournit également les classes nécessaires pour une manipulation simple et intuitive des documents XML.

Les fonctionnalités XML du .NET Framework et de la plate-forme de développement Visual Studio .NET, permettent une utilisation massive de l'XML. Microsoft entend visiblement industrialiser l'usage d'XML dans les développements futurs.

.NET supporte les schémas

La première nouveauté concerne la structuration des documents XML. Jusqu'à présent, il fallait utiliser une DTD (Document Type Definition), avec toutes les limitations liées à ce format de méta-données. Il est désormais possible de mettre en œuvre des schémas compatibles avec la spécification XML Schema. Il subsiste néanmoins une restriction partielle sur le support de XML Schema. En effet, les fonctions de validation de documents XML utilisent encore l'ancien format XDR, au lieu du nouveau standard XSD, homologué par le W3C. Au vu du support de XML dans le reste de la plate-forme, ce point devrait être rapidement réglé.

Le processus de validation d'un document XML à partir d'un schéma est très simple. Il suffit de positionner un indicateur sur la classe XMLTextReader utilisée pour la lecture. Les modes de validation possibles sont : pas de validation, DTD et XML Schema. Il est nécessaire d'enregistrer un callback pour recevoir les erreurs de validation. Les erreurs de construction sont, elles, toujours remontées.

```
Bool validation_result = true;
XmlTextReader myReader = new
XmlTextReader("c:\\test.xml");
myReader.Validation = Validation.Schema;
myReader.ValidationEventHandler += new
ValidationEventHandler(this.ValidationEventHandle);
while (myReader.Read()){}

public void ValidationEventHandle (object sender,
ValidationEventArgs args)
{   validation_result = false;
    Console.Write("Document invalide " + args.Message);
}
```

Présentation de XML Schema

XML Schema est une extension officielle d'XML qui fait l'objet d'un processus de normalisation par le W3C. Son objet est la structuration des documents XML. A ce titre, elle est vouée à remplacer l'actuel système des DTD.

XML Schema permet la prise en charge des types simples et structurés dans des documents XML. Les types simples sont pris en charge grâce au namespace XSD (XML Schema Definition). Namespace doit être ici pris au sens XML du terme. XSD définit les types de bases tels que : entier, nombre à virgule flottante, date, chaîne, booléen …

Avec XML schema, il est désormais possible de typer les attributs d'un document, ce qui évite d'avoir ensuite à les contrôler. Voici un exemple de schéma XML correspondant à celui d'une classe C#. Ce schéma XML a été généré à partir d'une classe C#, par l'utilitaire xsd.exe livré avec le .NET Framework SDK.

```
<?xml version="1.0" ?>
<schema targetNamespace=""
        attributeFormDefault="qualified"
        elementFormDefault="qualified"
        xmlns="http://www.w3.org/1999/
XMLSchema">
    <element name="Client"
        xmlns:q1="" type="q1:Client"
        nullable="true" />
    <complexType name="Client">
        <all>
            <element name="Prenom"
                xmlns:q2="http://www.w3.org/
1999/XMLSchema"
                type="q2:string" nullable="true" />
            <element name="Nom"
                xmlns:q3="http://www.w3.org/
1999/XMLSchema"
                type="q3:string" nullable="true" />
            <element name="Solde"
                xmlns:q5="http://www.w3.org/
1999/XMLSchema"
                type="q5:decimal" />
        </all>
    </complexType>
</schema>
```

L'exécution des programmes développés pour .NET est prise en charge par le CLR (Common Langage Runtime). La présence du CLR a permis de standardiser les types de données, et ce quel que soit le langage utilisé pour écrire le programme. En fait, une String en VB.NET et une String en Cobol .NET sont toutes les deux des String .NET

Cette standardisation des types permet aux langages .NET de sérialiser des objets. De plus cette sérialisation est automatiquement prise en charge par le Framework.

Pour sérialiser un objet, il faut néanmoins respecter deux conditions :

- La classe de l'objet doit posséder un constructeur sans argument.
- Toutes les variables membres doivent être sérialisables.

Trois modes de sérialisations des objets sont disponibles. Ils correspondent à des besoins distincts. Les exemples qui suivent sérialisent/désérialisent des instances d'une classe appelée Client. En voici la définition :

```
namespace netTest
{
   using System;

  public class Client
  {
        public string nom =“”;
        public string prenom =“”;
        public decimal solde = 0.0;

      public Client()
      {
      }

    public Client(string aNom, string aPrenom, decimal
aSolde)
        {
            this.nom = aNom;
            this.prenom = aPrenom;
            this.solde = aSolde;
        }
  }
}
```

La sérialisation binaire

C'est le mode le plus courant. Il assure la persistance des objets sous forme binaire. Ce mode est de loin le plus compact. Il s'utilise au travers de la classe BinaryFormatter contenu dans le namespace System.Runtime.Serialization. Formatters.Binary.

Exemple de sérialisation/désérialisation en mode binaire :

```
//Sérialisation binaire
Stream s = (new
File(“myClient.txt”)).Open(FileMode.Create);
BinaryFormatter bf = new BinaryFormatter();
bf.Serialize(s ,new Client(“Dupond”,”Jean”,1000.0));
s.Close();

//Désérialisation binaire
Stream s = (new
File(«myClient.txt»)).Open(FileMode.Open);
BinaryFormatter bf = new BinaryFormatter();
Client c = (Client)bf.Deserialize(s);
s.Close();
```

La sérialisation SOAP

C'est le mode qui est utilisé pour transmettre des objets à un service Web. Il s'utilise à travers la classe SOAPFormatter contenu dans le namespace system.Runtime.Serialization. Formatters.Soap.

