



LE MAGAZINE DU DÉVELOPPEMENT

Programmez !

Mensuel • Avril 2003 • N°52 • 5,95 €

DEVELOPPER avec LES COMPOSANTS

ENQUÊTE OFFSHORE

Le développement à prix cassé

Sous-traitance en Inde, Russie :
les méthodes, outils, témoignages

PRATIQUE

- Démarrer PHP
- Programmation Windows avec C#
- ASP.NET : l'authentification par formulaire
- Démystifier J2EE avec Jboss

TESTS

- 4D : le cru 2003
- WINDEV 7.5, rival de Delphi ?

BELGIQUE 6,45 € - SUISSE 12 FS
LUXEMBOURG 6,45 € - CANADA 8,95 \$ CAN

M 04319 - 52 - F: 5,95 €

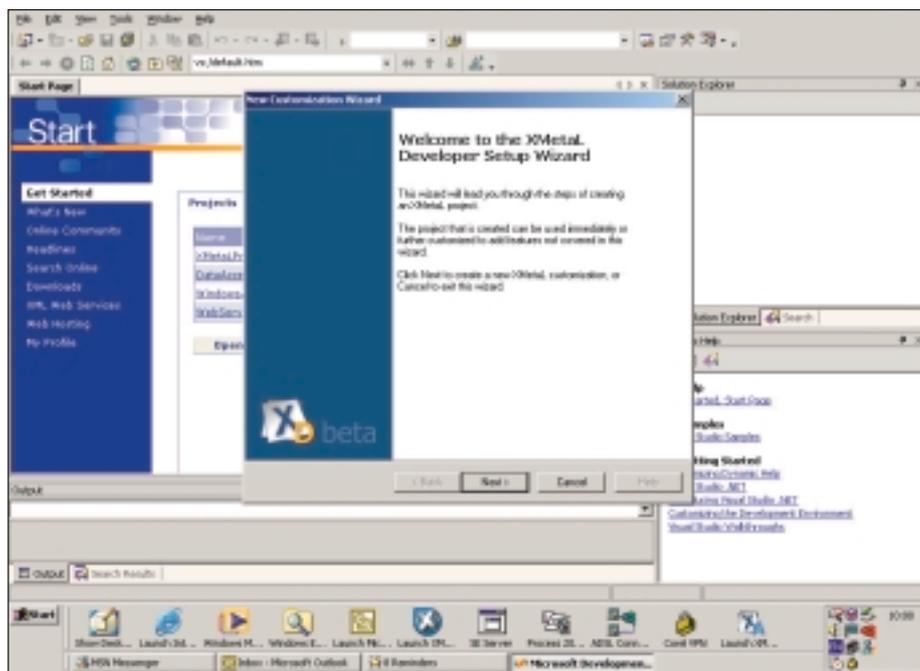


Printed in France - Imprimé en France

XMetaL 4 : plug in XML pour .NET

Destinée aux grands comptes, la suite XMetaL 4 se veut une plate-forme intégrée simplifiant le développement, la gestion et l'implémentation d'un environnement "XML Authoring". Mais pour ce faire, il est nécessaire au préalable de disposer d'une architecture Visual Studio .Net

Après 18 années d'existence, une implantation reconnue dans les métiers graphiques avec des produits tel que la suite CorelDraw, une entrée sur le marché de la gestion des processus d'entreprise (EPM ou Enterprise Performance Management) grâce au rachat de la société Micrografx, Corel élargit les possibilités de création de ses clients en lançant aujourd'hui XmetaL 4. Ce logiciel phare de création de contenus XML (Extensible Markup Language) a été développé à l'origine par Softquad. Composé des quatre modules Corel XMetaL Developer, Corel XMetaL Central, Corel XMetaL Author et Corel XMetaL for ActiveX, l'opus 4 de XMetaL permet l'utilisation d'un éditeur XML plus aisée, simplifie la gestion de l'XML. Son interface utilisateur facilite le développement de structures de contenus, la création de contenus ainsi que leur modification, leur réutilisation et donc leur maintenance. Le module Developer permet de créer les Schémas au standard W3C, les DTD (Document Type Declaration), les feuilles de styles (CSS), les fichiers XSLT (eXtensible Stylesheet Language Transform), les ressources, le code, etc. Les développeurs produisent la structure, l'organisation des documents et personnalisent également les interfaces utilisateurs ActiveX ou même Author à partir d'un code unique, avant de les mettre à disposition des différents collaborateurs d'une entreprise.



XMetaL Developer fonctionne en environnement Visual Studio .Net de Microsoft.

Réutiliser les composants

En permettant la réutilisation des composants, ce module a pour vocation de mettre en valeur les équipes de développement et de mieux gérer les compétences des différents groupes de travail quel que soit leur domaine d'application, VB, Java, Script, Com, Dom, etc. Ce module contient une boîte à outils (toolkit) accueillant des modules réutilisables, pré développés. Du point de vue ouverture, Corel XMetaL Developer supporte WebDAV, l'intégration des modules CMS (Content Management System) et dispose d'un assistant (wizard) d'importation de base de données. Corel XMetaL Central assure pour sa part la gestion de l'environnement XML. C'est à dire le déploiement, les mises à jour, la maintenance des applications XML au sein de l'entreprise et la suppression des informations relatives à l'environnement XML. Il sert de référentiel en gérant les DTD, Schémas, personnalisation, macros, modèles... A noter que la maintenance

des applications peut s'effectuer à distance en utilisant la technologie Soap (Simple Object Access Protocol) et les Web Services. Quant aux deux modules clients, ils assurent la relation avec l'utilisateur final. En fonction des connaissances en informatique, ce dernier disposera, via son navigateur Web, soit du client Author qui permet de modifier plus en profondeur le contenu et son interface (en ajoutant par exemple un correcteur orthographique, un thesaurus ou des marques de révisions des documents); soit au contraire d'un client léger de type ActiveX qui lui permettra juste de créer un document aux normes définies pas les développeurs.

Réduire les coûts

Avec cette suite, Corel cible principalement les marchés de la finance, les banques, les sociétés Hi-Tech, les administrations, les lycées et universités, les industries, les ISV... Le canadien espère réduire les coûts de maintenance et diminuer de manière drastique la charge de travail des équipes de développement. Toutefois, il ne faut pas oublier que XMetaL s'appuie sur l'infrastructure Visual Studio .Net de Microsoft. "XMetaL Developer peut être véritablement assimilé à un véritable plug-in pour la plate-forme .Net", précise Jean-Yves Fiou, responsable avant-vente chez Corel France. Ce qui ne sera peut être pas du goût de tout le monde !

Informations additionnelles

XMetaL 4 est le second produit lancé par Corel cette année, mais six sont encore à venir. A noter, les modules XMetaL 4 Central et Developer sont, et resteront dans la langue de Shakespeare. Seul le module Author est en français. Quant à ActiveX il est fonction des créations.

Tarif

Corel XMetaL Developer
999 Euros (599 Euros pour la mise à jour)

Corel XMetaL Author
456 Euros par licence

Corel XMetaL for ActiveX
456 Euros par licence

Serveur iBolt : Magic Software mise sur l'EAI

La solution EAI iBolt de Magic Software est destinée aux PME. L'éditeur se positionne ainsi sur un marché porteur, dans lequel il n'y a que peu de concurrence.



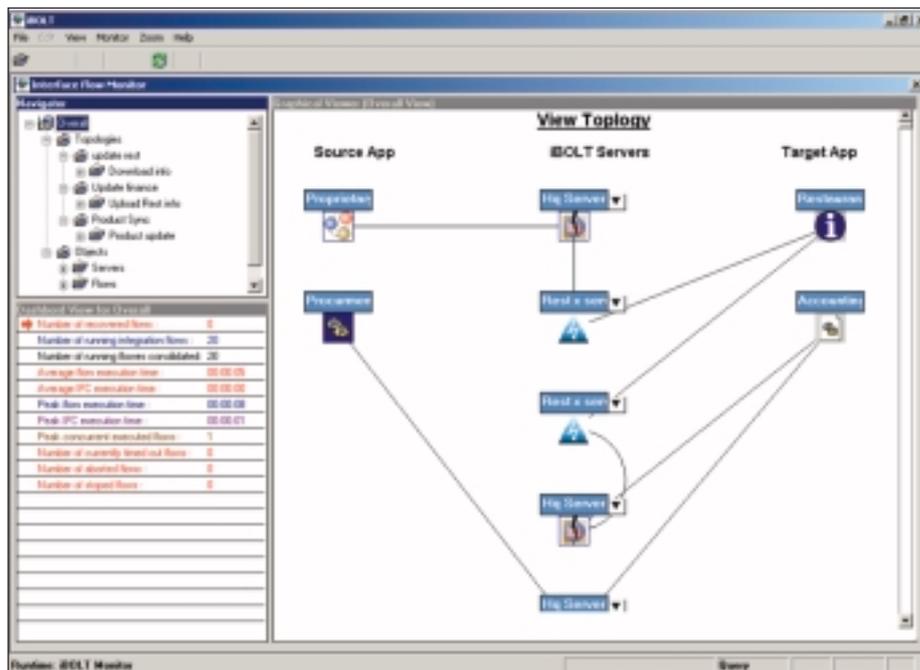
Avigdor Luttinger, responsable du programme iBolt chez Magic Software

Depuis sa création, Magic Software a du opérer deux revirements importants pour s'adapter au marché. Connue d'abord comme éditeur d'outils de développement, la société a du chercher depuis d'autres ouvertures. En

1999, le champ d'action a été étendu vers le commerce électronique et la gestion de la relation client. Aujourd'hui, Magic Software doit une nouvelle fois changer de cap, pour se diriger cette fois vers l'EAI (Enterprise Application Integration).

Un modèle économique viable

Les analystes prévoient en effet une croissance annuelle de 25 % dans ce secteur pour les prochaines années. Le marché de l'EAI devrait atteindre un volume de l'ordre de 25 milliards de dollars en 2005. De plus, l'EAI devrait être au centre des préoccupations des directeurs informatiques. Le quart d'entre eux la considère déjà comme étant critique pour l'entreprise et IDC prévoit que ce nombre va doubler d'ici deux ans. Pour Magic, il s'agit d'un marché à ne pas rater, d'autant que la société possède les compétences nécessaires. " Nous sommes nés adultes ! " déclare Avigdor Luttinger, responsable du programme iBolt, " l'EAI n'est pas



L'orientation des flux de données entre les différents composants du système d'information s'effectue à l'aide d'une interface graphique, le Flow Editor de iBolt.

un nouveau métier, pour nous ". En effet, il a suffit à Magic d'étendre simplement les fonctions de son moteur eDeveloper à l'EAI. Le serveur iBolt est donc l'aboutissement de la technologie et de l'expertise de Magic.

Une interface graphique pour diriger les flux

iBolt est basé sur eDeveloper, qui est le moteur de l'application. La dernière version en date est la 9.4, qui supporte les standards J2EE, .NET, XML, les Web Services et COM. Une application eDeveloper peut désormais fonctionner en client COM, en utilisant des ActiveX et des objets OLE, ou bien en serveur COM en fournissant des informations aux autres applications. eDeveloper 9.4 supporte

également Microsoft Messaging Queue, MQ Series et Java Messaging Server. Le développement de l'application EAI, c'est-à-dire l'orientation des différents flux, s'effectue à l'aide de Flow Editor. Il s'agit d'une interface graphique avec laquelle le développeur définit quels sont les flux d'informations qui vont transiter entre tous les composants du système. Le Flow Editor génère alors un repository de méta données en XML. L'exécution est ensuite effectuée par le composant Integration Flow Supervisor. C'est lui qui réalise l'orchestration dynamique des flux. Un message doit souvent traverser de multiples composants. Le Supervisor oriente donc ces messages en fonction des scripts générés par le Flow Editor, mais aussi en fonction des données contenues dans le message. Ce qui est le fin du fin dans la technologie EAI. Le Supervisor est déployé sur le iBolt Integration Server. Ce serveur peut fonctionner en multi tiers. Un serveur iBolt peut être placé sur plusieurs serveurs physiques, ce qui permet ainsi de partitionner les flux. Enfin, iBolt fonctionne sous différents OS (Windows, Unix, Linux, OS/400 ...) ainsi que sous de nombreuses bases de données (MS SQL, DB2, Oracle ...). ■

Alain COUPEL

Magic Software

Magic Software n'a rien d'une start up, puisqu'elle a été cofondée en 1984 par David Assia, qui en est l'actuel PDG. Cotée au NASDAQ (MGIC) depuis 1991, cette société israélienne a son siège social à Tel-Aviv. Elle emploie environ 700 personnes à travers le monde, réparties sur une cinquantaine de pays. Magic bénéficie d'un réseau important de plus de 2500 partenaires et le nombre de développeurs utilisant les outils Magic (les magiciens !) dépasse les 120 000. Enfin, 1,5 millions d'utilisateurs travaillent sur des applications Magic. Le chiffre d'affaire global se répartit à 54 % pour les licences, 36 % pour les services professionnels et 10 % pour les applications. Ce qui est une répartition standard pour ce type d'entreprise. Ce chiffre provient à 37 % de l'Amérique du nord, 33 % de l'Europe et de 27 % de l'Asie. Le panel de clients est très large puisque pratiquement tous les secteurs de l'économie y sont représentés.

4D version 2003 : haro sur les web services

L'éditeur français 4D vient juste de sortir sur le marché la toute nouvelle version de son SGBDR, 4D. Il s'agit de la première mise à jour payante depuis presque 2 ans. Cette cuvée 2003 apporte de très nombreuses corrections, améliorations et quelques nouveautés particulièrement intéressantes. La première grosse surprise de cette version est la possibilité de créer, de déployer et d'utiliser des Web Services. C'est sans doute l'évolution la plus marquante de cette version 2003. L'environnement supporte SOAP 1.1 et WSDL mais pas UDDI, ce qui peut restreindre quelque peu la mise en œuvre de cette fonction. Pour créer un service web, il suffit de choisir la méthode à convertir, de cocher deux options Web Services et c'est tout ! 4D traduit automatiquement le code 4D en XML. On peut ensuite directement, via 4D, publier son Web Service en activant le serveur Web intégré au SGBD. Cette implémentation ouvre de nouvelles perspectives intéressantes à 4D dans l'entreprise. Car, désormais, 4D pourra être utilisé en environnement hétérogène, via

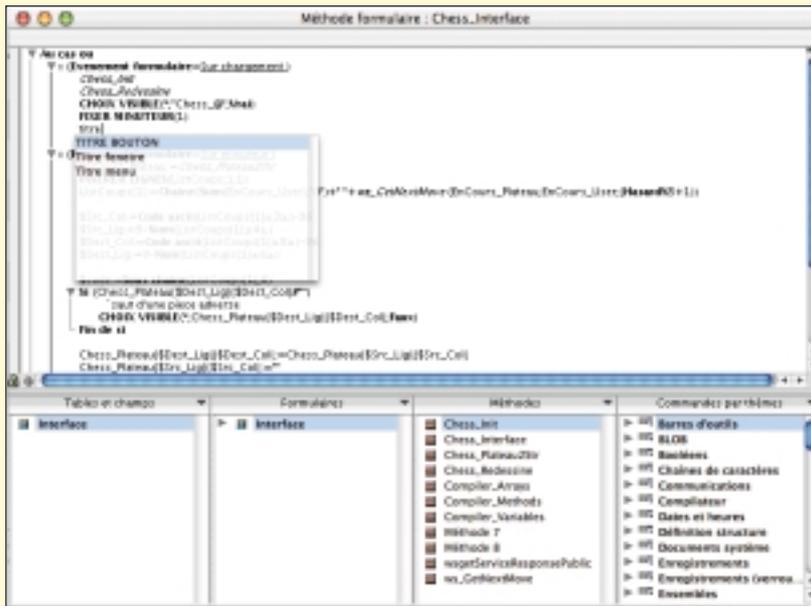
ronnement hétérogène (SAP, WebSphere, etc.). Les développeurs seront ravis d'apprendre qu'il y a désormais un compilateur intégré, permettant de compiler et de tester immédiatement son application (disponible sur certains packs

sables et sont des fichiers XML. Petit plus : on peut définir l'emplacement du curseur quand le code est inséré. Très pratique... De plus, chaque méthode peut maintenant contenir jusqu'à 32 000 lignes de codes. Et la fenêtre de l'éditeur

peut se diviser en une multitude de splits. Outre l'annulation multiple, notons une petite fonction bien sympathique : " commenter, décommenter ". On peut mettre en commentaire un bloc d'un clic, ou le décommenter. Actuellement 4D n'intègre pas de vrai CVS ou de fonction de versionning. 4D Insider propose une fonction d'historique (limité). Il faudra sans doute attendre la prochaine version, 4D nom de code Goldfinder. Une des volontés de 4D est de préserver le code existant. Cela signifie qu'un code provenant d'une base version 5.x ou 6.0 fonctionnera

(toutefois des fonctions / commandes auront peut-être été modifiées ou supprimées).

François Tonic



uniquement). Ce compilateur sert avant tout aux tests. Pour la built, il faut toujours passer par le générateur d'application. L'éditeur d'état a été largement amélioré, avec la notion de tableaux croisés dynamiques. De plus, le module se présentant sous la forme d'un plug-in, on peut directement intégrer l'éditeur dans une fenêtre de l'application. L'utilisateur aura alors la possibilité de créer son propre état. L'option HTML est maintenant disponible. L'éditeur de méthode subit aussi quelques modifications. Une des plus intéressantes est la possibilité d'utiliser la touche Tab quand on commence à toucher une commande, fonction... une liste est alors affichée et on peut choisir la bonne correspondance. Pour le code récurrent, les concepteurs ont introduit le concept de macros. On peut ainsi mettre en macros des procédures ou des fonctions revenant régulièrement. Les macros sont bien entendu personnal-

4D 2003 : premières impressions

À l'heure où nous écrivons ces lignes, nous avons en notre possession une des dernières versions bêtas. L'environnement est très fluide et réactif. On apprécie les améliorations de l'éditeur de code, notamment sur la structuration du code par bloc, que l'on rétracte et déploie. La fonction Commenter / Ne pas commenter se révèle très pratique pour les longs codes. On apprécie aussi la possibilité de personnaliser les colonnes de la partie basse de l'éditeur. La compilation avec le vérificateur de syntaxe est le bienvenu. Regret récurrent, l'impossibilité d'ouvrir plusieurs bases simultanément. La partie services Web semble bien implémentée et fonctionnelle.



des services web. L'inverse est vrai aussi. On peut intégrer et utiliser des Web Services aux applications 4D. Il devient enfin possible d'intégrer rapidement et facilement 4D dans un envi-

4D : hier, aujourd'hui, demain

Un éditeur français qui réussit à l'étranger !

Dans le paysage informatique français, peu d'éditeurs ont su se faire un nom international. Parmi les quelques élus, 4D, anciennement ACI, est sans doute une des plus belles réussites françaises dans l'édition de logiciels informatique. Ce fut pour Programmez ! une bonne occasion de rencontrer Laurent Ribardière.



Cela fait presque vingt ans que les noms 4D, 4e Dimension, ACI hantent l'univers Mac et depuis quelques années le monde Windows. Linux sera pris en compte en 2004. Aujourd'hui, 4D c'est environ 170 - 180 collaborateurs à travers le monde, via sept filiales en plus de la filiale française. La première installation à l'étranger a été réalisée aux Etats-Unis, et Apple y a joué un rôle important. C'est aussi le second marché de l'éditeur français. L'aventure débute en 1984. Laurent Ribardière crée alors ACI. Aujourd'hui, l'environnement 4D, c'est à



peu près 50 000 développeurs référencés. Selon l'éditeur, il y aurait environ douze mille développeurs actifs. Il est difficile de se faire une idée précise du nombre d'utilisateurs 4D. Il y a deux sortes de développeurs 4D : ceux qui suivent de près les évolutions et ceux qui développent avec une version et ne font rien durant des années, vu que la, ou les applications 4D tournent. Si 4D est un SGBDR, il peut servir de serveur d'applications, pour générer de vraies applications Mac et Windows.

Bientôt Linux

Concernant les grandes dates, nous avons voulu connaître les plus marquantes selon Laurent Ribardière : "Le client-serveur a été

un des grands changements. En 1993, nous avons sorti un 4D Client-Serveur. Je pense que c'était important. En 1995, le passage sous Windows ne s'est pas fait sans mal. Sur la percée Internet, nous avons pris un peu de retard. On a introduit un serveur http en 1997. Nous sommes un des seuls éditeurs à proposer un SGBD incorporant un serveur web. C'était un des premiers serveurs d'applications... Le portage sous MacOS X a mobilisé une partie de nos ressources. En 2004, Goldfinger va changer, je pense, bien des choses, par exemple l'ouverture vers de nouvelles plates-formes comme Linux. Ce sera une grande étape."

Depuis sa création le SGBD 4D a subi de multiples évolutions et changements. Laurent

Ribardière résume assez bien ceux ci : "La version 5 fut un événement majeur. La 5.5 devait initialement se nommer 6.0. Elle marquait l'apparition de la plate-forme Windows. La v6.0 a constitué la première véritable version fonctionnelle de l'édition Windows. Avec la v6.5, on a fourni un gros effort sur le support et la prise en compte du web. Pour la v2003, nous nous sommes demandés ce que désirait le développeur au quotidien ? et nous avons essayé d'en tenir compte. L'ergonomie développeur est très importante pour nous."

Web Services dans 4D

Avec la sortie de 4D 2003, l'éditeur propose une ouverture significative vers les Web Services. Cette implémentation permet à 4D de

Des évolutions importantes

v5.0 : importants changements dans 4e Dimension

v5.5 : première version Windows.

v6.0 : véritable première version Windows du SGBD totalement fonctionnelle

v2003 : prise en compte des Web Services

v2004 : de nouvelles améliorations, 4D v6.x continue à évoluer...