Exemple de sérialisation/désérialisation en mode SOAP :

```
//Sérialisation SOAP
Stream s = (new
        File(“myClient.txt”)).Open(FileMode.Create);
SoapFormatter sf = new SoapFormatter();
sf.Serialize(s ,new Customer(“Dupond”,”Jean”,1000.0));
s.Close();
//Désérialisation SOAP
Stream s = (new
            File(«soapmode.txt»)).Open(FileMode.Open);
SoapFormatter sf = new SoapFormatter();
Customer c = (Customer) sf.Deserialize(s);
s.Close();
```

La sérialisation XML

Ce mode présente l'intérêt de sérialiser les objets sous forme de documents XML. Cela permet de les désérialiser facilement depuis d'autres plate-formes que .NET. Par ailleurs, ce mode est conforme au standard XSD de XML Schema. Attention cependant, il est particulièrement verbeux. Il s'utilise à partir de la classe XMLSerializer contenue dans le namespace System.Xml. Serialization.

Exemple de sérialisation/désérialisation en mode XML :

```
//Sérialisation XML
Stream s = (new
File(“myClient.txt”)).Open(FileMode.Create);
XmlSerializer xs = new XmlSerializer(typeof(Customer));
xs.Serialize(s ,new Client(“Dupond”,”Jean”,1000.0));
s.Close();

//Déserialisation XML
Stream s = (new
File(«myClient.txt»)).Open(FileMode.Open);
XmlSerializer xs = new XmlSerializer(typeof(Customer));
Client c = (Client)xs.Deserialize(s);
s.Close();
```

Deux nouvelles classes pour lire et écrire des documents XML

La manipulation de contenu XML dans le .NET Framework repose principalement sur deux classes abstraites. L'une permet de parcourir du contenu XML et l'autre d'en produire. Ces classes permettent de traiter le contenu XML sous forme de flux. Cette gestion par flux diminue fortement les problèmes de consommation mémoire que l'on pouvait rencontrer avec le DOM sur des documents volumineux.

De plus, que cela soit en SAX ou en DOM, le parcours d'un document XML est mal adapté au développement objet. C'est pourquoi, en plus de l'implémentation des standards SAX et DOM, .NET fourni de nouveaux mécanismes d'accès plus intuitifs.

SAX est une technologie de type **push**. L'utilisateur enregistre des callbacks qui sont appelées par le parseur durant la lecture du document. Microsoft propose également un mécanisme de type **pull** : c'est l'utilisateur qui va chercher l'information. Il fonctionne à partir de la classe abstraite XMLReader qui détient un curseur sur le document XML en cours de lecture. Le curseur circule à sens unique, il n'est pas possible de revenir en arrière. Cette petite restriction permet des implémentations très performantes.

Il existe trois implémentations de XMLReader :

- XMLTextReader qui permet la lecture depuis un flux textuel,
- XMLNodeReader, dédié à la lecture depuis une arborescence d'objets de type DOM,
- XSLReader, dédié à la lecture de feuille de style XSL-T.

Exemple de lecture d'un document XML avec XMLTextReader :

```
XmlTextReader myReader = new
XmlTextReader(“c:\\fournisseur.xml“);
while (myReader.Read())
{
    switch (myReader.NodeType)
   {
        case XmlNodeType.Element:
            if (myReader.HasAttributes)
            {
            //Traitement des attributs
            }
      break;
        case XmlNodeType.Text:
            //Lecture du texte
        break;
   }
}
```

Il existe une mécanique similaire pour l'écriture de documents XML. Il est basé sur la classe abstraite XMLWriter qui détient un curseur à sens unique. XMLWriter possède des méthodes qui permettent d'ajouter des éléments, du texte ou des attributs à la position désignée par le curseur. Il en existe deux implémentations :

- XMLTextWriter, pour la génération de documents textuels,
- XMLNodeWriter qui permet d'écrire dans une arborescence de type DOM.

Exemple d'écriture d'un document XML avec XMLTextWriter :

```
XmlTextWriter writer=new XmlTextWriter(“doc.xml“,null);

writer.Formatting = Formatting.Indented;
```

```
writer.WriteStartDocument();
writer.WriteComment("XML et .NET !");
writer.WriteStartElement("racine");
   writer.WriteStartElement("element",null);
      writer.WriteAttribute("attr_String","valeur1");
      writer.WriteAttrInt64("attr_int64",12345678);
   writer.WriteEndElement();
writer.WriteEndElement();
writer.WriteEndDocument();

writer.Flush();
writer.Close();
```

Résultat des appels à XMLWriter :

```
<?xml version="1.0" ?>
<!— XML et .NET ! —>
<racine>
   <element attr_String="valeur1"
            attr_int64="12345678"  />
</racine>
```

De plus, il est maintenant possible de lire et d'écrire des données de type entier, date ou réel dans des champs textes ou attributs, et ceci, directement depuis les classes XMLReader et XMLWriter. Cela allège la charge des développeurs qui n'ont plus besoin de contrôler dynamiquement le typage.

Les requêtes XPath avec .NET

Autre nouveauté importante, le support du langage XPath (voir encadré). XPath fait l'objet d'une API dédiée dans le namespace system.xml.xpath. XPath peut-être utilisé de deux manières différentes :

- via une implémentation de la classe abstraite XMLNavigator, qui permet d'appliquer des expressions XPath à une arborescence XML,
- grâce au transformateur XSL-T intégré.

L'utilisation d'XPath se fait par l'intermédiaire de la classe DocumentNavigator. Cette classe travaille sur un objet de type XmlDocument. Ses méthodes, telles que Select et SelectSingle, permettent de faire des requêtes XPath sur le document. On dispose ensuite de méthodes telles que MoveTo, qui offrent les fonctionnalités de navigation sur l'ensemble résultat.

Exemple de requête XPath à résultat de type ensemble :

```
XmlDocument xdoc = new XmlDocument();
xdoc.Load(«doc.xml»);
XmlNavigator nav = new DocumentNavigator(xdoc);
nav.Select("//Racine/Element");

while (nav.MoveToNextSelected())
{
  //Traitements divers
}
```

Notons également qu'il est possible de compiler les expressions XPath (méthode **compile**) pour améliorer sensiblement les performances lors de l'exécution. Enfin, la méthode **evaluate** permet l'exécution de requête XPath qui renvoient des résultats de type scalaire (et non des ensemble de nœuds).

Exemple de requête XPath à résultat de type scalaire :

```
XmlDocument xdoc = new XmlDocument();
xdoc.Load(«doc.xml»);
XmlNavigator nav = new DocumentNavigator(xdoc);

String result = nav.Evaluate("sum(//Produits/Prix)");
```

La méthode evaluate permet d'écrire, en une ligne, des commandes qui, avec des outils comme DOM, nécessiteraient plusieurs dizaines voire centaines de lignes de code.

Le langage Xpath

XPath est un langage dont l'objet est la localisation de ressources dans les documents XML. Il permet la récupération d'éléments ou d'ensembles d'éléments qui composent un document XML. XPath est à XML ce que les paths sont aux systèmes de fichiers traditionnels. Il fait l'objet d'une spécification du W3C.

Regardons de plus près à quoi ressemblent les expressions XPath aussi appelées Location Paths :

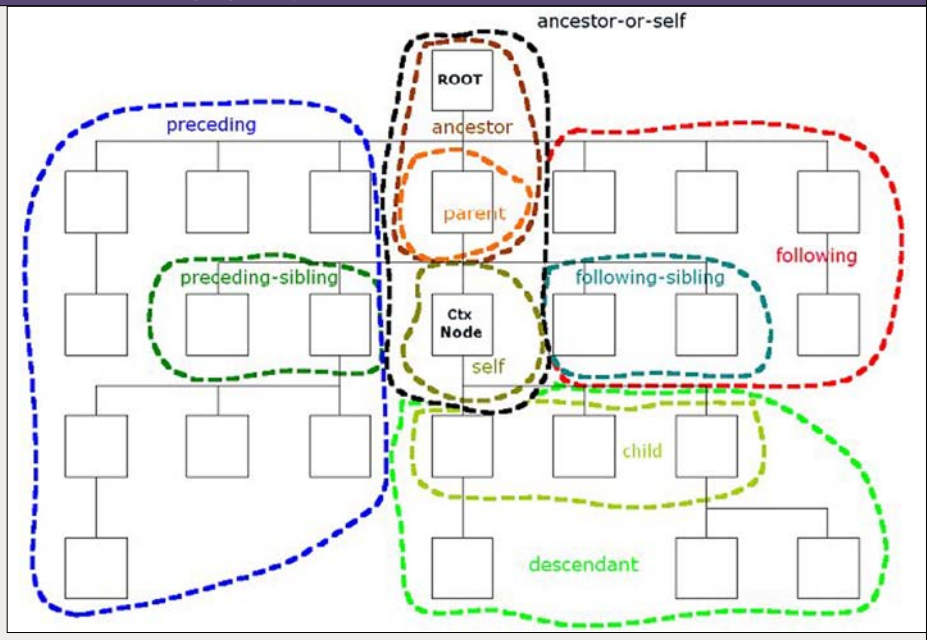
```
child::*
child::*/child::para
/descendant::olist/child::item
/child::para[attribute::type="toto"]
child::*[self::chapter or self::appendix]
```

Ces expressions peuvent être relatives à un nœud (context node) de l'arborescence XML (location path relatif) ou bien relatives à la racine du document XML (location path absolu). Les location paths absolus sont aisément repérables car ils commencent par le caractère /, ce qui n'est pas sans rappeler la notation des paths sous UNIX.

Les location paths sont constitués d'un ou plusieurs Location Step séparés par le caractère /. Chaque location step comprend trois parties distinctes, l'axe, le test node et des prédicats.

L'Axe précise la nature des relations qui lient les différents nœuds d'un ensemble résultat. Il oriente le Location Step en lui donnant un sens de parcours dans l'arborescence du document XML. Les axes disponibles sont :
child, preceding, descendant, attribute, parent, namespace ancestor, self, following-sibling, descendant-or-self preceding-sibling, ancestor-or-self, following

Le Test Node précise le type des éléments que l'on souhaite inclure. Il existe deux possibilités :



Représentation visuelle des axes XPath sur un document XML

- * pour remonter tous les éléments,
- **nom-élément** pour remonter uniquement les éléments de types nom-élément.

Si l'on souhaite remonter plusieurs types d'éléments il faut utiliser * et ajouter des prédicats.

Les prédicats sont des fonctions de filtrage qui s'appliquent à un ensemble de nœuds. Ils s'écrivent entre crochets et sont facultatifs. Voici quelques exemples de prédicats :

```
[child::title]
[position()=1]
[attribute::type="warning"]
[self ::chapter or self::appendix][position()=last()]
```

XPath dispose également d'une bibliothèque de fonctions intégrées qui facilitent l'écriture des location paths. On y trouve des fonctions de manipulation de chaînes et de nombres, ainsi que les opérateurs arithmétiques et booléens habituels. Il existe aussi une notation abrégée qui permet de réduire sensiblement la taille des location paths.

Le langage XSL-T

XSL-T est un langage dont l'objet est la transformation de documents XML d'un format vers un autre. Le format d'arrivée n'est pas obligatoirement du XML. Contrairement à XPath, XSL-T est un langage intégralement écrit en XML.

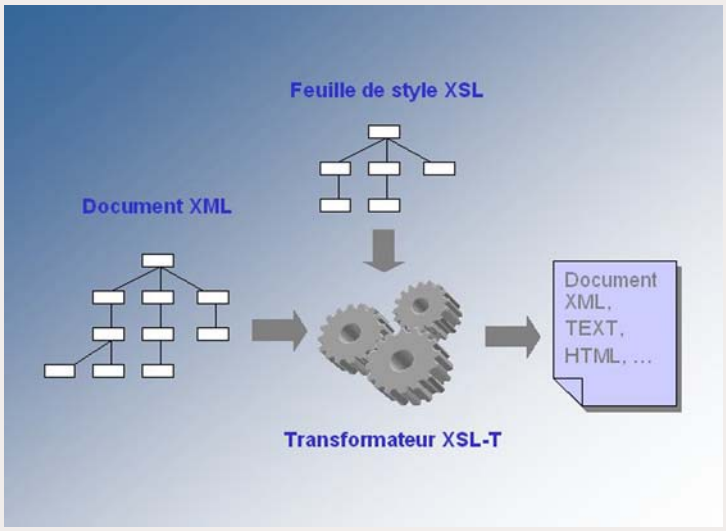
Les deux principales applications d'XSL-T sont la génération de contenu (généralement du HTML) et les transformations de données d'un format vers un autre, les fameuses moulinettes. Regardons de plus près un exemple de transformation XSL-T.