4D nom de code **Goldfinger** : refonte totale de 4e Dimension !

devenir une solution plus attrayante pour les entreprises. Le SGBDR pourra ainsi se fondre en toute transparence avec l'environnement informatique de l'entreprise et communiquer avec les autres applicatifs, via de simples Web Services. Laurent Ribardière explique la stratégie autour de SOAP et plus largement autour des services web : "SOAP est un changement majeur. On peut faire du remote procedure sans passer par RPC. Des développeurs utilisent depuis quelque temps SOAP dans 4D, grâce aux versions bêtas. SOAP permet d'ouvrir 4D. Il est possible, par exemple, par un Web Service de dialoguer avec SAP. SOAP est une fonction importante pour 4D. Et qui peut faire aussi bien du SOAP client que du SOAP serveur. Des Web Services sont publiés par SOAP et WSDL. Il aurait fallu intégrer UDDI immédiatement, c'est ce que l'on fera dans une future version. C'est nécessaire, UDDI est de facto un standard. Il y a encore des problèmes de référencement des Web Services. Soit une entreprise décrit son Web Services, soit la documentation est pauvre. Il reste encore une étape à franchir sur ce point-là. Le Web Service résout bien des problèmes. On peut aussi y limiter les accès aux données. Mais je pense que le service web va mettre un certain temps à se mettre en place. C'est un nouveau mode de raisonnement. Je crois que le Web Service va nous aider. Les gens nous reprochent parfois d'être un format propriétaire. L'intégra-



tion du service web va nous permettre d'avoir un format universel pour les données ! Concernant les standards, on ne peut pas adhérer à tous. Nous sommes une petite structure. On va suivre les principaux, il ne faut pas réinventer la roue à chaque fois !"

Comment faire évoluer un produit et suivre les développeurs ?

Tous les développeurs sont confrontés au problème des mises à jours. ACI puis 4D ont toujours eu le souci de préserver l'existant. Un des points forts de 4D est de pouvoir migrer en douceur de l'ancien code vers une version récente. Seules quelques méthodes peuvent être supprimées ou modifiées. Tout est fait pour faciliter les mises à jours. Sur l'évolution du produit en lui-même, le fondateur explique la démarche de la société : "Nous avons beaucoup de contacts avec nos clients. Chaque année, on organise dans différents pays des réunions avec les dévelop-

peurs. C'est un lieu d'échange et de récolte de demandes. On a aussi le site Bug Base, avec une rubrique spéciale "suggestions". Là, ce sont souvent de petites demandes. Des réunions sont aussi organisées avec de gros clients. Ils sont encore plus écoutés. Quelques mails nous arrivent directement. Il existe également des associations d'utilisateurs (notamment aux USA). De nouvelles fonctions viennent aussi de l'analyse du marché, comme ce fut le cas avec SOAP. On essaie de suivre les évolutions."

1 Code, des versions

Le plus intéressant avec 4D, c'est sa volonté de "coller" aux marchés nationaux. Si le code de 4D reste le même partout dans le monde, l'éditeur essaie de s'adapter à chaque marché. Ainsi, pour les Etats-Unis, 4D tente de répondre aux attentes américaines. Beaucoup de développeurs 4D aux USA ne savent pas que 4D est un produit français... 4D n'impose en rien "l'exception française". Comme souvent dans la plateforme Mac, 4D bénéficie d'une communauté active et souvent passionnée par le produit. C'est peut-être là une partie du succès de 4D. La passion des développeurs et l'écoute de ceux-ci par les ingénieurs. ■

Quelques dates à retenir

1984 : fondation de la société ACI.

1985 : sortie de la première version de 4e Dimension.

1993 : 4e Dimension implémente le mode Client-Serveur, une avancée importante.

1995 : 4e Dimension s'attaque au marché Windows.

1997 : Implémentation d'un serveur http directement dans le SGBD.

1999 : le serveur web complet fait son apparition.

2002 : Sortie du 4D Business Kit.

2003 : Sortie de 4D 2003.

2004 : Sortie prévue de 4D, nom de code GoldFinger, nouveau moteur, nouveau core.

WINDEV 7.5

Le rival de Delphi ?

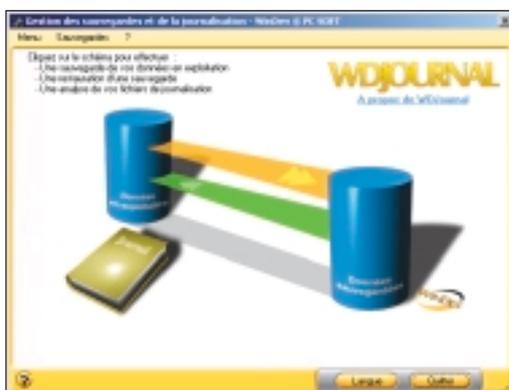
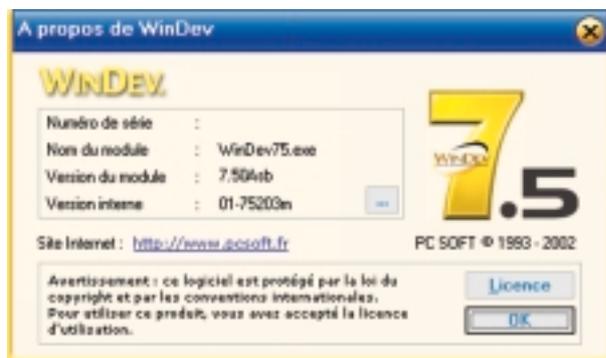
Passerelle idéale entre les environnements 16 et 32bits de Microsoft grâce à la génération d'une application propre à chaque plate-forme à partir d'un seul et même code source, WinDev 5.5 s'est particulièrement distingué dans le domaine de l'informatique de gestion. L'arrivée de la version 7 a non seulement marqué le passage définitif à l'ère 32bits (avec une réécriture complète du produit) pour structurer les évolutions futures, mais a introduit un ensemble de nouvelles technologies : base de données HyperFile 7, conception UML, correcteur d'IHM, etc

Après une année et demie d'exploitation, la version 7.0 du célèbre atelier de génie logiciel de PC Soft, laisse la place à une nouvelle mouture : WinDev 7.5. Résolument communicante et orientée vers l'urbanisation des applications, elle s'ouvre aux principaux standards du marché (J2EE, .NET, SOAP, XML), ne se limite plus à la gestion et concurrence férocement les autres outils de développement sur leur propre domaine de prédilection.

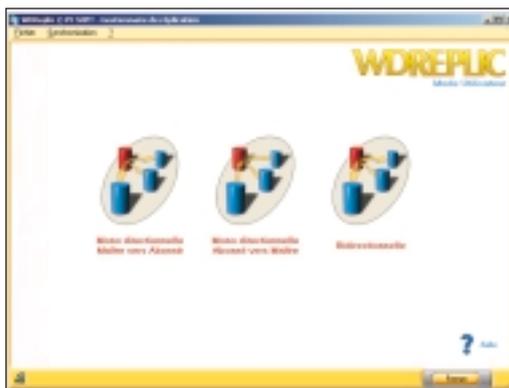
L'expérience acquise et la mise à l'épreuve de ses capacités s'expriment par une nouvelle présentation, enrichie et mature. Pour mémoire, la plate-forme de développement comporte, au sein d'une interface homogène et commune (en français), des outils de conception (Dossier d'Analyse, Merise, UML, ...), des éditeurs (Générateur d'IHM, d'Etats, Requêteur SQL Visuel, ...), une base de données (HyperFile) et ses applications connexes (réplication, transaction, journalisation, ...) architecturés autour d'un langage orienté objet : le W-Language

WinDev 7.5

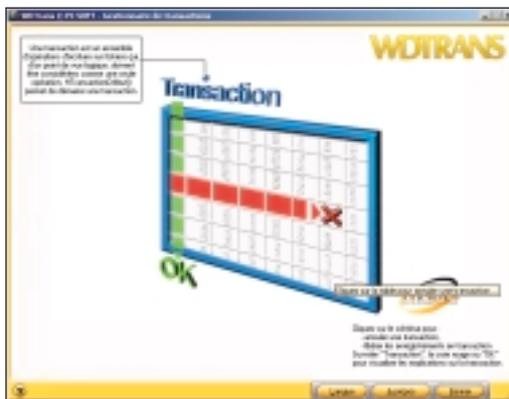
Le passage d'une version à l'autre s'opère sans douleur et ne requiert pas l'usage d'une procédure de migration (comme pour la version 5.5). L'importation d'anciens projets (5.5) reste toutefois disponible et bénéficie de l'expérience acquise sur les projets déjà migrés par les utilisateurs de la version 7. Les messages s'avèrent dorénavant plus explicites. L'assistant de migration offre également de meilleurs résultats. Les projets migrés



Utilitaire de journalisation



Utilitaire de réplication



Utilitaire de gestion des transactions.

ne requièrent une modification au niveau du code (et de leurs états) que lorsque la logique de dénomination d'une entité ou d'emploi d'une instruction a considérablement évolué. A l'aide des nouvelles possibilités offertes, l'intervention se solde alors souvent par une optimisation du code en termes de volume (moins de lignes) et d'efficacité (plus rapide). La migration d'un projet 5.5 nécessite donc un minimum de préparation et l'observation scrupuleuse des indications du dossier d'analyse d'impact (généralisé automatiquement). Au premier abord, la version 7.5 constitue d'avantage une évolution de la version 7 qu'une révolution (comme ce fût le cas pour la 7 par rapport à la 5.5). L'ergonomie de l'environnement, quoique peaufinée, ne dérout pas les habitués. Globalement, elle adopte la charte graphique similaire à celle retenue par Microsoft pour les versions XP de son système d'exploitation : esthétique et fonctionnelle. Les points clefs ne se dévoilent qu'ultérieurement, à l'usage.

WD7.5 intègre un mécanisme de gestion de composants. Attention, il ne s'agit pas ici d'un simple artefact graphique doté de propriétés, cliqué / déposé sur le fond d'une fenêtre et associé à une portion de code ("superchamp" en WD). L'entité s'incarne sous la forme d'une agrégation d'éléments nécessaires à son fonctionnement (analyse, fenêtres, états, ...) et correspond à une fonction précise : envoi de fax, dialogue avec un automate, service d'objets distribués, etc. Une fois conditionnée, sa distribution se réalise de façon commerciale (ou non) sous la forme de trois fichiers : le composant (wdk), sa description pour son intégration au sein d'un projet en cours (wdi) et l'archive des fichiers connexes (wdo). La définition, la génération (complète ou cliente) et l'intégration d'un composant s'opèrent à partir

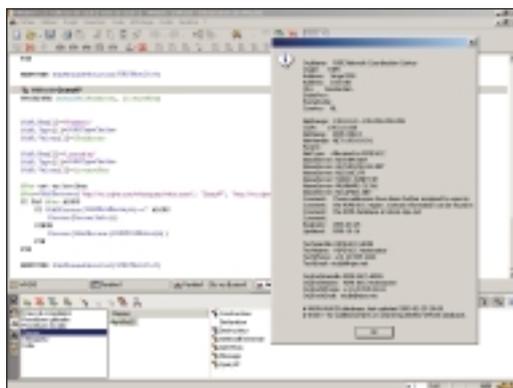
du menu "Projet / Composant", à l'aide d'assistants. L'ensemble des composants disponibles s'affiche dans le "kouglof" (volet inférieur bas de l'environnement intégré). Une icône et un court libellé précisent leurs fonctions respectives et permettent d'en consulter une description détaillée : informations relatives à l'auteur, mode d'emploi avec exemples. Les projets d'exemples livrés intègrent progressivement cette forme, pour faciliter le réemploi des éléments clés mis en évidence : Agent Microsoft, Codes Postaux, Moteur de Recherche, Gestion de Fax, de Login, etc.

WhoIS... la classe !

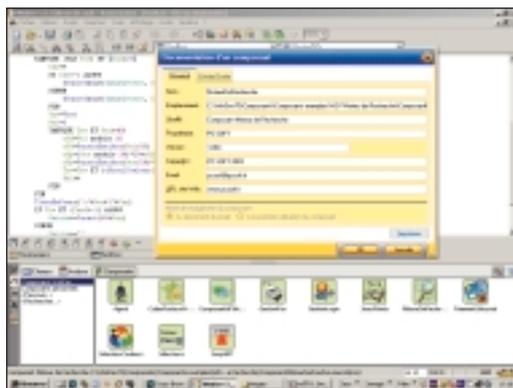
WD7.5 s'épanouit pleinement avec les technologies liées à l'emploi d'Internet. La gestion des threads et des sockets permet d'implémenter facilement des applications communicantes clientes, ou fournissant un service : serveur HTTP délivrant des documents HTML, WML ou XML à titre d'exemple. Les fonctions usuelles se montrent très facilement accessibles. La récupération du résultat d'une requête HTTP (POST ou GET) s'effectue ainsi en une seule instruction : HTTPRequête (https supporté). La manipulation de celui-ci, sous forme de chaîne de caractères ou par l'intermédiaire des fonctions XML proposées, donne accès aux données à extraire. Parmi la vingtaine d'instructions proposées, l'instruction XMLRecherche retient l'attention. Elle permet de parcourir tout ou partie du document, selon une condition relative à la nature des entités concernées (balises, attributs,...) et au mode de parcours : niveau courant, incluant ou non les sous-éléments, ... XMLNomElément et XMLExtraitChaîne permettent alors de récupérer le nom du nœud et sa valeur. Le dialogue avec le protocole SOAP et les services Web (.Net, J2EE) sont pleinement supportés et autorisent l'interconnexion de systèmes. L'importation d'un service s'effectue à partir de sa description WSDL (Web Service Description Language). A partir de l'indication de l'URL ou du chemin d'accès au fichier, l'assistant rédige au choix une classe ou une collection de procédures (selon vos préférences : POO ou non). Vous pouvez dès lors vous pencher sur les nombreux services proposés et agrémenter vos applications. L'usage de l'annuaire "http://www.webservicelist.com" offre un référentiel assez étoffé. L'adjonction d'un ser-



A la recherche d'un service web...



WhoIS... la classe !



Accès aux composants à partir du kouglof

vice repéré, par exemple "WhoIS", s'opère alors en copiant / collant l'URL de la description WSDL stipulé. Le code W-Langage suivant en illustre l'usage : "loWIS est un WhoIS", "Info(loWIS:QueryIP("195.246.150.253", ""))" (seulement 2 lignes). WhoIS est la classe générée par l'assistant. L'instruction "info" affiche les informations relatives à l'adresse IP indiquée, ou à son réseau d'appartenance. En pratique, l'usage de services Web autres qu'informels ou publics requiert l'acceptation d'un cahier des charges et l'adoption de modalités sous forme contractuelles, au même titre que toutes relations entre partenaires : clients, fournisseurs, transporteurs, ... WD7.5 autorise également la réalisation de service XML sous forme de bibliothèque

(WDL) à déployer sous Apache 1.3.x, IIS 5.1 ou iPlanet 4.1 (sous Windows). L'assistant, lors du conditionnement du service, génère sa description au format WSDL nécessaire à son emploi et permet ainsi son référencement et son exploitation.

Au niveau des différents éditeurs (de codes, de requêtes, d'états, ...), l'accent porte sur l'ergonomie et l'accessibilité. L'éditeur de requête offre une sélection simplifiée des éléments constituant la requête (zones, tris, opérations diverses, ...) et exprime la requête, non seulement en SQL mais également en langage naturel. La lecture de l'expression paramétrée en français dans le texte permet de réagir rapidement en cas d'erreur et surtout permet d'exprimer rapidement et simplement une requête objet de controverse sous une forme exploitable par tous (en particulier pour ceux qui ne connaissent pas SQL). L'éditeur d'IHM comporte désormais un champ supplémentaire : "web camera". A priori, l'apport d'un champ affichant la séquence vidéo transmise par une caméra reliée au micro-ordinateur ressemble à un gadget... surtout si le poste n'en dispose pas. Par contre, dès lors qu'un périphérique de ce type s'avère exploitable, le point de vue change : acquisition de photo pour le fichier des employés, de vignettes pour les fiches produits, etc. En fait, il se montre d'autant moins anodin que les bases de données sont couplées avec des supports multimédias : sites web, catalogues sur cédérom, etc. L'adjonction d'images, de sons, de commentaires audio ou vidéo, humanise les applications et donne un autre sens au mot interaction.

Un exemple de la lettre du support technique (n°51) propose même d'effectuer de la vidéo surveillance.

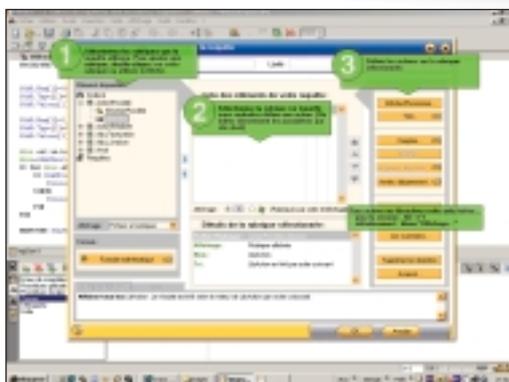
Delphi ?

WinDev 7.5 ne connaît toujours pas de concurrent direct. Les prétendants éventuels se situent dans une toute autre gamme de prix, à fonctionnalités plus ou moins équivalentes. Un produit - concurrent ? - vient toutefois à l'esprit (ne soufflez plus) : Borland Delphi Entreprise (en déclinaison complète). Les philosophies respectives de ces plateformes de développement diffèrent radicalement. Alors que WinDev se présente sous la forme d'un atelier de développement doté d'un langage, Delphi réside en un langage

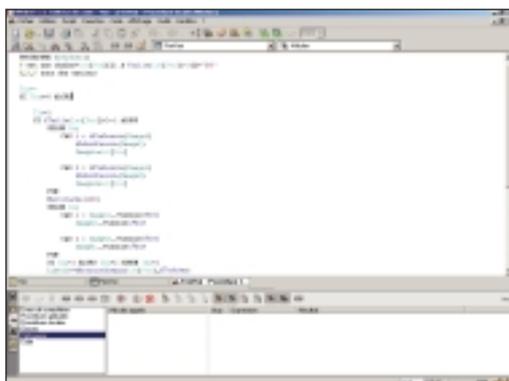
aggloméré à un ensemble d'outils d'origines diverses, pour répondre aux besoins usuels. Ces derniers changent au fur et à mesure des versions et peuvent se montrer incompatibles. Trois éditeurs d'états (Rave Report, QuickReport et ReportSmith) et trois familles de composants Internet (Indy, FastNet, NetManage) se sont ainsi succédés au fil des versions. L'argument clef de l'offre "une application Linux et Windows à partir d'un seul code source" implique la compatibilité entre Kylix et Delphi et nécessite l'emploi des composants multi plates-formes (CLX) en remplacement de ceux de la traditionnelle VCL, etc. La compatibilité ascendante revendiquée entre les versions se limite de fait au seul langage, à moins de ne pas exploiter les apports connexes (et alors quel intérêt d'évoluer de l'une à l'autre ?).

Avantages et inconvénients

WinDev, contrairement à Delphi, peut compiler du code à la volée en cours d'exécution, ce qui étend considérablement les possibilités : intégration de macros utilisateurs (en WD), utilisation de scripts (en WD), implémentation de serveur Web / XML utilisant au sein des documents le langage WD, algorith-



Editeur de requêtes, nouvelle formule



Editeur de code

me à base de code qui s'auto modifie, etc. L'indirection permet d'accéder à des zones écrans, fichiers, ... par l'intermédiaire de leurs noms exprimés sous forme de chaîne de caractères. La résolution opérée en exécution permet ainsi de modéliser des traitements : le nom ou la racine du nom des zones à traiter

peuvent être passé en argument. De nombreux détails facilitent la vie du WinDev-développeur à titre d'exemple : un glisser / déposer d'une zone d'un fichier sur le fond d'une fenêtre crée la zone de saisie, avec le libellé et la connexion à la source de données. En Delphi, il faut d'abord ajouter les composants : TTable, TDataSource et TDBEdit, puis éditer leurs propriétés respectives afin de référencer le TTable au niveau de TDataSource, puis le TDataSource au niveau de TDBEdit (en précisant la zone exploitée). Par contre, WinDev ne produit pas de DLL (mais des WDL spécifiques à WD). Il ne s'interface pas toujours facilement avec les objets COM / DCOM. L'usage d'un compilateur externe - non commercial de préférence - s'avère donc alors ponctuellement requis pour accéder à la fonction souhaitée (sauf si déjà disponible sous forme d'OCX ou d'une DLL).

Conclusion

Au risque d'une répétition, la conclusion énoncée pour la version 7 reste d'actualité. La convivialité et la polyvalence de l'environnement en font un outil hors pair. Le W-Langage, mixte intelligent de plusieurs langages, ne se limite plus - à la manière de Java - à en retenir les éléments pertinents, mais il s'enrichit de version en version. ■

Jean-Marc QUERE

Le point de vue de BORLAND

Nous avons demandé à Borland d'exprimer son avis sur la partie comparative de notre article.

Comparer Delphi et Windev n'est pertinent que du point de vue de la simplicité de mise en oeuvre. Mais Visual Basic eut été plus approprié, dans la mesure où PCSoft positionne Windev comme l'outil complémentaire (voire de substitution) à VBScript ou aux outils traditionnels du monde de la bureautique, tels que les gestionnaires de bases de données personnelles comme Access ou dBase.

Delphi est un environnement de développement intégré, compilé, reposant sur un framework (bibliothèque de classes) et un compilateur. Rien à voir avec un L4G. Il est donc par essence comparable à un Visual C++ ou Borland C++Builder. D'ailleurs avec quoi est écrit Windev? Mon petit doigt me dit qu'il s'agirait d'un environnement de développement intégré... de chez Borland!

De plus, comparer Delphi édition Entreprise, à

une édition Professionnelle, voire Personnelle, ne penche évidemment pas en notre faveur... du point de vue du ratio fonctionnalité/coût.

Enfin techniquement, le fameux framework de Kylix (le Delphi pour Linux) s'appelle CLX et la compatibilité ascendante est garantie entre les deux produits, dans la mesure où les deux l'utilisent aujourd'hui!!!

Le bémol venant des spécificités des OS (tels que COM/DCOM/COM+) qui ne sont pas restituables sur l'un ou l'autre.

Pour ce qui est du développement base de données, la bonne pratique conseillée depuis la version 3 de Delphi s'appelle le DataModule. Celui-ci permet de séparer la logique de gestion de données, de la logique métier et de l'interface utilisateur. Soit architecturer les développements pour un fonctionnement en mode distribué multi niveau. Inutile donc de paramétrer les

composants nommés dans l'article à chaque utilisation. L'objet, cela sert à cela!!!

Enfin, et c'est un des points majeurs en faveur de Delphi, l'écriture de DLL est une fonctionnalité native du produit. Faire appel à un deuxième outil, comme suggéré - même s'il est gratuit - ne va pas dans le sens de la rationalisation, ni des compétences techniques, ni des coûts directs et indirects, ni des sources.

Bruno de Combiens (Borland France)

- WinDev7.5 est effectivement partiellement réalisé en partie en C++ (le reste en WinDev), mais il ne s'agit pas d'un environnement de chez Borland (c'est du Visual C++)

- ...] compatibilité ascendante est garantie entre les deux produits dans la mesure où les deux produits *aujourd'hui* l'utilisent [... effectivement *aujourd'hui* d'où la remarque sur l'ascendance. ■

Jean-Marc QUERE

Altaprofits choisit le stockage natif XML

Altaprofits, site de courtage d'assurances, s'appuie pour son système d'information, sur une base de données relationnelle native XML. Un choix innovant, qui vise la performance des fonctions de manipulations et de mise à jour des données.

Créé en 1999, Altaprofits est un site de courtage qui a pour objectif : la commercialisation des contrats d'assurance vie, à travers le Web. " Notre portail permet aux clients particuliers et aux professionnels, d'accéder à différentes informations. Ils peuvent par exemple, simuler en temps réel une police d'assurance en fonction de la situation du futur assuré, accéder à des informations boursières, ou encore acheter directement leur contrat " déclare Philippe Jentrelle, responsable informatique du site. Pour réaliser le site, le cahier des charges repose sur une architecture Web hautement sécurisée et sur le format XML, pour faciliter les échanges de documents avec les clients. " La solution retenue devait répondre à deux impératifs. Le premier concerne l'échange de données à partir de plates-formes hétérogènes. Le second, concerne la conservation des formulaires dans leur intégralité pour des questions juridiques. A ce niveau, le format XML est incontournable dans le processus de description et d'archivage des documents Web " explique Philippe Jentrelle.

SGBDR ou SGBD native XML ?

En 99, Altaprofits est confrontée à deux possibilités : soit choisir entre les SGBDR dotés d'outils de mapping pour faire le lien entre les données non structurées et les documents XML, ou bien, opter pour une base de don-

nées native XML. "Notre choix s'est orienté vers la nouvelle génération de SGBD native XML, avec Tamino de Software AG. L'outil présente en effet des atouts dans le cadre de notre activité, avec la prise en charge directe des documents structurés : factures, contrats, bons de commandes, etc" indique le responsable informatique. Les bases de données relationnelles sont structurées selon un modèle rigide de champs et d'enregistrements, qui se prête mal à la représentation de la diversité et de la richesse des relations qui définissent

croître les temps de réponses de notre site " poursuit Philippe Jentrelle qui s'avoue satisfait de son choix.

Le stockage natif XML : risque et performance

"Nous partions de zéro. Nous n'avions pas à gérer l'existant d'une base de données rela-

tionnelle, ce qui est effectivement plus simple à mettre en place" remarque Philippe Jentrelle.

De son côté, Nicolas Farges, responsable R&D chez SQLi, trouve ce choix risqué. "En effet, les bases de données natives XML posent des problèmes, car les outils manquent de standards et de maturité. Il impose des développements et une administration spécifique. Il présente peu d'intérêt (excepté les performances)

puisque tous les acteurs proposent désormais des outils

de mapping pour faire le lien entre les données existantes et XML. " Mais pour Altaprofits, Tamino représente l'outil idéal. "Justement les performances sont un critère majeur pour notre activité. Le stockage natif XML du document directement indexé (contenu et structure), et l'implémentation physique adaptée à l'accès direct, simple et rapide de l'information, sont deux atouts indéniables " rétorque Philippe Jentrelle et d'ajouter " En outre, notre base s'administre facilement, car il y a beaucoup d'automatisa-



Le portail permet une simulation de prix en temps réel.

la structure des documents XML. En outre, un SGBDR dispose juste d'un index de tri, alors que Tamino intègre trois types d'index : texte, tri et un index de structure. De leur côté, les bases de données natives XML stockent les documents XML dans leur format d'origine, ce qui supprime le processus de liaison des données et des tables et améliore les performances, en particulier dans les fonctions de manipulations et de mise à jour des données. " Aucun mécanisme intermédiaire ne vient ralentir les accès ce qui nous permet d'ac-

tion et s'avère plus rapide que les SBDR traditionnelles, dans la mesure où les technologies sont adaptées aux documents XML "

Intégration et maintenance facilitées

" Cette architecture s'appuie sur des standards du marché et suit les directives des organismes de normalisation (W3C). De ce fait, son intégration et sa maintenance au sein des différents serveurs Web, serveurs d'applications, ou systèmes d'exploitation, sont facilitées " commente le Responsable informatique du site. Après trois ans de production, Philippe Jentrelle n'a pas regretté son choix. " Eviter les conversions entre stockage issu du relationnel et les documents XML nous est apparu comme la meilleure solution pour accroître la performance de notre site. "

Annie Lichtner

Java, XML et Cocoon...

Une architecture trois tiers est retenue pour garantir une meilleure répartition des charges et une optimisation des ressources. Cette architecture a prouvé ensuite son intérêt (modularité, évolutivité et maintenabilité) lors des évolutions majeures du site. C'est le framework d'application Cocoon, du groupe Apache, qui a été utilisé. Il permet de décomposer toutes les fonctions en trois tiers : les données XML, les règles métiers implémentées dans des feuilles de transformation XSLT et les pages de présentation issues de transformation en HTML, personnalisables en fonction du courtier. Un seul serveur Pentium 4 sous Windows NT intègre les trois couches. Il comprend : le serveur Web http avec Microsoft IIS, les échanges confidentiels (mot de passe, données contrats) sont sécurisés en https et l'ensemble des données est exclusivement

stocké, échangé, géré et publié par le serveur d'informations XML Tamino. L'ensemble des flux échangés et des données stockées dans Tamino sont modélisés en XML. Les traitements sont codés en Java, via le serveur JRun. En outre, les formulaires sont représentés en XML en s'appuyant sur les travaux du groupe XForms du W3C. Enfin, les documents contractuels sont générés automatiquement au format PDF, à partir de données XML, par le mécanisme de transformation XSLFO. Les parties sécurisées (connexion, consultation des éléments du contrat, passage d'actes) sont protégées par une connexion SSL. La non répudiation des actes non signés est assurée par un mécanisme de copie d'e-mail, destinée à un cabinet d'huissiers de justice. Les contrats sont ensuite envoyés à l'assureur par une connexion en temps réel, via le protocole CORBA.

Programmez !

SPÉCIAL 5^e ANNIVERSAIRE

Recevez gratuitement CodeChargeStudio l'outil RAD Web

Bénéficiez d'une remise exceptionnelle sur le prix de l'abonnement : 36,45 € au lieu de 45 €, prix normal. Cela représente 29 € d'économie par rapport au prix de vente au numéro, soit 45% !

Mais nous avons décidé d'aller plus loin, grâce à l'éditeur YesSoftware et son distributeur

Kapitec Software, en vous **offrant un exceptionnel outil de développement professionnel : Code Charge Studio version 2.0, vendu 499,95 €**



- ✓ **Rapide** : Réduisez votre temps de développement de sites Web dynamiques, sites institutionnels ou Intranet.
- ✓ **Puissant** : CodeChargeStudio est le 1er outil de programmation qui emploie un moteur de génération de code pour créer automatiquement des applications Web avec bases de données.
- ✓ **Intuitif** : spécifiez les interactions avec un simple clic, grâce aux assistants
- ✓ **Intégré** : cet IDE (environnement de développement intégré) comporte: éditeur HTML, Visual page designer, Code editor, et toutes les fonctions de navigation, de visualisation, d'intégration, de mise à jour.
- ✓ **Solutions pré-construites** : nombreuses applications fonctionnelles.
www.gotocode.com support.codecharge.com

- ✓ **Compatible** :
 - **Editeurs** : HTML: fonctionne avec les principaux logiciels du marché : Dreamweaver, GoLive, FrontPage.
 - **Langages** : Supporte ASP.net/C#, ASP/Vbscript, JSP, Java servlets, ColdFusion 4.0, PHP 4.0 et Perl 5.0. Supporte les serveurs d'application de l'industrie: ISS, ColdFusion, WebSphere, Weblogic, Netscape Enterprise Server
 - **Bases de données** : connexion rapide avec virtuellement toute base de données, dont SQL Server, DB2, Oracle, MySQL, Access
- ✓ **Sécurité** : intègre un Security Management, qui simplifie l'authentification et la gestion des accès, mots de passe etc.

Pour en savoir plus sur CodeCharge Studio : http://www.kapitec.com/Produits/CodeCharge/fr/codecharge_stu.htm

ABONNEZ VOUS IMMEDIATEMENT : voir coupon page 43



Licence du logiciel offerte par l'éditeur - RESERVE aux 100 premiers abonnés inscrits, à réception du règlement de leur abonnement - L'opération est gérée par KAPITEC Software. Les licences seront livrées par voie électronique sous 2 semaines, à réception de la demande dûment enregistrée par l'éditeur YesSoftware. www.programmez.com

Développer avec les composants serveur

PAR SAMI JABER

Les composants sont des briques logicielles autonomes pouvant être contrôlées dynamiquement et assemblées entre elles pour former des applications. S'il fallait définir le terme le plus répandu de toute l'histoire de l'informatique, "composant" serait sans aucun doute en tête d'affiche. Comment en serait-il autrement pour un mot aussi générique...



Pourtant, en effectuant une brève recherche sur Internet, voici la définition avancée pour composant : "Un composant est une unité primaire de déploiement et de composition spécifiée par une interface décrivant des services et des dépendances"

Si elle peut vous laisser perplexe, cette définition met en évidence plusieurs aspects clés de la technologie, sur lesquels nous insisterons tout particulièrement dans ce dossier. Tout d'abord, le terme "unité primaire de déploiement". Le déploiement prendra tout son sens lorsque nous aborderons concrètement les serveurs d'applications et les plates-formes J2EE, .NET ou les environnements du Libre. La notion de "composition" sera ensuite l'occasion d'insister sur l'importance de la granularité des composants. Celle "d'interface" quant à elle, nous mènera vers la notion

de contrats de services. Et les "dépendances" enfin, seront au centre du couplage entre composants et dicteront l'orchestration des services.

C'est dans cette optique que nous avons souhaité aborder un tel dossier. Le but n'est pas de vous donner de quelconques remèdes miracles quant au développement de composants métier ou technique, mais de défricher avec vous les nombreuses facettes et vertus des composants modernes, en insistant sur leur particularité dans leur plate-forme cible.

Nous prendrons, comme fil conducteur tout au long du dossier, l'exemple ô combien célèbre du composant Banque proposant les services de Débit et de Crédit. Vous verrez que si la plupart d'entre eux possèdent les mêmes caractéristiques, ils diffèrent surtout par leur implémentation respective.

1 • QU'EST-CE QU'UN COMPOSANT ?

Les composants sont des briques logicielles autonomes pouvant être contrôlées dynamiquement et assemblées entre elles pour former des applications. Pour faire un parallèle avec le monde du bâtiment, un composant représente une pièce quelconque, une porte ou une fenêtre, alors que l'objet constitue la matière première de votre construction. Autrement dit, la brique, le bois ou le verre nécessaires à la fabrication de ces éléments.

Ce bref raccourci ne suffit pas à mettre en évidence la notion dynamique d'un composant ou d'une application. Ainsi, contrairement à une maison qui, une fois construite, revêt un caractère figé, le composant lui,

vit au gré des échanges applicatifs et constitue l'élément premier d'un service. Le vocabulaire dans le monde des composants est essentiel pour bien comprendre leurs enjeux et leurs caractéristiques propres.

Chaque terme a une signification importante, dont vous verrez les effets, lorsque nous

aborderons les différentes implémentations de chaque plate-forme.

Réutilisation

La réutilisation constitue l'argument fondamental du modèle de composants. Il n'y a pas de réutilisation sans composant. L'objectif principal de la réutilisation est de tirer parti d'un service donné, pour éviter de redévelopper systématiquement le même service dans un contexte différent. Un objet n'est pas forcément réutilisable, un composant OUI.

La notion de service ou d'opération

Le but ultime d'un composant est de fournir un service donné, sans distinction entre service technique et service métier. Un composant métier Compte fournira les opérations Débitier, Créditer ou AfficherSolde, alors qu'un composant technique

proposera des méthodes telles que Sauvegarder, Ouvrir-Fichier ou ExtraireImage. Les services constituent la partie dynamique d'un composant et sont implémentés dans un langage de programmation donné (C++, Java, C# ou encore PHP).

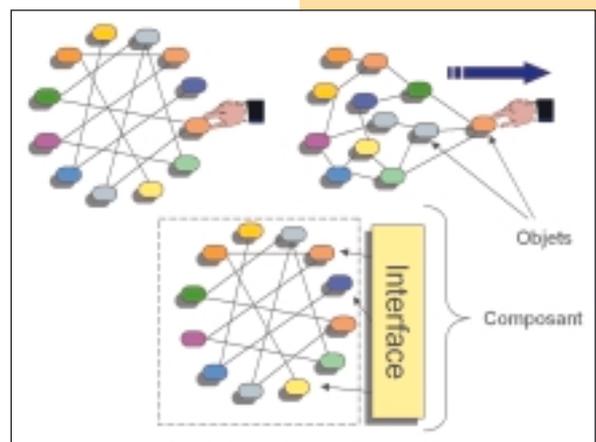
Contrat et interfaces

Les interfaces identifient et formalisent la liste des services proposés par un composant. Le terme contrat est ici très important, car une interface représente le contrat passé entre l'utilisateur d'un service et le fournisseur du service. L'interface sert également à dissocier les services de leur implémentation technique, pour éviter l'effet de couplage fort que nous verrons plus loin. En pratique, un contrat est représenté par une interface, dont le langage dépend de la plateforme cible : IDL dans le monde Corba, Java dans le monde J2EE ou C dans le monde C/C++.

Composition et relations de dépendances

Pour bien comprendre cette notion fondamentale, prenons un cas concret. Celui des composants électroniques. Une carte mère a besoin de plusieurs composants, tels qu'une carte vidéo et des barrettes mémoire pour fonctionner.

Il y a donc une relation de dépendance entre ces trois composants qui, pris séparément, peuvent très bien fonctionner. Imaginez maintenant que la carte vidéo s'appuie sur la mémoire d'un autre PC. Il vous sera alors impossible d'utiliser (ou plutôt de réutiliser) cette carte sans risque de briser la contrainte initiale liée à la mémoire. Cette mésaventure est due à la dépendance forte, existant entre ces deux



composants de nature très différente. En pratique, pour contourner ce problème, un composant ne sera réellement utilisable que lorsqu'il constituera une unité complète et indivisible.

Granularité

La difficulté, lorsqu'on aborde les composants, consiste à définir le niveau de granularité dans lequel on se place. Si on prend l'exemple précédent de la carte mère, quel est réellement le composant ? La carte mère, La carte vidéo ou la puce électronique ?

Si un composant peut correspondre à un objet simple, contenant un service affichant " Bonjour tout le monde ", il peut également être composé de plusieurs centaines de milliers d'objets ou de lignes de codes. Ainsi, les applications bureautiques telles que Excel ou Word constituent des composants à part entière. Qui l'eut cru ...

L'encapsulation

L'utilisateur d'un composant ne connaît pas la structure interne ni l'implémentation des services d'un composant. Non seulement il n'en a pas l'utilité, mais il ne doit jamais tenter de briser cette règle d'encapsulation. Le but étant d'empêcher que toute évolution dans l'implémentation d'un composant n'impacte l'utilisateur final. C'est ce qui permet à un constructeur de cartes vidéo de proposer toujours plus de fonctionnalités et de puissance aux utilisateurs, sans les contraindre à changer de carte mère.

Objet vs Composant

Cette question est un vaste débat, qui anime souvent les longues soirées d'hiver entre initiés ! En règle générale un objet fait appel à toutes les facettes de la programmation orientée objet. Il tire parti du polymorphisme, de l'héritage et possède souvent un état (l'ensemble des attributs de l'objet). Un objet ne constitue pas nécessairement une unité primaire de déploiement, du fait de ses nombreuses dépendances avec d'autres objets. Mais plus encore, un composant s'appuie sur les services techniques d'un Framework, alors qu'un objet est en général indépendant de tout socle logiciel.

Les services techniques proposés par l'infrastructure

Il n'y a pas de composant technique sans plate-forme d'exécution minimale. Si le dialogue entre composants s'effectue souvent par le biais d'un bus logiciel ou middleware, certains composants tirent parti de services tels que la sécurité, les transactions ou la persistance. En règle générale, on a coutume d'appeler ce socle technique un serveur d'applications.

Le couplage

Le couplage correspond à la nature et au type de dépendance pouvant exister entre deux composants. Le couplage le plus fort est celui reliant deux composants par leur implémentation. Vient ensuite le couplage moins contraint, identifié par les interfaces et enfin le couplage plus lâche, constitué par des échanges de messages (MOM, outils...)

2 • TYPOLOGIES PAR PLATE-FORME : J2EE, .NET ET LE MONDE DU LIBRE

Abstract : Voyons maintenant comment chaque composant tire parti de sa plate-forme cible pour implémenter un service donné.

Les trois types de plates-formes objets prédominantes du moment sont : J2EE, .NET et l'environnement libre. Pour autant, ces plates-formes ne constituent pas à elle seules la totalité des applications existantes du marché. C'est le cas notamment de CCM (Corba Component Model) qui propose un modèle de composant très novateur, ciblant de multiples environnements. A ce sujet, il est surprenant de voir que CCM ne trouve aucun écho dans la presse spécialisée, alors qu'il s'appuie sur une architecture multi plate-forme et multi langage.

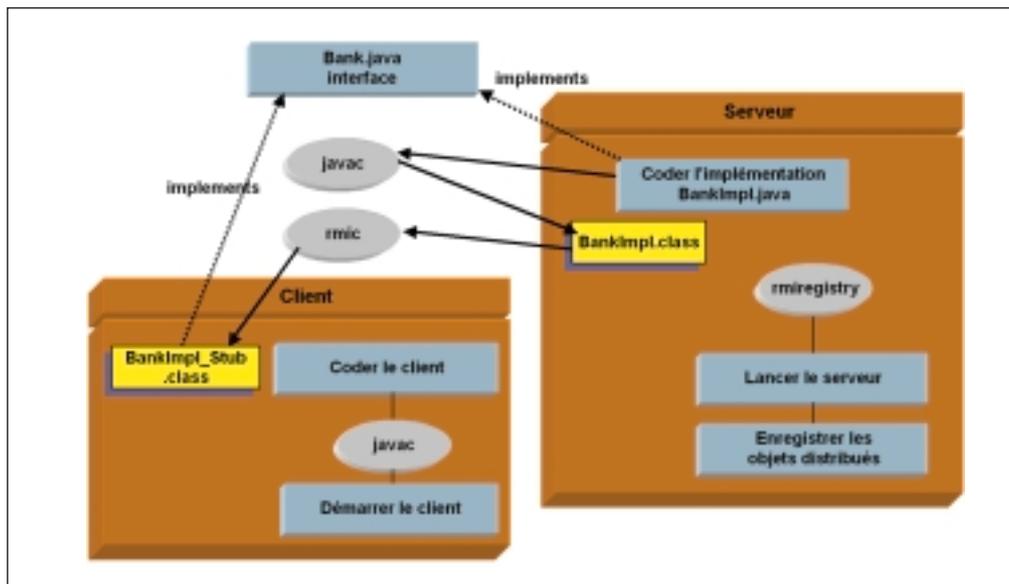
Les services techniques

Gardez à l'esprit qu'un composant est avant tout lié à l'infrastructure technique dans laquelle il puise ses ressources. Il est donc très important de saisir que l'intérêt principal d'un serveur d'applications est la fourniture de ces fameux services techniques :

● **La distribution** : (ce service) assure que l'invoca-

tion de services entre composants sera pris en charge par un bus logiciel (RMI, Corba, Soap, ...)

- **Les transactions** : assure la cohérence des appels en base de données et le respect des propriétés ACID (Atomicité, Consistance, Isolation et Durabilité) d'une transaction. Les transactions interviennent souvent dans le cadre de verrous, dits " pessimistes " ;
- **La sécurité** : assure que l'accès à un service donné se fera dans le cadre d'un appel sécurisé et authentifié ;
- **La montée en charge** : prend toutes les dispositions nécessaires à la montée en charge d'un composant (cache, activation juste à temps, réplification-synchronisation, etc) ;
- **La concurrence** : assure la cohérence des données d'un composant à état, notamment lorsque plusieurs personnes accèdent simultanément au même service ;



- **Le nommage** : fournit un annuaire permettant de localiser des composants de manière transparente ;
- **L'invocation asynchrone** : fournit une alternative à l'invocation de méthode synchrone par le biais de middlewares orientés messages
- **La persistance** : assure la synchronisation du modèle d'objets métier avec une base de données quelconque ;

Si la matrice précédente met en évidence quelques carences de l'environnement libre sur le plan des services techniques, il n'en demeure pas moins que cette plate-forme propose aujourd'hui de nombreux Frameworks ou langages de programmation objet. Toutefois, les initiatives dans ce domaine sont encore peu nombreuses, nous pouvons citer Zope, et l'hégémonie des applications J2EE/Java sur ces plates-formes n'y est pas étranger.

Les composants Java RMI (Remote Method Invocation) insérer illustration 2

Il nous faut différencier deux types de composants en Java. Les composants s'appuyant sur un middleware tel que Java RMI, sans aucune affiliation avec un serveur d'applications, et les composants Enterprise JavaBean, liés au socle J2EE et tirant parti des services techniques proposés dans la matrice précédente.