Voici le document source (client.xml) :

```
<?xml version=»1.0" ?>
<listclient>
    <client Prenom=»Jean» Nom=»Dupond»
        Pays=»France»/>
    <client Prenom=»Pierre» Nom=»Durand»
        Pays=»Suisse» />
</ listclient>
```

Voilà maintenant la feuille de style (xmltrans.xsl) :

```
<?xml version=»1.0"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
  <xsl:output indent="no" encoding="ISO-8859-1" />
  <xsl:template match="listclient">
    <customerlist><xsl:apply-templates />
    </customerlist>
    </xsl:template>
  <xsl:template match="listclient/client">
    <customer>
        <firstname>
            <xsl:value-of select="@Prenom" />
        </firstname>
        <lastname>
            <xsl:value-of select="@Nom" />
        </lastname>
        <country>
            <xsl:value-of select="@Pays" />
        </country>
    </customer>
  </xsl:template>
</xsl:stylesheet>
```



Principe de fonctionnement d'un transformateur XSL-T

Et le résultat de l'application de cette feuille de style XSL-T sur le document XML (result.xml) :

```
<?xml version=»1.0" encoding=»ISO-8859-1" ?>
<customerlist>
    <customer>
        <firstname>Jean</firstname>
        <lastname>Dupond</lastname>
        <country>France</country>
    </customer>
    <customer>
        <firstname>Pierre</firstname>
        <lastname>Durand</lastname>
        <country>Suisse</country>
    </customer>
</customerlist>
```

Le support de XSL-T dans .NET

L'implémentation du langage de transformation XSL-T (Voir encadré) dans les bibliothèques du CLR se trouve dans le namespace System.Xml.Xsl. La classe à utiliser est XslTransform.

Exemple de transformation XSL-T avec XslTransform :

```
...

XmlDocument mydata = new XmlDocument();
mydata.Load(«clients.xml»);
XmlWriter writer = new XmlTextWriter(Console.Out);
XslTransform xslt = new XslTransform();
xslt.Load("xmltohtml.xsl");
xslt.Transform(new
DocumentNavigator(mydata),null,writer);

...
```

Le .NET Framework fournit bien plus qu'un support de l'XML. Il place XML au centre de toute représentation de données en permettant par exemple de sérialiser tout objet sous une forme XML. Cette sérialisation facilite le dialogue entre les systèmes même en cas d'hétérogénéité.

Le .NET Framework est également constitué par ADO.NET. Cet ensemble de composants remplace ADO et se charge de l'accès aux bases de données dans .NET. Là encore XML est présent. En effet ADO .Net fournit une nouvelle classe très intéressante nommée *dataset***.** Cette classe permet non seulement d'extraire des enregistrements d'une base de données, mais également de sérialiser les données et les méta-données sous forme XML.

Les classes fournies par .NET facilitent la manipulation de contenu XML et rendent son utilisation plus confortable et plus rapide. On peut ici rapprocher l'approche de Microsoft de celle de JDOM dans le monde Java.

Du point de vue des performances, MSXML a toujours été parmi les outils les plus véloces. L'arrivée de la version 4.0 apporte encore d'importantes améliorations, en particulier dans le transformateur XLS-T.

Dans la perspective d'utilisation des services Web, la qualité du support d'XML et de ses compléments (XSL/T, XSD, XQUERY, XPATH, SOAP, …) est un facteur déterminant dans le choix d'une plate forme de développement. Assurément, Microsoft l'a bien compris ! ●

ALLER PLUS LOIN

Le portail officiel .Net de Microsoft:
http://msdn.microsoft.com/net/

Le lexique XML :
http://www.neoxia.com/fr/lexxml.php3

Classes et objets dans Visual Basic .NET

Les notions de classe et d'objet existaient déjà dans les versions précédentes de Visual Basic, mais c'est la version 7 du langage, qui fait partie de l'infrastructure .NET, qui lui donne ses lettres de noblesse. Cet article présente les principales caractéristiques des classes et montre un exemple de mise en œuvre de l'héritage.

Gérard Frantz
est expert en systèmes et développement Windows depuis 1984. Auteur de nombreux livres de programmation pour Windows, il contribue à des magazines informatiques en France et aux USA.
Gerard@GerardFrantz.com

Dans Visual Basic. NET, tout est objet, donc classe puisqu'un objet est l'occurrence d'une classe. Cela ne se limite d'ailleurs pas au seul langage Visual Basic. Les classes développées avec un des langages supportés par .NET (C#, C++ ou JavaScript en standard, ou un des nombreux autres langages en cours de développement par d'autres éditeurs) peuvent travailler ensemble. On peut ainsi créer un objet dans un langage à partir d'une classe développée dans un autre langage, ou encore définir une nouvelle classe en Visual Basic qui hérite d'une classe définie en C#, par exemple. Mais commençons par examiner comment on peut définir une classe et créer des objets avec Visual Basic.

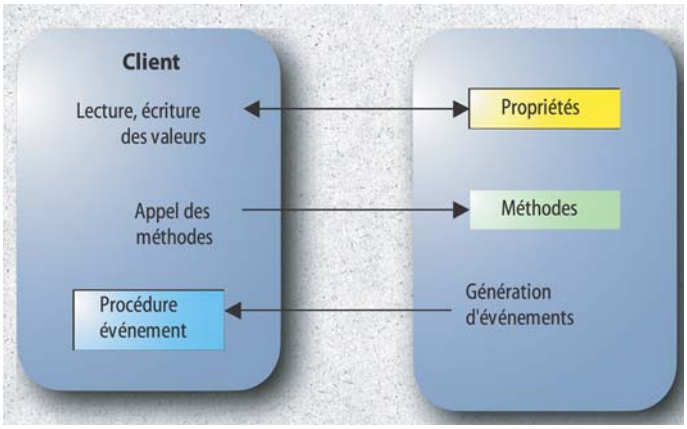


Figure 1. L'interface d'une classe.

Une classe est définie dans un module source d'extension vb. Plus de fichiers d'extension frm, bas ou cls avec cette nouvelle version : vous pouvez placer le type de code que vous voulez dans n'importe quel fichier vb. La définition d'une classe est délimitée par un bloc Class… End Class. Une classe appelée Compte (pour la gestion d'un compte bancaire) est ainsi définie :

```
Class Compte
   ' Code de la classe
End Class
```

Pour créer un objet à partir de cette classe, il suffit d'écrire ailleurs dans le code :

```
Dim cpt As New Compte()
```

Ou encore (les deux écritures sont équivalentes, notez que le Set des versions précédentes a disparu) :

```
Dim cpt As Compte
cpt = New Compte()
```

Propriétés et méthodes

Une classe n'est utile qu'à travers son interface, qui comprend l'ensemble des propriétés et des méthodes que l'on peut appliquer aux objets de la classe, ainsi que les événements que ces objets sont susceptibles de générer (figure 1). Les propriétés sont en quelque sorte les attributs des objets de la classe. On peut définir une propriété, soit à l'aide d'une variable non privée (généralement,

une variable déclarée Public), soit avec une procédure Property. Cette dernière méthode est préférable, car elle autorise l'écriture de code qui sera exécuté quand la valeur d'une propriété sera lue ou modifiée.

La propriété Solde de notre classe Compte peut être définie ainsi :

```
Class Compte
   ' Variable privée pour la propriété
   Private mSolde As Decimal
   ' Propriété Solde
   Property Solde() As Decimal
      ' Ecriture de la propriété
      Set ' N'autorise qu'un solde non négatif
         If Value >= 0 Then mSolde = Value
      End Set
      ' Lecture de la propriété
      Get
         Return mSolde
      End Get
   End Property
End Class
```

La définition d'une propriété se fait entièrement à l'intérieur d'un bloc Property, qui comprend des sous-blocs Set (pour la modification de la valeur) et Get (pour la lecture). La variable Value est automatiquement définie à l'intérieur du bloc Set et contient la nouvelle valeur affectée à la propriété. À une propriété est généralement associée une variable privée, appelée ici mSolde, pour le stockage de la valeur.

Une propriété peut être en lecture seule (ou plus rarement en écriture seule). Il suffit, pour cela, de ne définir qu'un bloc Get et d'ajouter le mot-clé ReadOnly. On peut ainsi rendre la propriété Solde précédente en lecture seule :

```
Class Compte
   ' Variable privée pour la propriété
   Private mSolde As Decimal
   ' Propriété Solde
   ReadOnly Property Solde() As Decimal
      ' Lecture de la propriété
      Get
         Return mSolde
      End Get
   End Property
End Class
```

Une méthode permet à l'utilisateur d'un objet de lui demander d'effectuer une action. Chaque procédure Sub ou Function non privée de la classe est vue comme une méthode. On peut ainsi ajouter la méthode Créditer à la classe précédente :

```
Public Sub Créditer(ByVal Montant As Decimal)
   mSolde += Montant
End Sub
```

Surcharge, constructeur et destructeur

Il est parfois utile de pouvoir appeler une même méthode, mais avec des paramètres différents. Les versions précédentes de Visual Basic avaient recours aux paramètres optionnels pour répondre partiellement à ce besoin. La nouvelle version autorise la surcharge des méthodes qui va beaucoup plus loin. La surcharge consiste en une nouvelle définition de la méthode qui ne diffère de l'originale que par le nombre ou le type des paramètres. Lors de l'appel de la méthode par un client, Visual Basic détermine l'implémentation à appeler selon la façon dont l'appel est effectué, en comparant le nombre et le type des paramètres.

Pour indiquer qu'une méthode est surchargée, il faut ajouter le mot-clé Overloads devant toutes les définitions de la méthode. On peut, par exemple, définir une nouvelle méthode Créditer qui prendrait un paramètre sous la forme d'une chaîne de caractères :

```
' Méthode Créditer (Decimal)
Public Overloads Sub Créditer(ByVal Montant As Decimal)
   mSolde += Montant
End Sub

' Méthode Créditer (String)
Public Overloads Sub Créditer(ByVal Montant As String)
   mSolde += Montant.ToDecimal
End Sub
```

Une classe peut aussi disposer de deux méthodes particulières appelées constructeur et destructeur. Le constructeur est automatiquement appelé chaque fois qu'un nouvel objet de la classe est créé. Un constructeur est une procédure Sub appelée New, par exemple :

```
Public Sub New()
   ' Solde initial
   mSolde = 0
End Sub
```

Un constructeur ressemble à une méthode, mais ne peut pas être appelé directement par le code. En revanche, comme toute méthode, un constructeur peut être surchargé. On peut ainsi définir un constructeur qui affecte une valeur initiale non nulle au solde (la bêta 1 de .NET imposait l'ajout de Overloads à tous les constructeurs, cela a disparu dans la bêta 2) :

```
Public Sub New(ByVal SoldeInitial As Decimal)
   mSolde = SoldeInitial
End Sub
```

L'appel du bon constructeur sera fait automatiquement lors de la création d'un objet :

```
' Appelle le constructeur par défaut
Dim cpt1 As New Compte()

' Appelle le constructeur avec paramètre
Dim cpt2 As New Compte(1000)
```

La destruction d'un objet peut se faire de façon implicite, quand la variable qui y fait référence est détruite, c'est-à-dire quand on sort du bloc où elle est définie. On peut aussi détruire explicitement un objet en affectant la pseudo-valeur Nothing à la variable. Dans tous les cas, la destruction effective n'est réalisée qu'à l'initiative de .NET, dans un processus particulier de récupération de la mémoire. L'application peut bien définir une méthode appelée Destruct, le destructeur, mais le moment où celui-ci sera appelé est aléatoire, il peut même ne jamais être appelé si, par exemple, l'application est terminée.