Le programme source suivant, correspond à l'interface du composant Banque en Java RMI :

```
package Banque ;

/**Interface définissant le service distant Banque*/
public interface IBanque extends java.rmi.Remote
{
    public void debit(int valeur) throws
        java.rmi.RemoteException;
    public void credit(int valeur) throws
        java.rmi.RemoteException;
}

```

Services techniques	J2EE/Java	.NET	Environnement libre (PHP, Perl, C, ..)
Distribution (middleware)	Corba (IIOP), JRMP (RMI)	.NET Remoting, Soap	XML-Rpc, Soap
Modèle de composant d'entreprise	Enterprise JavaBeans	Service Component (COM+)	Pas de standard (Zope)
Transactions	Automatiques et manuelles JTS/JTA (XA)	Automatiques et manuelles DTC, TIP (XA)	Manuelle
Sécurité	JAAS (intégré)	Intégré	Non intégré
Cache d'objets	Intégré	Intégré	Non intégré
Concurrence	Thread safe	Thread safe	Manuelle
Localisation des composants	JNDI	Aucun	Aucun
Invocation asynchrone	JMS, Message Driver Beans	Queued Component, MSMQ	Aucun
Persistance	EJB Entité, JDO, JDBC	ADO.NET	Manuelle
Connecteurs	JCA	Host Integration Server	Pas de standard

Rappelez vous la définition exacte d'un composant. Un composant possède une interface et une implémentation. Voici l'implémentation Java du même composant :

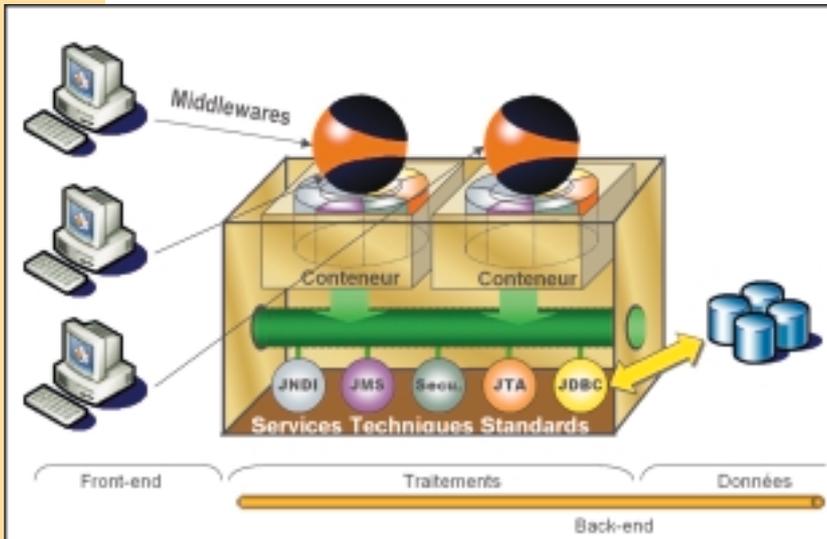
```
package hello;
import java.rmi.server.* ;
// Class that implements the RMI interface Hello.
public class BanqueImpl implements IBanque, UnicastRemoteObject {
    // Cette classe contient l'implémentation des services définis
    // dans l'interface
    private float solde = 0 ;
    public void debit(int valeur) throws java.rmi.RemoteException {
        solde-=valeur;
    }
    public void credit(int valeur) throws java.rmi.RemoteException {
        solde+=valeur;
    }
}

```

Cette forme d'écriture correspond aux applications Java développées avant l'apparition des ser-

► Composants

veurs d'applications. Aujourd'hui, de moins en moins de projets privilégient cette approche, car elle induit une dépendance forte avec le Framework technique RMI ou Corba, en fonction du protocole utilisé.



Par ailleurs, de tels composants distribués ne disposent d'aucun service technique (sécurité, transaction, cache) et requièrent un déploiement souvent périlleux, car ils nécessitent l'utilisation d'outils tiers, tels que des générateurs de Proxy et Skelettes (rmic).

Dans la pratique, hormis quelques rares exceptions, nous vous conseillons d'adopter un serveur d'applications, afin de masquer la tuyauterie de l'invocation distribuée.

Les composants J2EE (Enterprise Java Bean)

Tout ou presque, concernant les composants EJB, a été dit ou écrit. On ne compte plus le nombre d'ouvrages et de sites en tous genres vantant les mérites de cette technologie, un temps émergente, aujourd'hui incontestablement mature. Il semble toutefois que les ardeurs du début tendent à s'estomper au profit d'un scepticisme ambiant, notamment dans les rangs de la communauté Java, et ce, pour diverses raisons. Tout d'abord, l'objectif initial des EJB qui était de masquer et d'encapsuler l'ensemble des services techniques, a trouvé ses limites dans un modèle de développement toujours plus complexe. Aujourd'hui personne ne peut réellement prétendre écrire des composants EJB, sans posséder de solides compétences sur le sujet. Par ailleurs, les outils ou IDE, supposés décharger le développeur de toutes les tâches de configuration et de déploiement, peinent à atteindre un degré de maturité suffisant et souffrent souvent de manque de finition.

Malgré tout, l'avenir s'éclaircit. Preuve en sont, les nombreux efforts de la part d'IBM notamment,

mais également d'acteurs du monde du " libre " pour promouvoir la plate-forme Java du futur : Eclipse.

Concernant les types de composants, il convient de dissocier trois types d'EJB. Les EJB Session, contenant la logique applicative, les EJB Entités, destinés à masquer les opérations d'accès et d'écriture en base, et les EJB orientés Messages, totalement asynchrones. Si les premiers ont prouvé leur efficacité à travers les nombreux services qu'ils proposent, les seconds font face à de nombreuses critiques provenant de la communauté Java, qui les juge trop complexes et souvent peu performants. Par ailleurs, il semble de plus en plus probable que la prochaine génération de composants J2EE persistants, s'oriente vers un modèle proche du standard JDO et privilégie la notion d'objet métier épurée d'API techniques. Ce sujet est actuellement au cœur de nombreux débats et les prochaines spécifications J2EE présentées par Sun ne manqueront de susciter de vives réactions au sein de la communauté Java.

De manière plus pratique, voici à quoi ressemble un composant EJB Session, proposant les mêmes services que le composant précédent.

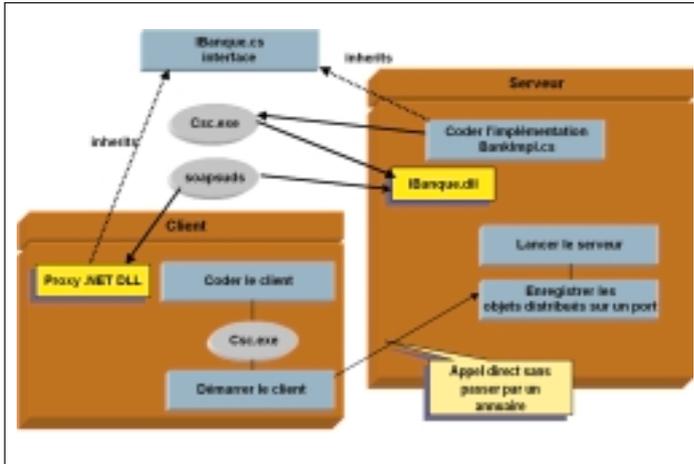
```
package banque;
import javax.ejb.* ;
import java.rmi.* ;
public class BanqueEJB implements SessionBean {
    private int solde = 0 ;
    public void debit(int valeur){
        solde -= valeur ;
    }
    public void credit(int valeur){
        solde += valeur ;
    }
    // Méthodes de notification des services techniques
    public void ejbActivate() {} ;
    public void ejbPassivate() {} ;
    public void ejbRemove() {} ;
}
```

Contrairement aux composants .NET, chaque composant EJB doit être accompagné d'une interface, d'une implémentation et d'une usine de classe (Factory). Les différents services techniques sont paramétrés à l'aide de fichiers de configuration au format XML (ejb-jar.xml) et le déploiement assuré au travers d'outils gratuits, tels que ANT (makefile XML).

Si le nombre d'objets nécessaires au développement d'un EJB est important, il existe aujourd'hui des outils permettant de prendre en charge un grand nombre de tâches. C'est le cas notamment de XDocLet, permettant à l'utilisateur de paramétrer son composant, via des commentaires insérés dans le code source.

```
/** This is the Account entity bean. It is an example of
 * how to use the
 * EJBDoclet tags.
 *
 * @see Account
 *
 * @ejb.bean
 *     name="banque/Compte"
```

D
O
S
S
I
E
R



Ces composants possèdent les mêmes inconvénients que leurs homologues Java, dans la mesure où le code nécessaire à leur implémentation est fortement couplé aux API du bus Remoting. Par ailleurs, la tendance actuelle va de plus en plus vers un support croissant des services de distribution par un serveur d'applications. Une démarche adoptée par le Framework .NET Enterprise Services avec COM+.

Les composants .NET Enterprise Services

Les " Serviced Component " ou " Enterprise Component " tirent parti du serveur d'applications COM+ et bénéficient de services techniques tels que la distribution, les transactions automatiques, la sécurité ou la montée en charge. Contrairement à Java, .NET ne propose aujourd'hui aucune notion de composant avec état (Stateful), peu enclin à la montée en charge. Malgré tout, plusieurs techniques ou modèles de conception permettent de pallier cette lacune en s'appuyant sur un objet distribué ou une base de données.

Pour mieux comprendre ce modèle de composant, analysons le code C# de notre fameux composant Banque avec COM+:

```
* type="CMP"
* indi-name="ejb/banque/Compte"
* local-indi-name="ejb/banque/Comptelocal"
* primaryKey-field="numeroCompte"
*
* @ejb.finder
* signature="java.util.Collection findAll()"
* unchecked="true"
* @ejb.transaction
* type="Required"
*
* @ejb.interface
* remote-class="ejb.interfaces.Compte"
* @ejb.value-object
* match="*"
*
* @version 1.5
*/
```

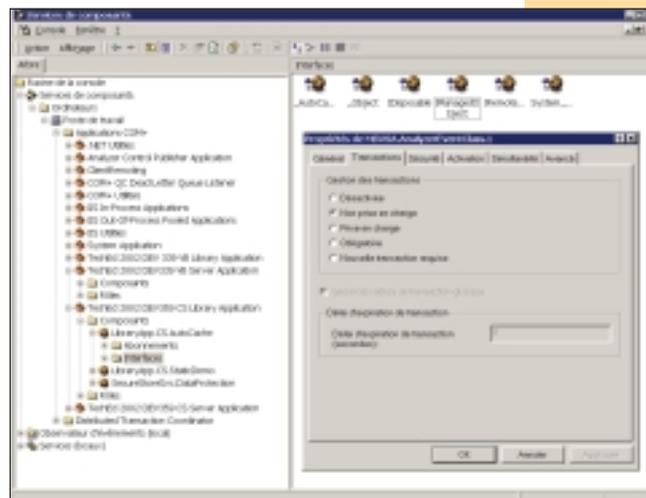
Lors de la phase de compilation et de déploiement, une tâche ANT est chargée d'invoquer l'outil XDoclet qui déclenchera la génération automatique des squelettes de code sources (interfaces et usines de classes). Voyons si le même composant en .NET est aussi trivial à réaliser.

Les composants .NET Remoting

Dans le monde .NET, il existe plusieurs types de composants et il n'est pas toujours aisé de s'y retrouver. Les composants .NET Remoting, quant à eux, se rapprochent plus des composants RMI de Java et ne requièrent pas la présence du conteneur COM+.

```
using System.EnterpriseServices;
namespace Banque {
    public class ComposantBanque : ServicedComponent {
        [AutoComplete]
        [ObjectPooling(MinPoolSize = 3,MaxPoolSize = 10)]
        public void Debit(int valeur) {
            solde=valeur ;
        }
        public void Credit(int valeur) {
            solde+=valeur ;
        }
        // méthodes de notification des services techniques
    }
}
```

```
using System;
using System.Runtime.Remoting;
using System.Runtime.Remoting.Channels;
using System.Runtime.Remoting.Channels.Tcp;
namespace RemotingSample {
    public class ComposantBanque : MarshalByRefObject {
        float solde = 0 ;
        public void Debit(int valeur) {
            solde=valeur;
        }
        public void Credit(int valeur) {
            solde+=valeur;
        }
    }
}
```



```
public override void Activate(){}  
public override void Deactivate()  
public override bool CanBePooled(){return true ;  
}
```

Vous remarquerez les grandes similitudes existant avec le composant EJB Session précédent, ainsi que la présence d'attributs de Runtime placés entre []. Ces attributs assurent le paramétrage des services techniques du serveur d'applications, pour chaque méthode de notre composant et diffèrent des tags XDoclet de J2EE, dans le sens où ils n'ont pas vocation à générer statiquement du code.

Le déploiement

L'unité de déploiement primaire en Java est la classe. La compilation d'un fichier source se traduit par la génération de ByteCode sous la forme d'un fichier nomclass.class. En Java, l'ensemble de classes constituant un EJB est archivé dans un fichier au format JAR. Cet exécutable est ensuite déployé dynamiquement (déploiement à chaud) ou statiquement (redémarrage du serveur) dans un serveur d'applications J2EE.

Concernant .NET, la démarche est très similaire, dans la mesure où les programmes sources produi-



sent une assembly (DLL ou EXE) au format MSIL (bytecode) déployée et exécutée dans le serveur d'applications maison de Microsoft, COM+.

Les services techniques de .NET

Tout fonctionne comme si .NET n'était qu'un modèle de composants et COM+ la plomberie chargée de les faire fonctionner. Pourtant, il y a une différence fondamentale entre .NET et son infrastructure technique COM+. En effet, .NET

fonctionne dans un univers managé avec un ensemble de types unifiés, alors que COM s'appuie sur des anciennes bibliothèques, sous un format binaire non managé. Si cette situation n'est pas pénalisante, dans la mesure où le code source des composants .NET n'est en aucun cas impacté par les API COM+, il est peu probable qu'à l'avenir Microsoft homogénéise son Architecture, pour mieux soutenir la comparaison avec les services techniques managés de J2EE.

Les composants du monde du libre

Le monde du libre fait office de pionnier dans la réflexion et la mise en place d'un véritable modèle de composants. Le noyau de Linux constitue ainsi une référence en la matière, avec une Architecture modulaire et " Pluggable ". Cet exemple démontre bien qu'un modèle de composant n'est pas nécessairement associé à la fourniture de services techniques par un serveur d'applications. En pratique, il suffit que l'ensemble des caractéristiques énoncées dans la première partie de ce dossier soient vérifiées, pour qu'un composant soit considéré en tant que tel.

Ceci dit, vous ne verrez jamais personne utiliser le modèle de composants C/C++ du noyau Linux, pour écrire une application de gestion. A chaque modèle de composant, correspond ses exigences. Et dans cette optique, on aura plus tendance à tirer parti de langages de scripts, tels que Php ou Python, pour développer ses composants. Par ailleurs, contrairement aux idées reçues, il est tout à fait possible de produire une application distribuée et orientée objet en Php, en Perl ou en Python. Le composant suivant le prouve aisément :

```
<?  
class Banque {  
    var $solde=0;  
    function Debit($value){  
        return (int) solde - value ;  
    }  
    function Credit($value) {  
        return (int) solde + value ;  
    }  
    function daysInMonth() {  
        if( $this->change)  
            $this-> calc();  
        return date( "t", $this->ts );  
    }  
}  
?>  
// $NouveauSolde = Banque::Debit(1000);
```

Php ne propose peut-être pas autant de services techniques que .NET ou J2EE, mais s'avère un concurrent redoutable de J2EE et .NET pour certains types de projet avec des contraintes fortes, privilégiant les délais de développement à la réutilisation. Concernant la distribution de ces composants, elle peut aisément être prise en charge par des outils tels que XML-RPC ou Soap pour php. ■

Les composants graphiques JavaBeans et Active X

Difficile d'aborder un dossier sur les composants sans aborder les composants ou contrôles utilisateur. Une grille de données, une liste de boutons radio ou un champ de formulaire constitue un composant, au même titre que leurs homologues serveur. Ils possèdent une interface, des dépendances, une granularité et sont réutilisables.

En revanche, le socle technique diffère, du fait de la vocation première de ces contrôles qui est d'apparaître dans un formulaire ou une interface graphique. Ils sont par ailleurs totalement désynchronisés du serveur d'applications et s'exécutent sur un poste client.

Dans le monde Java, les composants JavaBeans prédominent, avec un modèle basé sur la notion d'événement et de dépendances inter contrôles. C'est ce qui permet notamment d'attacher visuellement un contrôle à plusieurs événements (click, timer, etc) sans avoir obligatoirement à coder.

Dans le monde Microsoft, les Active X représentent l'équivalent des JavaBeans avec un modèle lié à COM (Component Object Model) et à Ole Automation pour l'assemblage de composants. Nous vous invitons à parcourir en annexe les sites de référence proposant ce type de contrôle en téléchargement.

Les Web Services, la glue universelle

Les modèles de composants que nous venons d'aborder dans ce dossier possèdent le même défaut. Ils fonctionnent très bien dans un environnement homogène, mais peinent à communiquer dans un environnement hétérogène. Grâce au duo Soap et HTTP, Les Web Services ont apporté un élan de jeunesse aux échanges inter-composants en remettant au goût du jour des modèles de distribution souvent bien éprouvés. Toutefois, s'ils apportent leur lot de nouveautés, les WebServices ne signent en aucun cas la mort de protocoles tels que RMI, Corba ou autres .NET Remoting. Ils constituent simplement la glue universelle, qui permettra à toutes ces technologies en mode connecté, de communiquer entre elles. Enfin, quoi qu'on en dise, les WebServices ne sont pas la réponse à tous les maux. Tant qu'ils

s'appuieront sur un protocole HTTP vieillissant et inadapté aux échanges en mode connecté, ils ne fourniront jamais des services essentiels, tels que les notifications asynchrones ou les Transactions distribuées. Essayez de faire un Chat avec les WebServices et vous comprendrez...

```
using System.Web;
using System.Web.Services;
namespace Banque {
    public class ComposantBanque : System.Web.
        Services.WebService {
        [WebMethod]
        public void Debit(int valeur) {
            solde-=valeur ;
        }
        public void Credit(int valeur) {
            solde+=valeur ;
        }
    }
}
```

LE MAGAZINE DU DÉVELOPPEMENT
Programmez !

Prochain numéro : n° 53 - Parution 30 avril

Dossier :
**Développement pour les
MOBILES**

3 • BANQUE POPULAIRE TOULOUSE MIDI-PYRÉNÉES

UN EXEMPLE DE DÉMARCHE PAR COMPOSANTS RÉUSSIE



Nicolas Ochoa (Responsable des projets .NET au service études informatiques de la Banque Populaire Toulouse Midi Pyrénées).

Un exemple de démarche par composants réussie

Il y a six mois, la Banque Populaire Toulouse Pyrénées a décidé d'engager une étude afin de migrer plusieurs applications existantes dans un environnement intranet.

Le choix de .NET

Après une période de réflexion et l'aide de la société de conseil Valtech, le choix s'est rapidement porté vers l'Architecture .NET. Les motivations de ce choix étaient multiples. Tout d'abord les contraintes en terme de délais imposaient la livraison de six applications dans un laps de temps très court, soit moins de six mois. Ensuite l'obligation de la plateforme cible à s'intégrer avec l'existant, notamment des bases de données sous AS400/DB2, UNIX/DB2, Sql Server et Terradata ainsi que les outils bureautiques de la gamme Office. Nicolas Ochoa, responsable des projets .NET au service études & systèmes informatiques explique : " La problématique était multiple. Tout d'abord, il fallait permettre aux équipes internes ayant une culture mainframe client serveur d'enrichir leurs compétences, de migrer rapidement vers ces technologies tout

pe mixte composée de consultants Valtech et de développeurs internes. Jean-Max Huet, l'architecte technique du projet attribue ce succès en partie à la productivité apportée par un outil tel que Visual Studio .NET associé au Framework ASP.NET.

Une démarche par composant privilégiée

Dès le départ, la démarche par composant a été privilégiée, car le besoin de réutilisation était grand. Les équipes du projet ont donc dû développer un socle de composants techniques destiné à homogénéiser l'accès aux différentes bases de données de la Banque. Puis dans une seconde phase, une analyse a été réalisée en UML afin d'identifier les composants métier réutilisables. Comme le précise Nicolas Ochoa, " Nous souhaitons mettre en place une architecture n-tiers afin de nous dégager de la problématique des composants métier mouvants dû à certaines dispositions légales, et également ne pas être lié à telle ou telle base de données puisque nous évoluons dans le cadre de regroupements et de mutations régulières en terme d'architecture ". Si le résultat est à la hauteur des espérances du service études & systèmes informatiques, il n'en reste pas moins que le



en utilisant un outil de développement intuitif dont les briques seraient intégrées et accessibles. ", Nicolas Ochoa ajoute " Le second point tenait à la nature même du métier bancaire. Il est nécessaire de répondre rapidement et par un coût raisonnable à des besoins de développement d'outils de pilotage ou de logiciels de gestion sans pour autant maîtriser en profondeur l'architecture .NET, du moins dans un premier temps."

Après quelques mois, pas moins de sept applications de gestion ont été développées par une équipe

projet a mis en lumière certains déficits de la plateforme .NET. Ainsi, comme le fait remarquer Jean-Max Huet : ".NET souffre de lacunes, notamment au niveau de la genericité de la couche d'accès aux données. Nous avons plusieurs bases de données à gérer et notre code était fortement lié à un provider spécifique, ce qui rendait nos composants moins réutilisables ". C'est donc au prix de quelques efforts, notamment dans le développement de composants

génériques d'accès aux données, que l'application a pu proposer l'interfaçage avec de multiples sources de données hétérogènes.

Aujourd'hui, le service études & systèmes informatiques est en passe de déployer au siège et dans les agences plusieurs de ces applications et les premiers retours semblent satisfaisants même si, aux dires de certains " les habitudes liés à l'ergonomie du client/serveur vont être difficiles à perdre ". ■

D
O
S
S
I
E
R

4 • COMMENT ORGANISER SES FRAMEWORKS ?

Avec le succès que connaissent actuellement les langages objets, la notion de "framework" est sur le point de devenir une démarche incontournable de tout développement. Ce terme, qui signifie à l'origine "cadre", apparaît en effet dans le discours de nombreux chefs de projets et architectes, comme une démarche structurante dans leurs développements. Mais quelle est sa réelle signification ? Comment un projet, tenu par des contraintes de coûts et de délais, peut-il être fondé sur la mise en oeuvre et l'utilisation de frameworks ?

Concrètement, un framework est un ensemble de classes logiquement dissociable du reste du code. En effet, un framework porte toujours en lui une notion de "réutilisation", au moins au niveau projet ; le framework garde alors son sens premier, qui est de fournir un "cadre" aux développements. A l'extrême, le framework est physiquement dissociable du projet (il se présente alors sous la forme d'une archive Jar sous la plate-forme J2EE et d'une Assembly sous Microsoft .NET), et peut être réutilisé pour d'autres développements. Dans les deux cas, cette notion de réutilisation implique que les concepteurs de frameworks soient capables de s'abstraire plus ou moins du métier de l'application développée. Plus la réutilisation sera grande, plus l'abstraction devra être forte.

Aujourd'hui, les développements autour des nouvelles technologies mixent de plus en plus deux démarches : l'utilisation de frameworks externes, techniques ou fonctionnels, et la mise en oeuvre de leur propre framework métier, en espérant ainsi combiner productivité et normalisation. Qu'apportent ces deux approches ?

Mettre en oeuvre un framework métier : une démarche de normalisation

Concevoir un framework métier est avant tout une démarche d'abstraction, la finalité étant de comprendre le métier de l'application développée (ou même d'un ensemble d'applications) pour en extraire une partie commune réutilisable. Encore une fois, la notion de réutilisation peut se limiter au contexte d'un seul projet, chaque domaine fonctionnel s'appuyant alors sur le framework métier. Prenons par exemple le cas des applications traditionnelles de gestion. Ces applications proposent de manipuler des informations métier et d'effectuer des opérations simples sur ces informations : création, édition, modification et suppression. Grâce aux technologies objet, il est possible de "mutualiser" ces traitements dans des classes abstraites, puis de faire dériver les objets concernés de ces classes : [figure 1](#)

Cette démarche peut être la première étape de conception d'un framework métier, même si son intérêt se limite à la normalisation des développements, puisque chaque classe dérivée devra implémenter concrètement les méthodes décrites précédemment. En revanche, il est possible de compléter cette première conception avec l'application de design patterns. Ainsi, avec l'application des patterns Data Access Object et Transfer Object, le modèle de conception s'enrichit considérablement : [figure 2](#)

Outre les avantages intrinsèques des patterns Data Access Object et Transfer Object (indépendance vis-à-vis du système de persistance des données, et normalisation du transport des informations), ce modèle de conception permet d'implémenter une partie des traitements, directement dans la couche framework. Les classes métier ne seront plus alors contraintes de surcharger les classes du framework, pour leur donner des implémentations concrètes.

Comme le suggère le modèle précédent, la mise en place d'un tel framework apporte de nombreux avantages, le principal étant de donner un cadre pour les développements de la couche métier. En effet, le développeur aura à charge d'implémenter les traitements propres de chaque classe et les méthodes abstraites, définies dans le framework. On conserve bien une notion de réutilisation, puisque le cadre ainsi défini sera repris constamment au cours du projet, et, s'il est suffisamment générique, pourra être conservé dans d'autres développements.

En revanche, il faut bien garder à l'esprit que ce type de framework est proche de l'application que l'on cherche à développer. Il existera obligatoirement une adhérence entre l'application et le framework en question (au niveau des types de données ou des signatures de méthodes par exemple). Il n'existe pas de framework "universel", capable de s'adapter dans toutes les situations, et même s'il existait, il introduirait une complexité de développement trop conséquente (imaginez par exemple de manipuler des méthodes prenant uniquement des données de type Object en paramètres et en sortie). La conception d'un framework métier reste donc avant tout un habile compromis entre compréhension du métier de l'application, et abstraction vis-à-vis de ce métier.

Utiliser un framework externe : une productivité accrue

L'utilisation de frameworks externes dans les développements est, pour les programmeurs, une op-

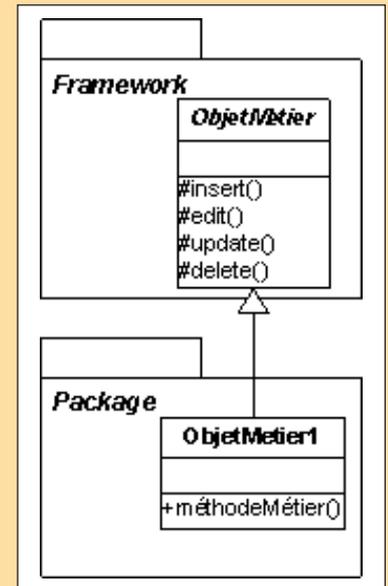


Figure 1

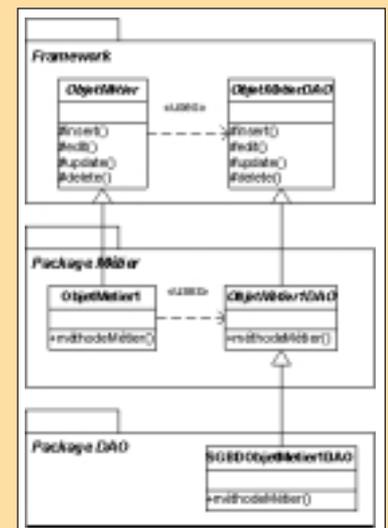


Figure 2

portunité de se décharger de tâches répétitives et lourdes. Qu'ils soient purement techniques (comme la gestion des logs) ou technico-fonctionnels (comme la gestion de contenu), intégrer un ou plusieurs frameworks externes, au sein d'une appli-

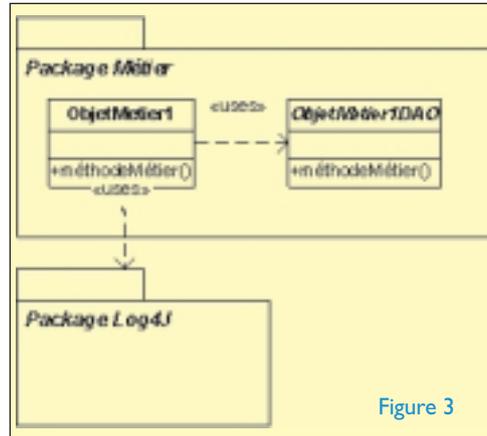


Figure 3

cation, limite les développements aux caractéristiques propres de l'application (interface homme machine, logique métier, etc.).

Considérons par exemple l'environnement J2EE ; il existe aujourd'hui une multitude de frameworks prêts à l'emploi, dont certains font même office de référence dans le développement d'applications Web. Lucene (moteur de recherche), Struts (implémentation du pattern MVC2 et bibliothèque de tags JSP), ou même Log4j (gestion des logs), sont autant d'exemples de frameworks qui sont au coeur des développements actuels. Mais comment intégrer correctement ces frameworks ?

Un problème inhérent à l'utilisation de frameworks externes est la dépendance entre l'application développée et le framework en question. En effet, toute modification sur le framework (ne serait-ce que pour des corrections de bugs) peut impacter l'application en question, et, au pire, nécessiter des modifications de code. Voici par exemple l'introduction du framework Log4j au sein de notre application précédente : [figure 3](#)

Cette utilisation classique de Log4j introduit directement une dépendance entre l'application et le package Log4j. Si une montée de version de Log4j est nécessaire, l'application "cliente" peut être impactée par des modifications (sur les signatures des méthodes ou des ajouts de classes par exemple). Encore une fois, l'utilisation de design patterns appropriés permet de résoudre ce problème ; il suffit pour cela d'inverser la dépendance et d'introduire un pattern Adapter : [figure 4](#)

Dans ce modèle, l'interface entre la logique métier et l'adaptateur est supposée stable. Dans ces conditions, toute évolution du package de logs demandera uniquement la création d'un adaptateur approprié, sans pour autant impacter la logique métier de l'application. L'utilisation de patterns a permis en quelque sorte de "stabiliser" la gestion des logs au niveau de l'application, pour s'affranchir du cycle de

vie du framework externe. Evidemment, certains frameworks sont beaucoup plus intrusifs dans une application, que simplement par l'import de classes. D'autres frameworks, Struts notamment, façonnent complètement une partie de l'application selon un modèle précis, qu'il est difficile de remettre en question.

Une démarche bénéfique, mais coûteuse

Si l'intégration de frameworks au sein d'une application n'est pas une obligation, il faut admettre qu'elle se justifie souvent, ne serait-ce que pour éviter de "réinventer la roue". La plupart des problématiques techniques classiques des applications Web trouvent aujourd'hui une réponse dans des frameworks ou solutions commerciales. Ceci est d'autant plus vrai dans le monde J2EE, du fait de la maturité de la plate-forme, et commence à être le cas sur l'environnement Microsoft .NET.

En revanche, il faut garder à l'esprit que l'organisation d'une application autour de frameworks introduit une complexité supplémentaire, complexité inhérente aux frameworks utilisés. Ainsi, utiliser Struts ou tout autre framework de présentation implique que l'équipe de développement monte en compétences sur Struts. Mettre en place un framework métier permet certes de normaliser et de structurer une application, mais la conception de l'application sera plus complexe.

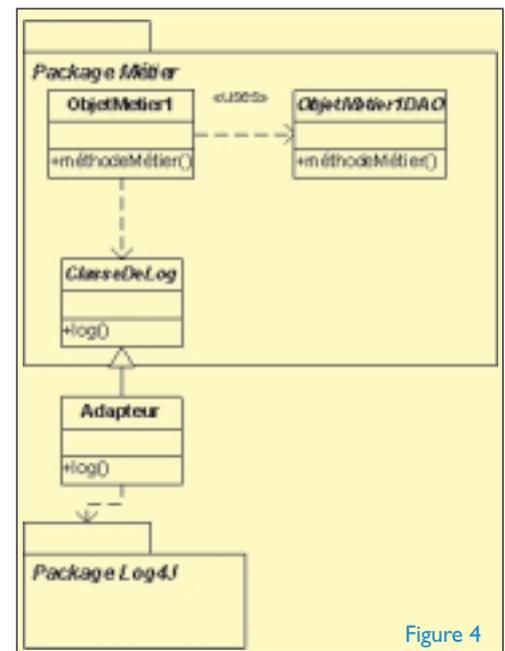


Figure 4

L'utilisation de frameworks n'est donc pas une "recette miraculeuse" ; même si cette démarche a de nombreux avantages, elle a également un coût associé.

Julien Soyer
Consultant Architecte
Groupe SQLI



Métier : intégrateur d'applications

par Carole Pitras

Une des nouvelles composantes des métiers du développement est la part croissante de l'intégration. Compte tenu du développement des progiciels au détriment des applications propriétaires, de la part croissante des composants et de la tendance accrue à la réutilisation (cf. précédent dossier de Programmez !), le métier d'intégrateur d'applications, ou ingénieur d'intégrations a été défini.

La définition de ce métier n'est cependant pas homogène. Selon les entreprises, l'intégrateur sera associé à la direction de projet au niveau de la maîtrise d'œuvre afin de l'assister au niveau de la planification ou au niveau de la réalisation, pour intégrer au sens propre les composants de l'application.

Sébastien Pautasso-Chadoutaud chez Oberthur Consultants, qui réalise une étude semestrielle sur les salaires des informaticiens, souligne que "chaque entreprise a sa vision de l'intégration d'applications" et indique que cette profession relève parfois de l'étude en amont du projet ou de la réalisation au niveau analyse/programmation.

Cette dichotomie se retrouve dans les définitions du métier d'intégrateur, qui varie dans les nomenclatures Cigref (qui représente les entreprises utilisatrices) et Syntec (syndicat des SSII et éditeurs).

Pour le Cigref, l'intégrateur d'applications, également intégrateur de développement, ou encore concepteur intégrateur "intervient dans le développement d'une nouvelle application ou dans la maintenance d'une application existante". Il participe, sous la responsabilité du chef de projets à la maîtrise d'œuvre, "au choix des différents composants logiciels (progiciels, bases de données, développements spécifiques...) et en assure l'assemblage, dans le respect du plan d'urbanisme des systèmes d'information de l'entreprise et de l'architecture retenue pour le projet."

"En ce qui concerne les développements spécifiques, les travaux sont effectués soit en interne par le développeur, soit en externe avec l'aide d'une société de services". Toujours selon cette définition, il assure les quatre missions suivantes : identification et sélection des composants techniques du projet, définition des interfaces et des éventuelles évolutions à apporter aux composants pour permettre leur intégration, réception, validation et assemblage de ces composants et enfin fourniture du système développé à l'intégrateur d'exploitation.

La définition du métier d'ingénieur intégration dans la nomenclature Syntec est la suivante : "Sur les projets dont le développement est relativement simple, l'intégration est généralement réalisée par les développeurs eux-mêmes. Mais, dans des cas de plus en plus nombreux, la complexité grandissante des systèmes logiciels ou matériels implique le recours à des spécialistes de l'intégration. La mission d'un ingénieur intégration consiste à rassembler les différents éléments de développement sur une plate-forme d'intégration. Cette tâche repose sur une importante phase de tests, respectant une méthodologie et des procédures rigoureuses : spécification, planification, réalisation, bilan. Il s'agit notamment de vérifier la compatibilité entre les différents élé-

ments, logiciels, matériels ou systèmes. Une fois les tests effectués, l'ingénieur intégration remédie aux bugs et aux erreurs, en étroite collaboration avec l'équipe de développement. Bien souvent, il lui revient de superviser la qualification du produit. Par la suite, il peut être associé à la conception des nouvelles versions des produits sur lesquels il est intervenu initialement."

Quel profil ?

Les ingénieurs intégration sont généralement des ingénieurs (Bac+4 ou 5 donc) expérimentés, ayant eu souvent une première expérience dans le développement, ou tout du moins technique, afin de connaître les outils utilisés, et qui se caractérisent par leur rigueur. Le Syntec relève que "le sens du travail de précision, la rigueur, le goût pour les questions techniques et la résistance au stress sont indispensables pour s'épanouir dans cette fonction. Un sens du relationnel s'impose également pour obtenir facilement des réponses à certaines questions très techniques, ou pour faire valoir son point de vue sur certains aspects du développement." Pour Stéphane Potdevin, directeur d'agence chez Helium, les qualités d'un intégrateur de progiciel ne diffèrent pas beaucoup de celles d'un développeur, si ce n'est une expérience préalable des systèmes d'information et de leurs contraintes, afin de concilier au mieux fonctionnalité du progiciel et méthodes du client utilisateur. Il doit bien sûr connaître le logiciel concerné, au niveau organisationnel et fonctionnel.

Comme tous les profils expérimentés et devant la recrudescence des progiciels, ce type de profil est assez recherché. Pour le Cigref "part croissante des composants de type progiciels dans les projets d'une part, complexité et foisonnement des technologies et des composants à maîtriser d'autre part, sont les deux facteurs qui expliquent le besoin croissant en intégrateurs d'applications, ainsi que le fort besoin de formation et de renouvellement des savoir-faire techniques de ces profils."

Quelle rémunération ?

L'étude Oberthur consultants distingue les salaires pratiqués dans les entreprises utilisatrices et dans les SSII/éditeurs. Dans les premières, la rémunération annuelle médiane d'un chef de projet est de 49 273 euros. Les intégrateurs compris dans ce panel sont plutôt au-dessous, dans la tranche 43 KE/49 KE. S'ils sont considérés comme le haut du pavé dans les métiers de l'analyse et de la programmation, leur salaire médian est de 45 269 euros et un quart d'entre eux gagne plus de 49 731 euros, avec une moyenne d'âge de 39 ans et 13 ans d'expérience professionnelle. Dans les SSII, la situation est différente à niveau hiérarchique équivalent. S'ils interviennent au niveau de la direction de projet, comme ingénieur étude confirmé, le salaire médian est de 41 712 euros, avec un quart qui a une rémunération inférieure à 36 862 euros et un quart une rémunération supérieure à 47 254 euros. S'ils entrent dans la case réalisation, le salaire est assez proche de celui pratiqué dans les entreprises utilisatrices, à savoir une médiane à 45 674 euros, un quart inférieur à 41 364 euros et un quart supérieur à 49 762 euros. ■



L'Ecole des Mines de Nantes

Créée il y a seulement dix ans, l'Ecole des Mines de Nantes a axé son projet pédagogique sur la prise en compte des besoins des entreprises.

L'informatique et l'apprentissage de la programmation sont présents tout au long des quatre années d'études : dès la deuxième année, programmation par objet, java, base de données sont des modules obligatoires, indique Philippe Cointe, qui dirige le département informatique. En troisième année sont abordés la programmation logique ou C++. Le département informatique a deux domaines de compétence forts, souligne-t-il : la programmation objets et la programmation par contrainte. Ce département compte 40 personnes, dont 20 permanents et des intervenants de sociétés comme Bouygues ou Ilog. Parmi les outils utilisés lors de la formation : environnement Eclipse d'IBM pour Java, UML (Rational Rose), le EJB d Sun, .net.

Christian Lardinois, directeur des études, souligne la "volonté d'innover en matière de pédagogie" avec un credo : "l'appren-



tissage par l'action". Un des éléments forts de cette approche est le dispositif PPAC (projet professionnel acquis et compétences) qui permet durant les trois premières années d'être progressivement accompagné et "d'élaborer son propre cheminement de formation". Outre l'enseignement technique, l'accent est mis sur la compréhension de l'environnement socio-économique de l'entreprise.



Ouverture à l'entreprise et à l'international

La formation est riche en expériences de terrain. En troisième année, un stage de trois mois doit obligatoirement être effectué à l'étranger. L'école projette d'ailleurs d'effectuer ce stage en six mois et certains élèves choisissent de passer l'intégralité de leur troisième année à l'étranger. En dernière année, un projet qui s'étend sur deux mois et demi est réalisé en partenariat avec une école étrangère. Les élèves travaillent ensemble à distance, puis sont réunis une semaine pour travailler ensemble. Le projet de fin d'études est mené lors d'un stage de six mois en entreprise. ■

CP

Admission : sur concours, à Bac +1 (math sup). Le concours est commun aux différentes écoles des Mines ;

Effectif : 150 élèves par promotion ;

Déroulement des études : deux années de tronc commun, puis une année un peu à la carte, pour s'orienter vers l'une ou l'autre des neuf options proposées en quatrième année. Parmi elles, deux dépendent du département informatique et sont très axées sur la programmation : génie des systèmes informatiques, orienté "architecture système et assemblage de composants", explique Pierre Cointe. L'autre est intitulée "Génie informatique pour aide à la décision", plus orienté vers la programmation par contraintes. Trois autres options concernent davantage les applications : "automatique et informatique industrielle", "gestion des opérations en production et logistique" et enfin "organisation et management des technologies de l'information".

Pour des informations supplémentaires : www.emn.fr

L'offshore programming

Le développement à prix cassé

PAR ANNIE LICHTNER



Karel de Gendre, www.karephoto.com

Pour faire face aux restrictions budgétaires, les entreprises font de plus en plus appel à des SSII spécialisées dans l'offshore. Qui sont elles ? Quelles méthodologies utilisent-elles ? Sur quels langages et outils s'appuient-elles pour leurs développements ? Comment gèrent-elles les exigences de leurs clients ? Voyage au pays du code Made in " India, Russia ou China... "

L'offshore programming est le terme le plus utilisé pour décrire le monde de l'outsourcing de la prestation informatique dans les pays étrangers où les niveaux de salaire sont faibles. Les pays en voie de développement : ex pays de l'Est (Ukraine, Roumanie, Moldavie), Afrique du Nord et plus récemment la Chine, dont les salaires sont très bas, proposent des prestations informatiques. Mais l'Inde et la Russie restent les deux principaux foyers pour l'offshore. Les raisons en sont simples : les prestations informatiques relèvent d'un bon niveau scientifique et de professionnels hautement qualifiés. La Russie dispose historiquement d'un bassin d'experts et l'Inde a su exporter ses

étudiants et ses compétences dans les pays développés. De fait, l'Inde a acquis des processus de management de projet à l'Occidentale, et n'a pas de mal à vendre, aujourd'hui, ses équipes d'informaticiens. A titre indicatif, un ingénieur français est rémunéré sur une base moyenne de 300 \$ la journée, alors qu'un ingénieur russe coûte environ 180 \$ et un ingénieur indien revient à 80 \$ la journée. Prestation séduisante, l'offshore est donc capable d'offrir une flexibilité dans les projets, tout en réduisant les charges fixes de l'entreprise. Dans les entreprises françaises, l'heure est en effet, aux restrictions budgétaires. Qu'il s'agisse d'une PME ou d'un grand-compte, les entreprises doivent relever un

D
O
S
S
I
E
R



triple défi : choisir le meilleur projet, en terme de qualité, de délai et de coût. En sous-traitant une partie des travaux, ou l'ensemble du projet à des développeurs étrangers, elles réduisent les coûts, tout en bénéficiant de compétences équivalentes, voire supérieures aux ingénieurs français. Selon le Gartner Group, une entreprise qui sous-traiterait la réalisation de son projet économiserait entre 30 et 40 %.

Un marché encore frileux en France

Toutefois cette évidence n'est pas de mise en France, car les développements offshore ne représentent que 5% des projets. Encore attachés au système de la régie (80 % des contrats de sous-traitance de projet informatique contre 20 % pour le forfait) la France est loin derrière les Etats-Unis, qui utilisent l'offshore dans 58 % de leurs projets, selon une étude menée par la société Tubbydev, SSII spécialisée dans l'offshore. A cela, une raison : la législation du travail américaine permet de louer

les services de développeurs étrangers, pour les faire venir au sein de l'entreprise cliente, en les rémunérant aux conditions de leur pays d'origine. Cette pratique est interdite en France où la législation et le code du travail s'appliquent à tous les travailleurs immigrés. Mais ce n'est pas la seule raison. L'offshore présente aussi des risques. Il y a bien sûr les barrières de langues et de culture mais aussi les problèmes liés aux infrastructures et les systèmes de communication, qui ne sont pas toujours fiables dans ces pays. Il existe d'autres désavantages qui sont fonction du pays avec lequel on travaille : problème du décalage horaire, des normes en matière d'import-export, de la taxation, de la sécurité de la

prestation, ainsi que du suivi des développeurs. Plus préoccupant, l'aspect juridique. Il faut savoir, par exemple, que dans les pays comme la Russie, la propriété intellectuelle du code réalisé n'est pas toujours protégée. Choisir son prestataire peut constituer dans ces conditions un exercice de voltige de haut vol. Reste que l'offshore est une approche que les développeurs français ne doivent pas négliger, même si le marché reste encore timide. Bien connaître les processus et les méthodologies de ces SSII, permet de mieux appréhender l'évolution du développement d'applications.