Les événements

J'ai indiqué que l'interface d'une classe comprend l'ensemble de ses propriétés, méthodes et événements. La mise en œuvre des événements est un peu plus complexe que celle des propriétés et des méthodes. Un objet peut générer un événement, qui peut être intercepté par le client de l'objet. On dit parfois qu'un événement est une méthode sortante : l'objet appelle son client. Mais cela est plus complexe dans la mesure où, si le client maîtrise l'objet qu'il crée, l'objet, lui, ne connaît pas son client a priori.

Pour qu'un objet puisse générer un événement, il faut tout d'abord le déclarer avec l'instruction Event. Par exemple :

```
Event Débiteur(ByVal Solde As Decimal)
```

La génération de l'événement se fait dans le code de la classe à l'aide de l'instruction Raise suivie du nom de l'événement et des valeurs des paramètres déclarés. On peut, par exemple, modifier la méthode Créditer afin qu'elle génère l'événement Débiteur si le solde du compte devient négatif :

```
Public Sub Créditer(ByVal Montant As Decimal)
    mSolde += Montant
    If mSolde < 0 Then
     RaiseEvent Débiteur(mSolde)
  End If
End Sub
```

Si la déclaration et la génération d'un événement se font dans la classe, le client doit effectuer certaines actions pour intercepter l'événement. Il existe en fait deux méthodes pour cela : le recours à la déclaration WithEvents, semblable à celle des versions précédentes, ou l'utilisation du nouveau AddHandler, plus dynamique et plus souple.

Une variable déclarée avec WithEvents représente un objet qui peut recevoir les événements de la classe correspondante. Un client de la classe Compte peut déclarer une variable de la façon suivante :

```
Dim WithEvents cpt As Compte
```

Cette déclaration impose quelques contraintes : elle doit être faite au niveau du module ou de la classe (pas dans une méthode) et ne peut être associée à l'emploi de New. Il faut alors créer l'objet de façon indépendante :

```
cpt = New Compte()
```

On peut dès lors intercepter les événements de la classe Compte, en définissant des procédures dont le nom est constitué du nom de l'objet et de celui de l'événement, séparés par un caractère souligné. Ces procédures seront appelées lors de la génération des événements correspondants :

```
Public Sub cpt_Débiteur(ByVal Solde As Decimal)
End Sub
```

L'utilisation de WithEvents est relativement figée, car la connexion entre une variable objet et le gestionnaire d'événements se fait au moment de la conception de l'application. De plus, si plusieurs variables de la même classe sont déclarées dans une application, on ne peut pas définir une méthode commune pour le traitement de chaque événement de la classe.

Le lien dynamique mis en place par AddHandler résout ces problèmes. Cette fois-ci, la variable objet est déclarée normalement, éventuellement avec New. Ensuite, AddHandler est appelé afin de relier un événement de l'objet à une procédure de traitement de l'événement :

```
Dim cpt As New Compte()
AddHandler cpt.Débiteur, AddressOf CompteDébiteur
```

Ici, CompteDébiteur est le nom d'une procédure chargée de traiter l'événement Débiteur, déclarée comme la procédure cpt_Débiteur précédente :

```
Public Sub CompteDébiteur(ByVal Solde As Decimal)
End Sub
```

Rien ne nous empêche alors de déclarer plusieurs variables de la même classe et de toutes les relier à la même procédure de traitement d'un événement, grâce à une suite d'appels à AddHandler.

L'héritage

Jusqu'à sa version 6, on pouvait dire que le langage Visual Basic était orienté objet, mais qu'il ne supportait pas la notion d'héritage, ce qui diminuait de beaucoup la portée de cette affirmation. On peut maintenant le clamer haut et fort : Visual Basic.NET est un véritable langage orienté objet, supportant pleinement les caractéristiques attendues de tels langages : l'encapsulation, le polymorphisme et l'héritage.

L'héritage permet de mettre en œuvre simplement la réutilisation de code. Pour utiliser l'héritage, il faut au moins deux classes : une classe dont on peut hériter, appelée classe de base et une classe qui hérite — ou dérive —de cette classe de base. Une classe dérivée peut elle-même être classe de base pour une autre classe dérivée, l'héritage peut donc se faire sur plusieurs niveaux. En revanche, une classe ne peut dériver directement que d'une seule classe. L'héritage multiple du C++ n'est pas possible dans l'infrastructure .NET.

Pour indiquer qu'une classe dérive d'une autre classe dans Visual Basic, il suffit d'ajouter une déclaration Inherits après la déclaration de la classe. La définition d'une classe CompteEpargne, spécialisation de la classe de base Compte, est donc la suivante :

```
Class CompteEpargne
  Inherits Compte

End Class
```

Si on se rappelle que le signe ":" sépare deux instructions sur la même ligne dans VB, on peut même écrire, pour se rapprocher de la syntaxe C++ :

```
Class CompteEpargne : Inherits Compte

End Class
```

Une des principales caractéristiques de l'héritage est la possibilité de redéfinir des méthodes de la classe de base dans la classe dérivée, ce qui permet de spécialiser la classe et de modifier le comportement de ses objets en conséquence. Pour qu'une méthode puisse être redéfinie dans une classe dérivée, il faut que la classe de base l'autorise, en indiquant que la méthode est Overridable.

Le listing 1 montre une version modifiée de la classe Compte, prête à être dérivée. La méthode Créditer y est déclarée Overridable et la variable mSolde est marquée Private, ce qui empêche à toute autre classe d'y accéder. On aurait pu la marquer Protected pour que les classes dérivées, et seulement elles, puisse y accéder. Dans cette classe, l'événement Débiteur dispose également d'un paramètre supplémentaire, Annulation, qui permet l'annulation d'une opération par le client.