Les différents modèles d'offshore

Le marché de l'offshore se décline en trois grandes catégories de prestataires. Tout d'abord, les SSII étrangères qui travaillent en direct avec les entreprises (les Russes : V6 Technologies, les roumains Soft Origin, etc). Elles disposent d'un bureau à Paris, ce qui facilite la relation pour la prestation. La seconde catégorie concerne les SSII qui jouent le rôle de relais vers des plates-formes offshore, qui peuvent leur appartenir ou pas. C'est le cas de Tubbydev, société française qui dispose de sa propre plate-forme d'offshore à Kiev, ou encore de Hampi, cabinet de conseil français qui a choisi de travailler avec les principales SSII indiennes. La troisième catégorie concerne les SSII françaises, qui ont noué des relations avec des SSII étrangères et vont même jusqu'à posséder leur propre centre de développement à l'étranger. Cette catégorie travaille sur des gros projets destinés essentiellement aux Grands-Comptes. C'est le cas de Valtech, Cap Gemini ou encore Atos Origin. Les prestations effectuées sont transparentes pour les clients, les SSII françaises restent discrètes à ce sujet. Elles estiment pour la plupart, qu'il s'agit là de leur cuisine interne et l'important pour le client est de livrer le projet fini. Mais attention, si chacune de ces catégories de SSII dispose de solides compétences en informatique, il existe des différences sur les méthodes de travail et de management. Pour les gros projets, la méthode RUP (Rational Unified Process), élaborée par Rational, reste en tête pour gérer les développements avec toute la rigueur nécessaire. Rappel sur cette méthodologie basée sur les processus d'itération. ■

**Russie
Inde
Chine**
878.000
développeurs
(en 2001)

1. Etats-Unis	2.318.000
2. Japon	460.000
3. Russie	361.000
4. Canada	331.000
5. Inde	304.000
6. Allemagne	304.000
7. France	269.000
8. Gr-Bretagne	264.000
9. Chine	213.000
10. Italie	200.000

+70% d'ici 2005!
À noter que, selon IDC, le nombre de développeurs dans le monde passerait de 7,8 millions en 2001 à 13,3 millions en 2005, soit une croissance de 70% !
source : IDC

1 • RUP : LA MÉTHODE LA PLUS UTILISÉE POUR LES GROS PROJETS

Au cours de 20 années d'échanges avec des clients impliqués dans des projets de développement de logiciels complexes, Rational a identifié une série de principes clés, étroitement liés à la réussite d'un projet. Les pratiques de base ont évolué avec le temps, à travers la collaboration de milliers de clients. Ce savoir-faire a pris corps avec Rational Unified Process (RUP), une base de connaissances ouverte, quelle que soit la plate-forme adoptée : J2EE, .Net, etc. Objectif : affecter efficacement les ressources, livrer les artefacts et respecter les délais.

"Les processus unifiés modélisés par RUP permettent non seulement de gérer avec rigueur les développements au forfait, par des équipes délocalisées mais aussi, grâce aux processus d'itération, d'impliquer l'utilisateur final dans toutes les étapes du développement et ainsi de lui permettre de modifier certaines de ses spécifications en cours de projet" explique Nasser Kettani, directeur marketing de Rational. Un développement géré par RUP se décompose en quatre étapes longitudinales : Inception (pré-étude), Elaboration (préparation), Construction (développement), Transition (déploiement vers le client). (cf schéma)

Ces quatre étapes subdivisent ainsi le projet et possèdent elles mêmes des sous-étapes, qui prennent en compte les itérations entre les différents acteurs du projet.

Chacune de ces étapes peut se décomposer en six parties verticales : Etude du métier, Cas d'utilisation, Analyse, Implémentation, Tests et Déploiement.

"L'originalité du cadre de développement RUP est de s'appuyer sur une approche itérative, centrée sur les cas d'utilisation et sur l'architecture" précise Nasser Kettani. Le premier aspect (méthode et développement itérative) fait que les parties verticales sont en permanence revues en cours des itérations. Le cahier des charges, le plan de qualité, et tous les autres documents ou modèles que produisent les développeurs, ou qui les guident au long du travail, ne sont pas des documents figés, mais évoluent tout au long du développement. Au même titre que le code, ils font partie des livrables du projet.

Cette méthodologie permet d'affecter à chaque collaborateur une tâche bien définie, par exemple : test, programming, design et permet de connaître en temps réel l'avancement de chacune des tâches. La société d'offshore peut ainsi interagir même à distance sur le projet pas à pas. "Cette méthodologie de développement rend possible une adaptation aux contraintes liées à la localisation de nos développeurs en Russie. Nous contrôlons en temps réel, l'avancement de leurs travaux, par l'intermédiaire de notre Intranet et nous pouvons à tout moment effectuer des corrections ou des modifications" remarque Pierre Mechenet. Selon Rational, RUP satisfait pleinement aux cri-

D
O
S
S
I
E
R

RUP-F : Présentation

Méthodes	Phases			
	Inception	Elaboration	Construction	Transition
Expression des exigences	[Graphique]			
Analyse	[Graphique]			
Implémentation	[Graphique]			
Tests	[Graphique]			
Validation	[Graphique]			
Création de projet	[Graphique]			
Gestion des processus	[Graphique]			

Iterations: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

Processus d'ingénierie du logiciel

- Il s'agit d'une adaptation francophone du RUP de Rational. RUP-F s'appuie sur la sémantique architecturale et établit des filigranes du RUP, mais permet de combiner simplifications et rationalité (voir "Pour en savoir plus").
- Le processus est un ensemble d'étapes partiellement ordonnées dont l'exécution vise à atteindre un objectif, dans le domaine de l'ingénierie du logiciel cet objectif est la réalisation d'un produit logiciel ou sa maintenance.
- Pour améliorer la qualité du produit final, il est nécessaire d'adopter un processus qui soit aussi de qualité.
- Pour cela, le processus permet d'appuyer sur les principes suivants :
 - Il s'agit d'une approche "disciplinée" pour définir les travaux à effectuer, les produits à réaliser et définir les responsabilités à l'intérieur de l'organisation de développement.
 - Il s'appuie sur les meilleures pratiques disponibles dans l'ingénierie du logiciel.
 - Il inclut la maintenance. Le produit logiciel est développé pour la première fois, le processus est ensuite de tout ou partiel de la définition des exigences. Mais

Les best practices du développement logiciel

selon la méthode RUP (Rational Unified Process)

- Développement itératif pour identifier et éliminer les risques avant qu'ils ne menacent le projet.
- Gestion des exigences pour garantir l'adaptation face à des changements inévitables.
- Utilisation d'architectures à base de composants, pour rendre l'architecture compréhensible et partagée par tous les intervenants.
- Modélisation visuelle pour obtenir et préserver une architecture de qualité.
- Vérification continue de la qualité pour s'assurer de celle-ci, à toutes les étapes clés du cycle de vie du développement.
- Gestion des changements pour permettre un développement en parallèle efficace, au sein des équipes et dans toute l'entreprise.

tères préconisés par United Modeling Language (UML) en ce qui concerne la méthode qui l'accompagne à savoir :

- pilotée par le cas d'utilisation, c'est à dire que la SSII travaille en fonction des attentes de son client
- centrée sur l'architecture système
- itérative et incrémentale.

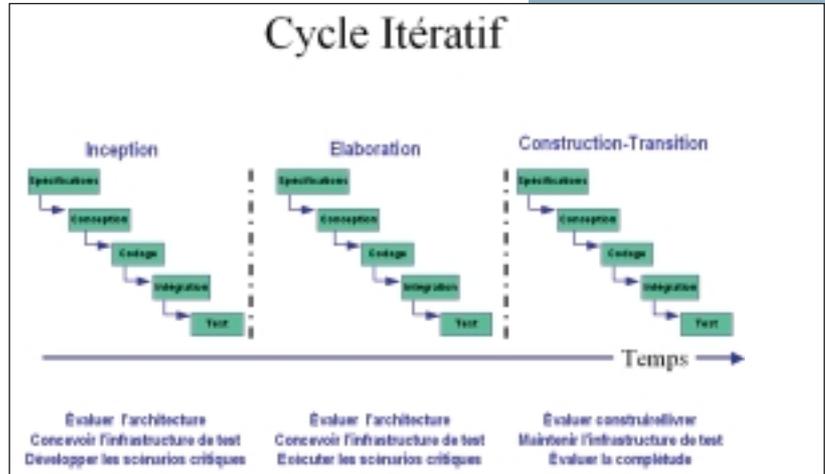
Pour bien comprendre ce principe, il suffit de voir les deux méthodes possibles :

Avec la méthode en cascade, les tests sont réalisés à la fin et en cas de problème, les développeurs sont obligés de revenir au début. L'évolution du système est plus difficile.

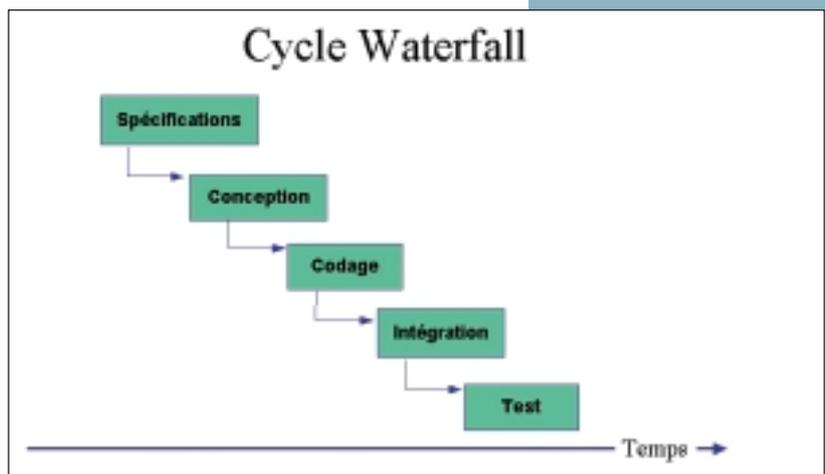
La méthode en itération prend en compte chaque sous-étape, les tests et les attentes du client, à chaque étape, les développeurs contrôlent qu'il est possible de passer à l'étape suivante.

Pour des projets de petite envergure, Rational a mis au point la méthode RUP for small Projects. Cette méthode reprend l'ensemble des process de développement présenté ci-dessus, mais elle est moins lourde en terme d'organisation et de modules à mettre en place. Il s'agit d'évaluer les différents points clés afin de préparer les principales étapes et artefacts qui permettront de maîtriser les risques de ce dernier.

Chez Tubbydev, la SSII n'utilise pas RUP en tant que produit commercial, mais plus comme cadre de développement, c'est à dire une méthodologie permettant de réaliser les six " best practices ", à savoir : développement itératif, gestion des exigences, architecture à base de composants, modélisation visuelle, contrôle de la qualité du développement, contrôle des impacts de changement. ■



Méthode en Itération



Méthode en Cascade

Définitions

La Régie :

Prestation qui consiste à louer les compétences d'un informaticien auprès d'une SSII pour une durée déterminée et à un tarif défini. La SSII a une obligation de moyen et elle est garante des compétences de ses informaticiens.

Le Forfait :

Il s'appuie sur un cahier des charges, la SSII fait une proposition commerciale avec un prix forfaitaire et un délai de livraison / réalisation. La SSII a une obligation de résultat et elle décide des moyens mis en œuvre

L'Externalisation

Baptisé aussi outsourcing, infogérance, consiste en une prise en charge totale ou partielle d'un processus de l'entreprise par une entité juridique distincte.

L'offshore

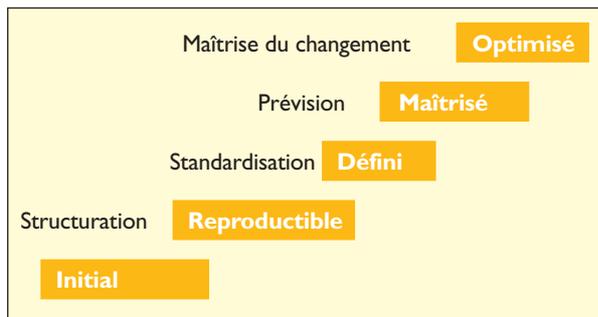
Il existe deux types de prestation offshore : l'offshore au forfait et l'offshore en régie délocalisée, baptisé aussi onshore. Dans le premier cas, les informaticiens sont des étrangers, qui travaillent au forfait par l'intermédiaire ou non d'une antenne locale. L'onshore ou body shopping, consiste de son côté, à louer des services de développeurs étrangers pour les faire venir au sein de l'entreprise en les rémunérant aux conditions de leurs pays d'origine. Pratique courante aux Etats Unis, elle est interdite en France où la législation et le code du travail s'appliquent à tous les travailleurs immigrés.

2 • CMM : LA CERTIFICATION QUI RASSURE

Pour s'imposer, les SSII spécialisées dans l'offshore doivent rassurer leurs clients en leur présentant une organisation et des méthodologies de développement adaptées. Pour aller plus loin, certaines d'entre elles proposent la certification CMM (Capability Maturity Model).

Des géants de l'offshore tels que Infosys en Inde, sous-traitant d'industriels français comme Airbus ou Valéo, misent sur la certification CMM. La certification CMM intègre 300 pratiques clés, qui précisent ce qui est à faire. Mise au point aux US par le Software Engineering Institute, elle est utilisée dans le monde entier, afin d'améliorer la façon de manager, de développer et de maintenir le logiciel. "Cette certification est le vrai enjeu qui va pousser toutes les sociétés offshore à monter en qualité et permet de juger le niveau de maturité de l'entreprise" remarque Jean-Yves Hardy, PDG de Valtech. Issu de l'expérience, le CMM est un modèle, décrivant quelles sont les pratiques à mettre en œuvre dans une organisation, afin d'améliorer et d'assurer la maîtrise des coûts. Le modèle est constitué de cinq niveaux de maturité : 1-Initial, 2-Reproductible, 3-Défini, 4-Maîtrisé et 5-Optimisé (cf Schéma 3)

La Structure du modèle CMM



Le niveau 1 "Initial" prend en compte le processus de développement qui n'est pas encore défini. La réussite du projet dépend du savoir-faire de quelques personnes clés dans l'organisation.

Le niveau 2 "Reproductible" concerne les entrées et sorties des activités logicielles. Elles sont gérées et contrôlées. Les règles sont connues et appliquées par les équipes. Le développement est planifié et suivi. Les produits sont vérifiés par rapport aux exigences initiales.

Le niveau 3 "Défini" les meilleures pratiques mises en œuvre au niveau 2 sur le projet.

Le niveau 4 "Maîtrisé" : concerne chaque processus qui est systématiquement mesuré. Les données sont consolidées et exploitées pour la prise de décision et l'anticipation des risques.

Le Niveau 5 "Optimisation" : les processus sont totalement maîtrisés et optimisés en permanence. Toute évolution est anticipée et gérée.

Si CMM rassure les clients, selon Hervé Six de chez Hampi, cabinet de conseil qui travaille avec des SSII indiennes, "Il faut rester pragmatique par rapport à tout cela. Pour tester le sérieux de la SSII, sa fiabilité dans les process et les méthodes, il suffit de lancer des pilotes sur des petits projets, avant de se lancer sur des projets de grande envergure."

Les outils : Clearquest et Clearcase en tête de proue

Le développement d'une application est un processus extrêmement complexe, compte tenu de la multiplicité des plates-formes, des protocoles, voire des équipes de développement décentralisées. Pour rester compétitives les SSII cherchent à intégrer les nouvelles technologies et les plates-formes les plus récentes, le tout sans ralentir la productivité. Elles ont tout intérêt à automatiser la gestion du développement d'un logiciel de manière efficace, afin de disposer d'un processus de conception performant. Dans ce contexte, la gestion de configuration logicielle (SCM : Software Configuration management) devient un outil de productivité. Cinq acteurs dominent le marché : Rational (racheté depuis peu par IBM), Merant, Computer Associates, Seran et Telelogic. En tête, Rational avec Clearcase (outil de gestion des artefacts logiciels) et Clearquest (outil de suivi de changement et des défauts). "Clearcase permet d'accélérer les rythmes de sortie grâce au développement parallèle, d'unifier les procédés de changement pendant le cycle de vie du développement, et de travailler en petites équipes, sans changer d'outil ou de procédé" constate Pierre Méchental. Mais surtout, l'outil de Rational permet un réglage unique avant le déploiement sur différents postes et des déploiements synchronisés, à partir de sites géographiquement distants. La force de Rational par rapport à ses concurrents ? Une offre intégrée avec des suites comme Rational Team Test. Cette solution permet de tester les applications les plus demandées en offshore, à savoir sur le Web, les ERP et les applications client / serveur dans les environnements Windows. TeamTest est construit autour d'un repository, basé sur une technologie serveur et qui est utilisé pour collecter toutes les informations nécessaires au bon dérou-

Les outils de configuration logicielle adaptés à l'offshore

Outils	Editeurs
Clearquest	Rational
Clearcase	Rational
PVCS	Merant
CM Synergy	Telelogic

D
O
S
S
I
E
R

lement du processus de tests. Au cœur de Team Test, il y a Rational Robot, un outil de test fonctionnel, que la société Rational présente comme une innovation technologique dans le domaine des tests sur objets. Ce dernier permet de réaliser des tests fonctionnels dans des environnements multiples : VB, Visual C++, Java HTML, Oracle Developer 2000, ainsi que les environnements ERP de SAP, People Soft et autres. Si Rational reste leader en matière de tests logiciels chez les SSII spécialisées dans l'offshore, quelques unes d'entre elles utilisent son concurrent : PVCS développé par Merant. PVCS Professionnel, PVCS Tracker et PVCS Configuration Builder forment une suite intégrée pour la gestion de la configuration logicielle. PVCS Plus ajoute un accès Internet, permettant aux équipes de collaborer en partageant des archives protégées et centralisées. La suite de Merant, comme celle de Rational, suit et indique les problèmes en cours aux différentes équipes, automatise les fabrications logicielles et assure le succès des développements standardisés et répliquables. Autre solution prisée par les SSII d'offshore : CM Synergy de Telelogic. Cette solution propose un référentiel distribué flexible, ainsi qu'une approche workflow. Elle propose, entre autres, le développement distribué par des équipes éloignées les unes des autres, un référentiel adapté au volume d'activité, l'intégration possible aux environnements de développements multiples, un service de migration détaillé etc. ■

Annie Lichtner

Les technologies utilisées en offshore, par ordre de priorité

Environnement et système

Windows NT
Unix
Linux (RedHat et Mandrake)

Langages

Delphi
PowerBuilder
WebObjects
Visual Basic
PERL
C++, Visual ++
Java
Java Script
PL/ SQL
HTML

Méthodes

RUP
Merise

Technologies Internet/ Intranet

Outils : SMTP, POP3, FTP, HTTP, FW I
XML
ASP
PHP
EJB
Visual Age for Java

Pour en savoir plus

Quelques SSII spécialisées dans l'offshore

<http://www.tubbydev.com>
<http://www.hampi-offshore.com>
<http://www.V6france.com>
<http://www.infosys.com>
<http://www.softorigin.com>

3 • TROIS EXEMPLES DE RÉALISATION EN OFFSHORE

L'offshore reste un sujet tabou dans les entreprises et bien peu souhaitent communiquer là dessus. Néanmoins, trois d'entre elles ont bien voulu nous livrer leur expérience avec les équipes de développement Russes.

MOBILITIS

Un site Web pour moins de 200 KF

PME spécialisée dans le conseil immobilier pour les entreprises, Mobilitis a fait réaliser son site Web en Russie. Un projet stratégique mené tambour battant par un entrepreneur qui sait gérer ses risques.

Pierre André d'Ornano, président de Mobilitis, est un entrepreneur atypique. L'offshore n'est pas chez lui un sujet tabou, bien au contraire ! " On ne peut pas avoir la prétention de faire partie des NTIC et avoir peur de faire développer son site au bout du monde. C'est absolument incompatible à mon point de vue " déclare-t-il sans détour. En février 2000, il

créait sa société de conseil en immobilier pour les entreprises. A l'époque, les nouvelles technologies explosent. " Après une première version de notre site Web qui ne nous satisfaisait pas, nous souhaitons aller plus loin, en intégrant de nouvelles fonctionnalités, comme la possibilité de générer automatiquement des appels d'offres, accéder à une application de gestion de projet en ligne, etc. Une refonte du site qui imposait des développements longs et coûteux " explique Pierre André d'Ornano.

Réalisée par une Web Agency, Mobilitis a du déboursier près de 800 KF pour cette première version. " Un investissement important pour une PME telle que la notre. Il n'était pas envisageable d'investir autant pour intégrer les nouvelles fonctionnalités " remarque son président. L'entreprise mène alors une réflexion sur la possi-



Pierre André d'Ornano
Président de Mobilitis

bilité d'externaliser le développement à l'étranger. " Nous avons choisi la SSII Tubbydev pour son sérieux et sa rigueur. Cette SSII travaille beaucoup avec la Russie, connue pour ses compétences en informatique. En outre, Tubbydev possède des ingénieurs en France et en cas de problème, le projet peut être rapatrié " ajoute Pierre André d'Ornano. En moins de quatre mois, le site est livré dans sa nouvelle mouture pour moins de 200 KF. " Les délais et les coûts ont été maîtrisés, malgré la complexité de notre cahier des charges " souligne le président. La prestation a été réalisée

au forfait, ce qui implique de bien définir le cahier des charges. " Le système est très souple. Nous avons travaillé avec un seul interlocuteur chez Tubbydev, qui répercute les informations aux développeurs en Russie. A ce titre, la faculté du chef de projet à faire exécuter les directives est d'autant plus grande qu'il n'est pas en contact direct avec les équipes. A mon sens cela avance plus vite, au contraire d'une Web Agency, qui dispose du personnel en régie. Elle doit gérer plusieurs projets en même temps. Résultat : la Web Agency peut se perdre dans des détails et s'éloigner des attentes de son client " estime Pierre André d'Ornano. Mais pour le président, la force d'une PME reste dans la prise de décision. " Dans une petite entreprise, les entrepreneurs savent gérer les risques, contrairement aux grandes sociétés, où la prise de décision est partagée entre les gestionnaires, les managers. C'est un atout indéniable pour rester compétitif ".