Le listing 2 présente la classe dérivée CompteEpargne, l'héritage étant marqué par la ligne Inherits Compte. Cette classe définit un événement spécifique, appelé Débit. Les deux constructeurs (procédures New) sont redéfinis dans la classe dérivée, mais ils ne font qu'appeler les constructeurs de la classe de base, grâce au mot-clé MyBase qui la représente. La méthode Créditer est définie avec Overrides pour indiquer qu'il s'agit d'une redéfinition de la méthode homonyme de la classe de base. Selon le montant passé en paramètre, cette méthode appelle la méthode de la classe de base, ou bien effectue son propre traitement (ici, génère un événement). Enfin, la classe dérivée dispose d'une propriété spécifique, Interêts.

La classe dérivée dispose de l'interface de la classe de base, et y ajoute des éléments complémentaires (figure 2).

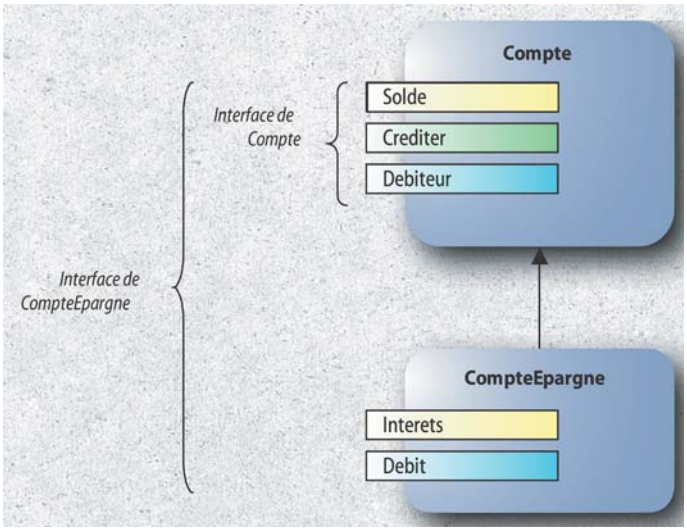


Figure 2. Interfaces de la classe de base et de la classe dérivée.

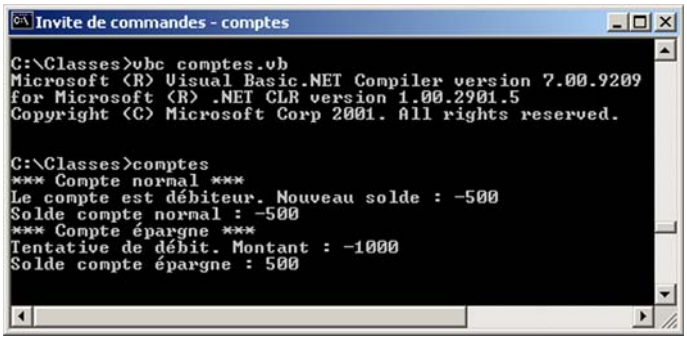


Figure 3. Compilation et etst des classes Compte et CompteEpargne.

Pour montrer l'utilisation des classes Compte et CompteEpargne, nous pouvons écrire un code qui crée un compte normal, le crédite d'un montant, puis le débite d'un autre montant et enfin affiche le nouveau solde, sur chacun des comptes (Figure 3). L'interception des événements est effectuée de façon dynamique. Les gestionnaires d'événements sont associés aux événements à l'aide de la fonction AddHandler, ce qui permet d'utiliser le même gestionnaire, CompteDébiteur pour les deux objets cptNormal et cptEpargne.

Le listing 3 montre le code du module Main qui réalise toutes ces actions. Le code des trois listings peut être placé dans un même fichier, par exemple Comptes.vb, qui peut être compilé dans une fenêtre de commande à l'aide de la commande, si vous avez installé le .NET Framework SDK :

```
vbc Comptes.vb
```

Cela produit un exécutable, Comptes.exe, dont l'exécution génère l'affichage de la figure 3.

Encore plus loin

La place manque ici por décrire dans le détail tout ce qui peut être fait à l'aide de classes et d'objets. Je vais cependant terminer en présentant les principales fonctionnalités disponibles.

Une classe de base peut être marquée abstraite à l'aide du mot-clé MustInherit, ce qui interdit de créer des objets directement à partir d'elle. Pour utiliser ce type de classe, il faut créer une nouvelle classe qui en dérive, et qui n'est pas abstraite. De plus, certaines méthodes peuvent être marquées MustOverride, ce qui impose que la méthode soit redéfinie dans une classe dérivée.

J'ai dit précédemment que l'héritage multiple de classes n'était pas admis dans l'infrastructure .NET. Une classe peut cependant implémenter plusieurs interfaces. Une interface ressemble à une classe, mais ne comprend pas de code d'implémentation, seulement la description de sont interface, c'est-à-dire les prototypes de ses propriétés, méthodes et événements. La déclaration d'une interface se fait dans un bloc

```
Interface NomInterface

End Interface
```

Une classe indique qu'elle implémente une interface à l'aide du mot-clé Implements. Elle doit alors effectivement comprendre du code d'implémentation des éléments de l'interface.

Une classe peut aussi comprendre des membres partagés, identifiés par le mot-clé Shared (Static dans d'autres langages). Un membre partagé peut être utilisé sans qu'il soit nécessaire d'instancier la classe, c'est-à-dire de créer d'objet à partir de la classe.