KYRIBA

Un site financier en ASP à mi-chemin entre l'onshore et l'offshore

Spécialisée dans la gestion de trésorerie pour les entreprises, Kyriba a mis en place un mode de commercialisation de ses services, basé en ASP (Application Service Provider). Un projet critique et complexe qui s'appuie sur deux sociétés offshore situées dans l'ex URSS. Kyriba est restée maître d'œuvre du projet, en sélectionnant les équipes et en imposant ses méthodes de travail.

Créée en février 2000, Kyriba est une société américaine qui a pour mission de gérer le cash des entreprises. " Le trésorier peut ainsi anticiper les mouvements, il reçoit des extraits de comptes, faire le rapprochement et prendre les bonnes déci-

sions pour investir ou emprunter " explique Eric O'Neill, CTO de Kyriba. Le business modèle de la société repose sur une offre en ASP, les entreprises clientes passent par une banque partenaire de Kyriba pour louer les services. Le projet est estimé à plusieurs dizaines d'années homme de développement, car précise Eric O'Neill, " Notre activité est complexe, à la fois au niveau fonctionnel mais aussi au niveau de la sécurité et de la technique. " Pour lancer l'activité, Kyriba décide de faire appel à l'offshore pour développer dans des délais courts, un

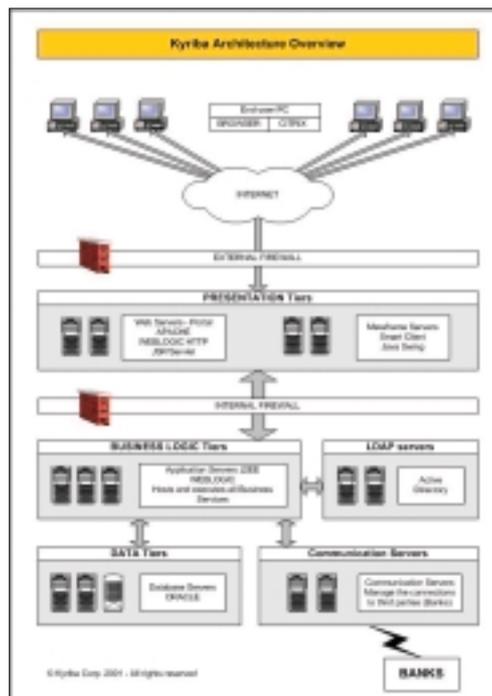


Kyriba, équipe Russe

an et demi environ, une première version du projet. " Début 2000, Internet explose, les salaires des développeurs en Java et EJB en France sont prohibitifs. Nous avons donc pris la décision de construire nous-mêmes, une équipe en offshore. " Partie de zéro, en quelques mois, Kyriba met en place une équipe d'une centaine de développeurs répartis sur deux sites : Kiev en Ukraine et Minsk en Biélorussie. " Nous avons choisi ces deux sites pour une raison essentiellement pragmatique. Travailler avec ces pays là, nous permettait d'être sur place en trois heures et d'éviter les décalages horaires, par rapport à l'Inde. On peut ainsi envisager des allers-retours sans problème et les salaires sont très bas " constate le CTO. Les deux SSII sont sélectionnées pour leur sérieux et leurs experts. " Nous avons demandé que leurs salaires soient supérieurs d'environ 20% aux salaires pratiqués localement, pour être sûr d'obtenir les meilleurs développeurs " remarque Eric O'Neill.

Pour Kyriba, il s'agit alors de tout construire sur place de A à Z. Première étape : mettre en place l'infrastructure, avec un réseau local dédié, pour éviter les risques d'intrusion. Seconde étape : créer une société à l'intérieur des SSII d'offshore en mettant en place une organisation spécifique. Exemple : isoler l'équipe du développement dédiée au projet de Kyriba, par rapport aux autres développeurs, pour éviter les risques d'échanges sur la partie sécurité et pouvoir mettre en place notre organisation, indépendamment de l'organisation présente

en offshore. " On a pris deux équipes sur deux pays, mais peu distantes géographiquement, pour se garantir en cas de problème. Depuis quelques mois, nous avons réduit les équipes à une seule, pour limiter les échanges et accroître l'efficacité. Les deux équipes étaient, chacune pleinement opérationnelles et nous n'avions plus besoin de couvrir nos risques en fonctionnant avec deux équipes " confie Eric O'Neill. Côté gestion des ressources, Kyriba a créé des petites équipes en France, à Minsk et à Kiev, avec un ratio d'une personne en France pour suivre environ 20 personnes à Kiev ou à Minsk " Notre souci était avant tout la méthodologie la plus adéquate. C'est un projet critique pour nous, il était hors de question d'arriver sans méthodologie et outils " Kyriba fait alors le choix Rational avec la méthodologie RUP et les outils Clearcase et Clearquest. " Clearcase est conçu pour travailler en offshore multi-site. Il nous permettait de répliquer du code source, de gérer le versionning et de voir en temps réel ce qui se passait. Clearquest de son côté, gérait la gestion du changement, permettait d'assurer la gestion des anomalies et de conserver la même vision sur les actions à mener sur le développement " constate Eric O'Neill. Pour aller plus loin, Kyriba s'est équipée de Tests Studio de Rational. Pour gérer avec beaucoup de détails les artefacts logiciels. " On a créé un plan qualité, qui nous a permis de savoir exactement ce qui était réalisé chaque jour, avec des tests automatisés de non régression et de performance pour chaque livraison ". Fin d'année 2000, une première couche des



mécanismes techniques et les spécifications sont rédigées par les développeurs. Le premier ensemble est livré en juillet 2001 et la réalisation finale en décembre 2001.

FIDELIPLUS

Répondre au time to market

L'activité de Fideliplus est centrée sur la fidélisation et l'incentive pour des partenaires de grandes sociétés. L'entreprise a fait appel à l'offshore pour le développement de son site, afin d'obtenir un niveau de service qualifié, à moindre coût et dans des délais courts.

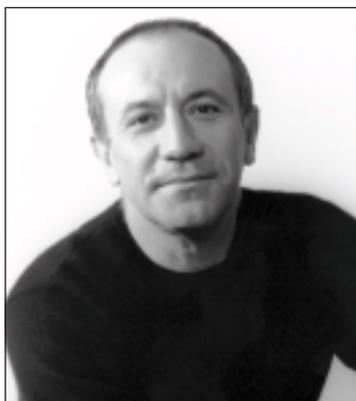
Filiiale du groupe DEXIA, la mission de Fideliplus consiste à mettre en place un système de fidélisation et d'incentive pour les partenaires, via le Net. " Nous avons fait appel dans un premier temps à des SSII classiques et en particulier à leurs ingénieurs qui ont travaillé en régie chez nous. Mais les coûts étaient tels, que très vite, nous nous sommes tournés vers l'offshore, pour bénéficier à la fois de compétences pointues et d'un tarif attractif " explique Bernard Corneau, président de Fideliplus. L'entreprise retient Tubbydev pour la réalisation de son site. " Nous avons établi un cahier des charges fonctionnelles, à partir d'exemples concrets, la SSII s'est chargée ensuite de le traduire en cahier des charges techniques " précise le président. La méthode retenue est RUP (Rational Unified Process). " Cette approche permet de définir 80 % du projet au départ. Elle laisse suffisamment de souplesse pour permettre au cours du développement, de le moduler en fonction d'idées nouvelles ". A cela s'ajoute aussi une interface de messagerie et de forum, pour communiquer en temps réel avec les développeurs. " Au bout de 8 mois, nous avons eu notre première version que nous avons affinée et nous avons apprécié à ce niveau la réactivité des développeurs pour nous satisfaire pleinement " constate Bernard Corneau. Le projet pourtant était jugé complexe par son aspect innovant. Restait à transférer les sources logicielles en France. " Une phase importante, aussi nous avons fait appel à un organisme externe (Logitas) pour récupérer toutes les sources logicielles, son mode d'emploi, afin de pouvoir le réinstaller facilement. Une étape qui s'est déroulée sans problème " conclut avec satisfaction Bernard Corneau. En résumé, le contexte économique pousse les grandes et les petites entreprises à diminuer leurs coûts de production, à l'instar des secteurs industriels du textile et de la chaussure. La question qui se pose aujourd'hui est de savoir si les pays occidentaux vont devenir des pays de maîtrise d'ouvrage, sous-traitant l'ensemble de leur production dans les territoires où la main d'œuvre est meilleur marché. Jusqu'à présent, on a longtemps pensé que cette mutation ne touchait pas le secteur tertiaire et plus particulièrement les secteurs technologiques. La tendance à l'offshore programming prouve le contraire, même si la France reste encore frileuse sur cette approche, mais peut être pas pour très longtemps. ■



Bernard Corneau, président de Fideliplus, avec Bernard Lerer

4 • EXTERNALISATION : ON RÉCOLTE CE QUE L'ON SÈME LE TÉMOIGNAGE DE ADAM KOLAWA, PARASOFT

La récession économique actuelle, en conjonction avec la peur de guerre et l'instabilité politique de certains pays, a sérieusement endommagé l'industrie infor-



matique mondiale, en particulier les secteurs logiciels en Europe et aux Etats-Unis. Afin de réduire les coûts et de rester bénéficiaires, les éditeurs de logiciels occidentaux ont commencé à externaliser leur développement sur des marchés à faible coût de main d'œuvre, à l'Est, plus

particulièrement en Chine et en Inde. Bien que profiter de ressources qualifiées et bon marché puisse sembler une alternative très intéressante aux salaires élevés des développeurs européens et américains, cette pratique est remplie de controverses et de pièges.

Les détracteurs de l'outsourcing craignent que les emplois ainsi externalisés ne retrouvent jamais leur pays d'origine, dégradant de façon permanente l'économie des états industrialisés de l'Ouest. Ceci est un problème sérieux, cependant, ma propre expérience m'a appris que la perte de ces emplois n'est rien en comparaison des dommages faits à notre industrie par la médiocre qualité des logiciels fournis par les prestataires étrangers.

Je délocalise une partie de notre programmation à la filiale polonaise de Parasoft depuis 1989. La main d'œuvre était très peu coûteuse en Pologne à cette époque, mais je ne me suis pas lancé dans cette entreprise uniquement pour réduire mes coûts à court terme. J'y suis allé, afin d'établir une relation à long terme avec un groupe de développement, qui pourrait fabriquer des produits de qualité. Les challenges étaient immenses. Tout ce que j'avais du faire aux Etats-Unis pour construire une équipe de développeurs solide, devait également être fait en Pologne : établir une culture d'entreprise forte et pro-

ductive, avec de bons procédés et des outils de prévention d'erreurs, qui permettent aux équipes de devenir adeptes des méthodologies et pratiques garantissant la fiabilité des produits.

Attendre 5 ans pour la qualité

Même avec un support important, il a fallu 5 longues années aux équipes polonaises pour apprendre à produire des logiciels de qualité efficacement. Rien ne marchait vraiment dans ce groupe, jusqu'à ce qu'ils apprennent à travailler comme une équipe éloignée des autres. Si je cherchais uniquement un moyen de réduire mes coûts immédiatement, alors cet arrangement aurait été complètement insensé. Il m'a fallu payer 5 ans de salaires sans réel retour. Il est certain que mes coûts étaient moins onéreux qu'ils ne l'auraient été aux Etats-Unis, mais j'étais toujours dans l'incapacité de capitaliser rapidement sur ce que produisait cette équipe. Cependant, cette solution était intéressante car mon équipe polonaise était une filiale, entière propriété de Parasoft, je cultivais des ressources dans lesquelles je pourrais puiser n'importe quand et dont les efforts étaient toujours concentrés sur le développement des produits Parasoft. Cette coopération était un investissement, une relation d'affaires à long terme, et j'ai du me montrer patient dans les premières années, sachant que mes efforts commenceraient probablement bientôt à me profiter. Cependant, pour la vraie délocalisation, où le travail est effectué par un fournisseur étranger, peut-on se permettre d'attendre 5 ans, avant qu'un code de qualité ne soit produit ? Les fournisseurs doivent apprendre à connaître vos produits, votre code, vos méthodes et comment travailler avec vous. Et cela prend du temps. Etes-vous prêts à travailler avec des prestataires sur le long terme ? Se sont-ils engagés à ne fournir que vous, ou s'agit-il d'une usine à code qui accepte de multiples projets ? Dans ce dernier cas, vous êtes garanti d'obtenir des produits de basse qualité, dénués de toute innovation. Ceci est le réel danger de la délocalisation, ce n'est pas l'expatriation des emplois mais les dommages consistants dont souffre l'industrie logicielle, à cause de produits de mauvaise qualité. La qualité des produits externalisés est-elle suffisamment élevée et consistante pour mériter les efforts nécessaires à déplacer des projets à l'étranger ? Pour certains secteurs la réponse est oui, mais pour le logiciel la réponse est un NON tonitruant.

Absence de standard pour créer un logiciel

Par le passé, des entreprises dans d'autres secteurs ont délocalisé leur production vers des pays à main d'œuvre bon marché, sans souffrir de défaillances de qualité. Cela est possible, car ces entreprises avaient des processus de fabrication bien définis, et avaient la possibilité d'intégrer ces personnes dans leurs procédés. La qualité était basée sur " comment était fait le produit ", et " qui l'avait fait ", n'avait aucun impact sur le produit final. L'industrie automobile en est le parfait exemple. Est-ce important que votre BMW soit faite à Berlin en Allemagne, ou à Spartanburg en Caroline du Sud ? Bien sûr que non. La raison pour laquelle l'industrie logicielle souffre tant de l'externalisation, est qu'il n'existe pas de procédés de développement communs à tout le secteur, pour créer un logiciel. La manière fortuite dont est créé un logiciel, fait que notre industrie est unique, les mêmes problèmes de qualité et de fonctionnalité qui infestent les magasins occidentaux vendant des logiciels, sont simplement exportés à l'étranger. A moins de prendre le temps d'installer de bonnes habitudes de travail, et du bon code chez un prestataire étranger, ou mieux même, construire une filiale dont vous avez le contrôle opérationnel complet, l'externalisation continuera à fournir les mêmes types de problèmes rencontrés chez nous. La main d'œuvre a beau être bon marché, les problèmes de qualité et de fonctionnalité persisteront.

Manifestement, l'externalisation ne peut être prise à la légère ou sans planification. Vous devez vous demander s'il existe des bénéfices substantiels à l'externalisation, à la fois sur le court et le long terme, au-delà du simple contrôle des coûts. Quand vous prenez en compte le coût réel, en ajoutant le coût de la qualité au tarif du prestataire étranger, vous trouverez alors que l'externalisation ne vous permet pas une économie, et que vous auriez eu tout intérêt à garder ce travail en interne. Pour moi, regarder au-delà des coûts du travail, démontre que très peu de bénéfices sont tirés, en ayant recours aux prestataires étrangers. Quand bien même l'outsourcing continuerait de recevoir l'approbation de l'industrie logicielle, un phénomène intéressant se produirait, et cela est quelque chose également tiré de mon expérience polonaise. Le personnel bon marché se qualifiant en écrivant du code, demande à ce que les salaires augmentent, accroissant ainsi les coûts de production. Un équilibre sera éventuellement atteint, où la productivité offshore et le coût du travail se stabiliseront. Dans cette situation, l'externalisation est-elle réellement moins coûteuse ? Pour ma part, l'externalisation est condamnée par

sa propre espérance. Au fur et à mesure que les employés qualifiés acquièrent du savoir, ils seront finalement tout aussi chers que sur le marché du travail occidental actuel.

Il existe un dernier problème à l'externalisation, qui n'a rien à voir avec le coût du travail. Le développement logiciel demande de la créativité, à bien des égards c'est plus un art qu'une science. Néanmoins, beaucoup de tâches externalisées vers l'Est, sont des travaux courants. Pour obtenir une réelle innovation de ces prestataires, vous devez être disposé

"La raison pour laquelle l'industrie logicielle souffre tant de l'externalisation est qu'il n'existe pas de procédés de développement communs à tout le secteur pour créer un logiciel. La manière fortuite dont est créé un logiciel, fait que notre industrie est unique"

à externaliser des tâches plus valorisantes, pas uniquement des travaux de programmation routiniers. Cependant, vous ne pouvez pas simplement vous décharger de tâches importantes, sur la main d'œuvre externalisée, aussi qualifiée soit-elle. Comme je l'ai dit précédemment, cela prend du temps pour obtenir une ressource performante et capable de produire du code de bonne qualité. Avec une filiale consacrée à mes besoins, cela a pris

5 ans. Un partenariat avec une usine à code, cela requiert un investissement encore plus important en temps, disons 10 ans, avant que les équipes de programmation ne commencent à restituer des produits réellement innovants, comme ceux produits par leurs homologues européens et américains.

80% des projets externalisés ne donneront pas de produits de bonne qualité

L'offshore est considéré comme une baguette magique par l'industrie logicielle, la réponse ultime au contrôle des coûts qui explosent, fragilisant l'industrie dans son ensemble, écrasant les compagnies les plus faibles et détruisant la rentabilité des géants de l'industrie. Cependant, de mon point de vue, au minimum 80% des projets externalisés ne donneront pas de produits de bonne qualité. Obtenir des produits de bon niveau de ces programmeurs requiert un investissement en temps, à l'opposé même de la notion d'externalisation du départ. L'outsourcing est un moyen de réduire les coûts dans l'immédiat. Ceci pris en compte, si la qualité des projets confiés à ces prestataires étrangers est mauvaise, êtes-vous réellement en train d'économiser ? Les coûts immédiats sont peut-être réduits mais, à long terme, les coûts de ces produits ne feront que croître. Si les projets externalisés ne sont pas liés à une filiale où l'investissement en temps et argent est cohérent sur le long terme, alors le recours à des prestataires offshore n'en vaut pas la peine. ■

Adam KOLAWA
Président de Parasoft.

DEMARRER PHP

(suite)

Nous avons vu le mois dernier, comment installer une plate-forme AMP (Apache Mysql PHP) sur votre poste.

Nous allons maintenant rentrer un peu plus dans le détail et vous présenter les bases de PHP.

Pour vos tests, installez vos fichiers *.php dans le répertoire racine de votre serveur (répertoire qui correspond à la directive DocumentRoot dans le fichier httpd.conf d'apache) et demandez à votre navigateur d'aller à l'adresse `http://localhost/*.php`. En France, plusieurs sites pourront vous aider lors de vos premiers pas : phpdebutant.org, phpfrance.com et phpapps.org.

PHP

NIVEAU : DÉBUTANT

Insertion de PHP dans HTML.

Le code PHP est directement immergé dans les fichiers HTML. Il peut figurer à différents endroits de ces fichiers, tout en étant entrecoupé de code HTML. Le début et la fin des portions de code PHP sont signalés grâce à des balises d'ouverture et de fermeture.

Variantes des balises d'ouverture et de fermeture PHP

<?	?>
<?php	?>
<%	%>
<script language="php">	</script>

Le style ASP (<% et %>) ne peut être utilisé que si vous avez défini la valeur On pour l'option asp_tags dans le fichier php.ini.

Les commentaires sont introduits par la séquence /* et se terminent par *. Par ailleurs, PHP utilise également les signes de commentaires // du langage C, qui ne doivent cependant être utilisés que pour de brefs commentaires sur une seule ligne.

Les commentaires

```
<?
/*
Commentaires sur plusieurs lignes
exemple
*/
// Commentaires sur une ligne
?>
```

Les commandes PHP doivent être placées entre les balises et se terminer par un ";".

Si vous l'oubliez vous verrez apparaître un PARSE ERROR lors de l'exécution de votre fichier.

Vous devez penser à nommer vos fichiers nom_du_fichier.php, donc avec l'extension php. Il est aussi possible de mettre .php3, mais cela n'est pas conseillé, car cette extension n'est plus utilisée depuis l'apparition de php4.

Dans un premier temps nous allons insérer du php dans des pages de format HTML.

Structure d'un document :

Afficher du texte avec la fonction echo()

```
<HTML>
<HEAD></HEAD>
<BODY>
<CENTER> HELLO ALL</CENTER><BR>
<?
echo ("ceci est du code PHP<BR>");
echo "simple non ? <BR>";
?>
</BODY>
</HTML>
```

Explications :

Pour indiquer au serveur que les lignes suivantes seront du PHP on écrit "< ?" et on ferme la balise pour indiquer au serveur que l'interprétation n'est plus nécessaire comme cela " ?> "

echo (" ceci est du code PHP
 ");

La commande echo indique au serveur qu'il doit renvoyer les

PRACTIQUE

informations contenues entre double quote. On remarquera qu'il existe plusieurs syntaxes possibles pour les echo (dans notre cas, avec ou sans parenthèses).

On notera aussi qu'il est possible de renvoyer du code HTML (
).

On peut aussi renvoyer le code HTML désignant une image :

```
< ?echo ("<img src= './image.gif'>");?>
```

Constantes, variables et Types de données.

Les variables en php se trouvent sous la forme \$nom_variable. Il n'est pas nécessaire de leur déclarer un type, car ceci est fait de façon automatique quand on lui assigne une valeur.

Les variables sont toujours signalées par le \$ qui précède leur nom.

Assignement de valeurs à des variables

```
< ?
$nom = " Kaptive " ;
$nombre = 10 ;
?>
```

Différents types de données de PHP

```
< ?
$a = 1 ;
// a est un entier.
$a = 1.2 ;
// a est un double.
$a = "hello" ;
// a est une chaîne.
$a[2] = "hello";
//a est un tableau et son 3e élément est "hello".
$a = true ;
//a est un booléen
?>
```

Opérateurs

Comme la plupart des langages, PHP dispose d'opérateurs dont voici une liste

Opérateurs arithmétiques (\$a=9;\$b=4;)				
Opérateur	Opération	Exemple	Description	Résultat
+	Addition	echo \$a + \$b;	Calcule la somme	13
-	Soustraction	echo \$a - \$b;	Calcule la différence	5
*	Multiplication	echo \$a * \$b;	Calcule le produit	36
/	Division	echo \$a / \$b;	Calcule la division	2.25
%	Modulo	echo \$a % \$b;	Calcule le modulo	1

Opérateurs de comparaison			
Opérateur	Signification	Exemple	Description
==	Egalité	\$h == \$i	Renvoie vrai si \$h est égal à \$i
<	Moins que	\$h < \$i	Renvoie vrai si \$h est plus petit que \$i
>	Plus que	\$h > \$i	Renvoie vrai si \$h est plus grand que \$i
<=	Moins ou égal	\$h <= \$i	Renvoie vrai si \$h est plus petit ou égal à \$i
>=	Plus ou égal	\$h >= \$i	Renvoie vrai si \$h est plus petit ou égal à \$i
!=	Différent	\$h != \$i	Renvoie vrai si \$h est différent de \$i

Erreur fréquente :

Ne pas confondre l'opérateur d'affectation "=" et l'opérateur de comparaison "==". Ceci entraîne souvent des boucles infinies (while (\$i=5)).