Faut-il utiliser l'héritage dans les applications Visual Basic ? Les mécanismes orientés objet et l'héritage de classes rendent évidemment le langage Visual Basic plus complexe. La conception et la mise en œuvre d'un ensemble de classes qui tire partie de l'héritage n'est pas une tâche triviale. Les programmeurs Visual Basic occasionnels peuvent continuer à utiliser le langage comme ils l'ont toujours fait, de façon simple pour créer rapidement des applications utiles. Mais les développeurs professionnels pourront exploiter les nouvelles possibilités du langage afin de créer des applications plus évolutives et plus faciles à maintenir, bénéficiant à la fois de la simplicité inhérente du langage Visual Basic, et des structures de programmation sophistiquées qui étaient préalablement réservées à des langages plus évolués comme le C++. ●

Listing 1 : La classe de base Compte

```
Class Compte
   ' Variable privée pour la propriété
   Private mSolde As Decimal

   ' Déclaration d'un événement
   Event Débiteur(ByVal Solde As Decimal, ByRef Annulation As Boolean)

   ' Constructeur par défaut
   Public Sub New()
      ' Solde initial
      mSolde = 0
   End Sub
   ' Constructeur avec un solde donné
   Public Sub New(ByVal SoldeInitial As Decimal)
      mSolde = SoldeInitial
   End Sub

   ' Propriété Solde
   Public ReadOnly Property Solde() As Decimal
      ' Lecture de la propriété
      Get
         Return mSolde
      End Get
   End Property

   ' Méthode Créditer
   Public Overridable Sub Créditer(ByVal Montant As Decimal)
      ' Nouveau solde
      Dim NouveauSolde As Decimal = mSolde + Montant
     ' Indicateur d'annulation
      Dim Annulation As Boolean = False
      ' Vérifie le nouveau solde
      If NouveauSolde < 0 Then
         ' Débiteur, avertit le client
         RaiseEvent Débiteur(NouveauSolde, Annulation)
      End If

      ' Si annulation, le mouvement n'est pas validé
      If Not Annulation Then
         mSolde = NouveauSolde
      End If
   End Sub
End Class
```

Listing 2 : La classe dérivée CompteEpargne

```
Class CompteEpargne
   Inherits Compte

   ' Evénement spécifique
   Event Débit(ByVal Montant As Decimal)

   ' Constructeurs
   Public Sub New()
      MyBase.New()
   End Sub
   Public Sub New(ByVal SoldeInitial As Decimal)
      MyBase.New(SoldeInitial)
   End Sub

   ' Méthode redéfinie Créditer
   Overrides Sub Créditer(ByVal Montant As Decimal)
      ' Ce compte interdit les débits (!)
      If Montant > 0 Then
         ' Appelle la méthode de la classe de base
         MyBase.Créditer(Montant)
      Else
         ' Génère un événement spécifique
         RaiseEvent Débit(montant)
      End If
   End Sub

   ' Propriété spécifique
   Public ReadOnly Property Interêts() As Decimal
      Get
         ' Calcul totalement fantaisiste !
         Interêts = Solde * 5 / 100
      End Get
   End Property
End Class
```

Listing 3 : Le module Main pour le test des classes

```
Sub Main()
   ' Compte normal
   Dim cptNormal As New Compte()
   ' Compte épargne
   Dim cptEpargne As New CompteEpargne()

   ' Gestionnaires d'événement pour les comptes
   AddHandler cptNormal.Débiteur, AddressOf CompteDébiteur
   AddHandler cptEpargne.Débiteur, AddressOf CompteDébiteur
   AddHandler cptEpargne.Débit, AddressOf Débit

   ' Compte normal
   With cptNormal
      System.Console.WriteLine("*** Compte normal ***")
      ' Ajoute 500
      .Créditer(500)
      ' Retire 1000
      .Créditer(-1000)
      ' Affiche le nouveau solde
      System.Console.WriteLine("Solde compte normal : " & .solde)
   End With

   ' Compte épargne
   With cptEpargne
      System.Console.WriteLine("*** Compte épargne ***")
      ' Ajoute 500
      .Créditer(500)
      ' Retire 1000
      .Créditer(-1000)
      ' Affiche le nouveau solde
      System.Console.WriteLine("Solde compte épargne : " & .solde)
   End With

   ' Attend une touche
   System.Console.Read()

   ' Déconnecte les gestionnaires d'événements
   RemoveHandler cptNormal.Débiteur, AddressOf CompteDébiteur
   RemoveHandler cptEpargne.Débiteur, AddressOf CompteDébiteur
   RemoveHandler cptEpargne.Débit, AddressOf Débit
End Sub

' Procédures de traitement des événements
Public Sub CompteDébiteur(ByVal Solde As Decimal, ByRef Annulation As Boolean)
   ' Affiche le nouveau solde
   System.Console.WriteLine("Le compte est débiteur. Nouveau solde : " & solde)
   ' Annule l'opération si le solde est inférieur à -1000
   Annulation = CBool(Solde < -1000)
End Sub

Public Sub Débit(ByVal Montant As Decimal)
   ' Affiche un message
   System.Console.WriteLine("Tentative de débit. Montant : " & Montant)
End Sub
```

RETROUVEZ LES ANCIENS NUMEROS DE LA LETTRE BIMENSUELLE DU DEVELOPPEMENT sur www.devreference.net

- Méthode : La conception dans l'Extreme Programming
- Dossier : Table ronde 'Les Services Web' | Des services Web avec J2EE | Des services Web avec Visual Basic
- Langage : Mapping objet relationnel avec Castor
- Interview : Anders Hejlsberg, créateur de langage
- Tribune : Développez jusqu'à 10 fois plus vite...

- Méthode : Le talon d'Achille du développeur
- Architecture : Les enjeux de la synchonisation
- Web : Zope, concepts et fonctionnalités
- Outil : WebGain Studio Professional 4.5

- Actualité : Du côté de chez Borland | Microsoft PDC 2001
- Méthode : Développement objet, best practices
- Programmation : Programmation par contrat et qualité logicielle
- Outil : Lotus Domino Rnext bêta 2

- Actualité : Borland JBuilder 6
- Méthode : Développement collaboratif, voie tracée ou utopie ?
- Architecture : EAI avec un hub de services applicatifs
- Algorithme : Compression de données BWT
- Web : Zope, prise en main
- Outil : WMware Workstation 3.0