Opérateurs logiques		
Exemple	Nom de l'opérateur	Évalué à Vrai quand :
\$h && \$i	ET (And)	\$i et \$h sont true.
\$h \$i	OU (Or)	Au moins l'un des deux est vrai.
\$h and \$i	ET (And)	\$i et \$h sont true.
\$h or \$i	OU (Or)	Au moins l'un des deux est vrai.
\$h xor \$i	Ou exclusif	L'un des deux est vrai.
! \$h	NON (Not)	\$h ne renvoie pas vrai

La concaténation de valeurs

Opérateurs de Chaîne (\$a='un';\$b='texte');			
Opérateur	Opération	Exemple	Résultat
.	concaténation	echo \$a.\$b;	un texte
.=	assignation et concaténation	\$a .= 'ajout'; echo \$a;	un ajout

```
< ?
$a = " hello " ;
$a .= " world " ;
//$a est égal à " hello world " ;
?>
```

Les opérateurs d'incrément et de décrémentation

Les opérateurs d'incrément et de décrémentation de PHP permettent d'augmenter ou de baisser la valeur des variables de 1. La syntaxe de ces opérateurs est empruntée au langage C. Ces opérateurs peuvent figurer devant la variable (++\$a : la variable est incrémentée puis évaluée) ou derrière la variable (\$a++ : la variable est évaluée puis incrémentée).

Opérateurs d'incrémentations (\$a=9;)			
Opérateur	Opération	Exemple	Résultat
++	Incrément	echo \$a++.'#.\$a;	9#10
		echo ++\$a.'#.\$a;	10#10
--	Décrément	echo \$a--.'#.\$a;	9#8
		echo --\$a.'#.\$a;	8#8

L'AUTHENTIFICATION PAR FORMULAIRE SOUS ASP.NET

Après avoir traité l'authentification Windows, nous allons aborder ce mois-ci l'authentification par formulaire. Ce mode d'authentification consiste à rediriger tout utilisateur vers une page de login, tant que celui-ci ne s'est pas authentifié.

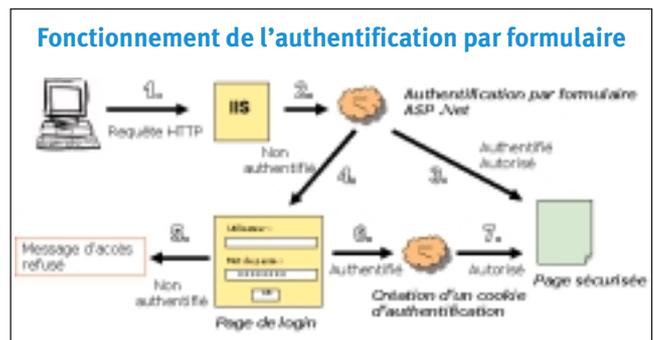
L'authentification peut-être réalisée soit à partir d'un fichier de configuration XML soit à partir d'un système externe (SGBD ou Active Directory).

ASP.NET

NIVEAU : INTERMEDIAIRE

Nous allons décrire la mise en œuvre du mécanisme d'authentification à base de formulaire. La liste des utilisateurs autorisés sera dans un premier temps décrite dans un fichier de configuration XML puis intégrée à une table des utilisateurs d'une base de données SQL2000. Regardons plus précisément le fonctionnement interne de l'authentification par formulaire.

Le mécanisme d'authentification à base de formulaire



Voici comment fonctionne sous ASP .Net l'authentification par formulaire :

- 1 ■ Un client soumet une requête http pour accéder à une page ASP sécurisée.
- 2 ■ IIS transmet la requête à ASP .Net pour authentification. Il faut cocher l'accès anonyme pour l'authentification IIS.
- 3 ■ ASP .Net vérifie si le client dispose d'un cookie d'authentification. Si l'utilisateur n'est pas authentifié alors il est redirigé vers une page de login.
- 4 ■ L'utilisateur saisit son identité et son mot de passe.
- 5 ■ Les informations d'authentification sont vérifiées (fichier de configuration, SGBD...). Si l'utilisateur n'est pas authentifié alors un message d'accès refusé est affiché.
- 6 ■ Les informations d'authentification ont été validées, un cookie d'authentification est généré.
- 7 ■ Si l'utilisateur est autorisé par ASP .Net alors il accède à la page demandée.

Passons à la pratique et mettons en place ce système d'authentification en déroulant les étapes suivantes :

- Paramétrage du fichier Web.Config
- Définition du mode d'authentification

- Définition de la page de redirection
- Définition de la liste des utilisateurs authentifiés avec cryptage de leur mot de passe
- Création d'un formulaire Web pour recueillir les informations d'identification du client
- Vérification des informations d'authentification du client à partir d'un fichier de configuration XML

Paramétrage du fichier Web.Config

Pour positionner l'authentification par formulaire, voici les modifications à réaliser dans le fichier Web.Config.

```
<authentication mode="Forms">
  <forms loginUrl="login.aspx" timeout="20">
    <credentials passwordFormat="MD5">
      <user name="christian" password="7FF135854376850E9711BD75CE942E07"/>
    </credentials>
  </forms>
</authentication>

<authorization>
  <deny users="?" />
</authorization>
```

Tout d'abord le mode d'authentification a été basculé à " Forms ". La balise *LoginUrl* indique la page vers laquelle l'utilisateur sera

systématiquement redirigé tant qu'il ne sera pas authentifié. La balise *timeout* indique la durée en minutes du cookie d'authentification. Les informations concernant les utilisateurs sont définies dans la balise *credentials*. Il faut ensuite définir pour chaque utilisateur un nom et un mot de passe au sein d'une balise *user*, seuls ces comptes seront habilités à s'authentifier sur notre application. Enfin, nous complétons la sécurité de notre application Web, en interdisant son accès à tous les utilisateurs non authentifiés.

Le cryptage du mot de passe :

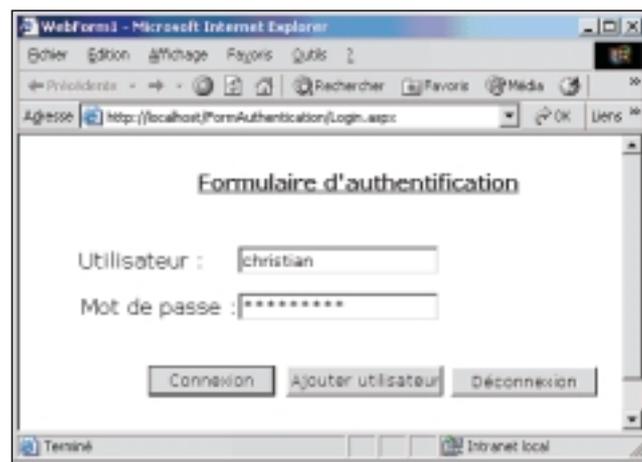
Mais me direz-vous, que signifie le format MD5 et comment crypter le mot de passe des utilisateurs ?

Le cryptage du mot de passe peut être réalisé grâce à la fonction **HashPasswordForStoringInConfigFile** de la classe **FormsAuthentication**. Elle permet de générer un mot de passe crypté selon un algorithme de hachage SHA1 ou MD5. Ces deux valeurs peuvent être utilisées comme format du mot de passe ainsi qu'une valeur "Clear" à utiliser si vous ne souhaitez pas le crypter.

Le formulaire d'authentification

Voici une copie de notre écran d'authentification.

Exemple de formulaire d'authentification



L'écran d'authentification est relativement simple avec une zone de saisie pour le nom de l'utilisateur, une zone de saisie pour le mot de passe et un bouton de connexion. Ne nous préoccupons pas pour l'instant des autres boutons.

Regardons maintenant le code exécuté lorsque l'utilisateur clique sur le bouton Connexion.

```
using System.Web.Security;
...

private void BtConnexion_Click(object sender, System.EventArgs e)
{
  if (FormsAuthentication.Authenticate(txtUtilisateur.Text,txtMotDePasse.Text))
    FormsAuthentication.RedirectFromLoginPage(txtUtilisateur.Text,false);
  Else lblMessage.Text = "Login incorrect !";
}
```

La classe **FormsAuthentication** fait partie de l'espace de noms **System.Web.Security**. Il faut donc l'inclure dans notre page.

La méthode **Authenticate** vérifie le nom d'utilisateur et son mot de passe à partir des éléments contenus dans la balise **Credentials** du fichier **Web.Config**. Si l'utilisateur est habilité, alors un cookie

d'authentification est généré et l'utilisateur est redirigé vers la page demandée initialement ; ces opérations sont réalisées par la méthode **RedirectFromLoginPage**. Cette méthode prend également un second paramètre qui indique si le cookie doit persister lors d'un changement de session.

Précisons que dans le cas où la méthode **RedirectFromLoginPage** ne peut pas identifier de page de retour alors la page **default.aspx** est affichée.

L'authentification des utilisateurs, via une base de données

Nous venons de mettre en place une authentification par formulaires, en validant l'identité de l'utilisateur à partir d'informations contenues dans un fichier de configuration XML. Cette solution fonctionne, mais peut devenir rapidement fastidieuse pour un administrateur chargé de saisir un grand nombre de comptes utilisateurs.

Essayons d'améliorer notre solution, en validant désormais les informations à partir d'une table " utilisateurs " d'une base de données SQL2000. Pour ce faire, nous devons :

- Modifier le fichier **Web.Config** en supprimant la balise **credentials**.
- Créer une table **SQL2000** qui contiendra les informations d'authentification.
- Développer notre propre fonction d'authentification " Authentifier " chargée de vérifier les informations saisies par l'utilisateur avec celles contenues en base.
- Modifier le code associé au bouton **Connexion** afin d'utiliser notre nouvelle méthode.

Description de la table " utilisateurs "

Cette table contient les deux champs suivants :

Champ	Type de données SQL2000	Description
nom	Varchar(20)	Nom de l'utilisateur
motdepasse	Varchar(50)	Mot de passe

Pour des raisons de sécurité, nous crypterons le mot de passe au format MD5.

Description de la fonction "Authentifier "

```
private bool Authentifier(string strUtilisateur, string strMotDePasse)
{
    bool bOk=false;

    // Cryptage du mot de passe
    strMotDePasse = FormsAuthentication.HashPasswordForStoringInConfigFile(
        strMotDePasse,"MD5");

    // Création d'une connexion SGBD
    SqlConnection oConnexion = new SqlConnection("user id=sa;
        password=:initial catalog=pubs;data source=pttravail");

    // Définition de la requête à exécuter
    SqlCommand oCommand = new SqlCommand("SELECT * FROM
        Utilisateurs WHERE nom='"+ strUtilisateur+ "'",oConnexion);

    try
    {
        // Ouverture de la connexion et exécution de la requête
        oConnexion.Open();
        SqlDataReader drUtilisateur = oCommand.ExecuteReader();
```

```
// Parcours de la liste des utilisateurs
while (drUtilisateur.Read())
{
    if (drUtilisateur["motdepasse"].ToString() == strMotDePasse)
    {
        bOk = true;
        break;
    }
}
catch
{
    bOk = false;
}
oConnexion.Close();
return bOk;
}
```

Nous cryptons d'abord le mot de passe saisi par l'utilisateur, puis nous sélectionnons la liste des mots de passe de l'utilisateur qui cherche à s'authentifier, enfin nous la parcourons, en comparant chaque mot de passe avec celui crypté précédemment. Dès qu'un mot de passe a été trouvé nous retournons **Vrai** à la fonction appelante.

Il ne nous reste plus qu'à modifier le code associé au bouton **Connexion** et à invoquer notre méthode **Authentifier**.

```
private void BtConnexion_Click(object sender, System.EventArgs e)
{
    if (Authentifier(txtUtilisateur.Text,txtMotDePasse.Text))
        FormsAuthentication.RedirectFromLoginPage(txtUtilisateur.Text,false);
    else
        lblMessage.Text = "Erreur d'authentification, l'utilisateur ou le mot de passe
            n'existent pas!";
}
```

La déconnexion de l'utilisateur

La plupart des sites, qui proposent un moyen de s'authentifier par formulaires, fournissent également un bouton de déconnexion. La méthode **SignOut()** de la classe **FormsAuthentication** réalise cette opération de déconnexion en détruisant le cookie d'authentification, ce qui obligera l'utilisateur à s'authentifier de nouveau pour accéder à l'une des pages de votre application.

Voici le code exécuté lorsque l'utilisateur clique sur le bouton **Déconnexion**.

```
private void BtDeconnexion_Click(object sender, System.EventArgs e)
{
    FormsAuthentication.SignOut();
    Response.Redirect("Login.aspx");
}
```

Nous détruisons le cookie d'authentification et redirigeons l'utilisateur vers la page de login.

Conclusion

Nous avons pu nous rendre compte de la facilité de mise en œuvre de l'authentification par formulaire avec **ASP .Net**. Il est ainsi possible en quelques lignes de code et à moindre coût de sécuriser toute application **Web Intranet** ou **Internet**. Le mois prochain nous terminerons notre série d'articles en traitant le mode d'authentification **Microsoft Passport Single Sign In**.

Christian PEYRUSSE

DEMYSTIFIER J2EE AVEC Jboss

(1^{ère} partie)

Je vous propose dans une série de trois articles une visite guidée dans la mise en œuvre de ce serveur d'applications et la réalisation de quelques composants J2EE : une JSP et un EJB de type Session Bean. Tout cela, dans un environnement de développement confortable, c'est à dire en utilisant un IDE ergonomique et disponible sans frais. Vous découvrirez à cette occasion, pourquoi Jboss est l'outil qu'il vous faut pour aborder J2EE.

J2EE

NIVEAU : DÉBUTANT

J2EE a la réputation d'être une norme riche, puissante mais complexe. Construire une application mariant Servlet, JSP et EJB serait affaire de spécialistes ; une aristocratie technique rompue à l'art des moniteurs transactionnels, des API Java et d'XML. Le propos de cet article est de vous montrer que tout ceci relève essentiellement de la légende.

En effet, tout développeur Java est capable de devenir rapidement un développeur J2EE compétent. Les API qui constituent J2EE sont à la fois peu nombreuses, très logiques, très cohérentes et simples, à l'exception d'un petit nombre d'entre elles.

Le problème n'est pas J2EE, mais le traitement que bien souvent les fournisseurs de serveurs d'applications en ont fait. Il est donc démontré que l'on peut faire compliqué à partir d'une norme simple. Mais on peut aussi faire simple : c'est justement le tour de force réalisé par une équipe internationale de développeurs talentueux. Le résultat est éloquent : le serveur d'applications Jboss, dans sa version 3 est probablement la réalisation récente du monde J2EE la plus significative. Et c'est gratuit!

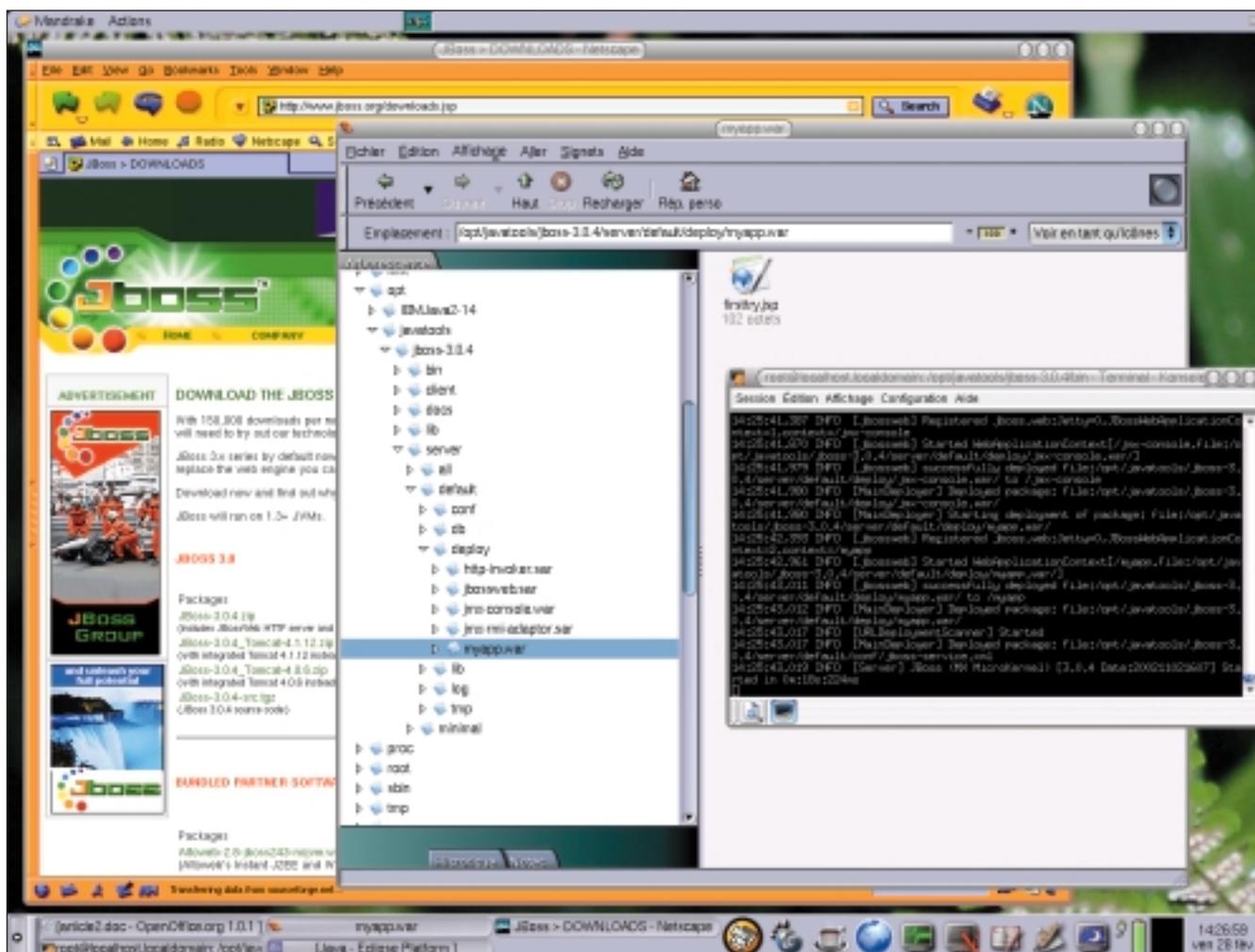
Je vous propose dans une série de trois articles une visite guidée dans la mise en œuvre de ce serveur d'applications et la réalisation de quelques composants J2EE : une JSP et un EJB de type Session Bean. Tout cela dans un environnement de développement confortable, c'est à dire en utilisant un IDE ergonomique et disponible sans frais. Vous découvrirez à cette occasion, pourquoi Jboss est l'outil qu'il vous faut pour aborder J2EE.

Installation et découverte

La première étape de notre démonstration est l'acquisition du serveur Jboss lui même. Inutile de sortir votre chéquier ou votre carte bleue : le produit est disponible sur le CD-ROM joint à ce magazine (J2EE\jboss3.0.4.zip). Si vous disposez d'un accès Internet confortable, vous pouvez aussi télécharger les versions les plus récentes au fur et à mesure de leur apparition, à partir du site de Jboss (www.jboss.org/downloads.jsp). C'est complètement gratuit pour un usage non limité. Ce fichier a une taille raisonnable: 28-32 Mo. Jboss est un serveur d'applications complet. Il intègre donc un serveur Web dynamique qui peut être soit Jetty, soit Tomcat. Vu de l'extérieur, cela n'a aucune importance : quel que soit votre choix, vous aurez affaire à Jboss. L'utilisation de Jetty ou de Tomcat est transparente.

Maintenant que nous disposons de Jboss-3.0.4.zip, il faut vérifier que nous avons installé une version récente du JDK. La version 1.4.x s'impose aujourd'hui. Pour les utilisateurs de Windows je conseille le JDK 1.4.1 de SUN (disponible à partir du site <http://java.sun.com/j2se/1.4.1/download.html>). Ceux qui préfèrent Linux opteront probablement pour le JDK 1.4.0 d'IBM (disponible à partir du site <http://www6.software.ibm.com/dl/lxdk/lxdk-p>) qui est sensiblement plus performant que celui proposée par SUN sur cette plate-forme. Ces deux distributions de JDK sont fournies sur le CD-ROM.

Il est temps de procéder à l'installation de Jboss. Pour cela, nous allons extraire le contenu de JBoss-3.0.4.zip et le placer quelque



Arborescence de JBoss, intégrant l'application "Myapp.war". Le serveur JBoss est lancé, comme en témoigne la fenêtre Terminal au 1er plan.

part dans l'arborescence de fichiers. Par exemple dans le répertoire `c:\javatools`. (windows) ou `/opt/javatools` (Linux). L'installation de Jboss est terminée. Oui, vous avez bien lu : *ter-mi-née*. Jboss est probablement le seul serveur d'applications que l'on peut installer sans s'en rendre compte !

Sans pour autant être une boutade, c'est une affirmation un peu rapide. Il faut en effet que le JDK soit correctement configuré pour que tout fonctionne bien. Jboss a en effet besoin que la variable d'environnement `JAVA_HOME` soit définie. Elle doit désigner le répertoire racine du JDK (exemple : `c:\java\jdk-1.4.1`). Une première solution est de définir cette variable d'environnement de manière globale. Sous Windows NT/2000 faire "Panneau de configuration > Système > Avancé > Variables d'environnement > Variables Système". Sous Linux il faut déclarer cette variable en rajoutant les lignes suivantes :

```

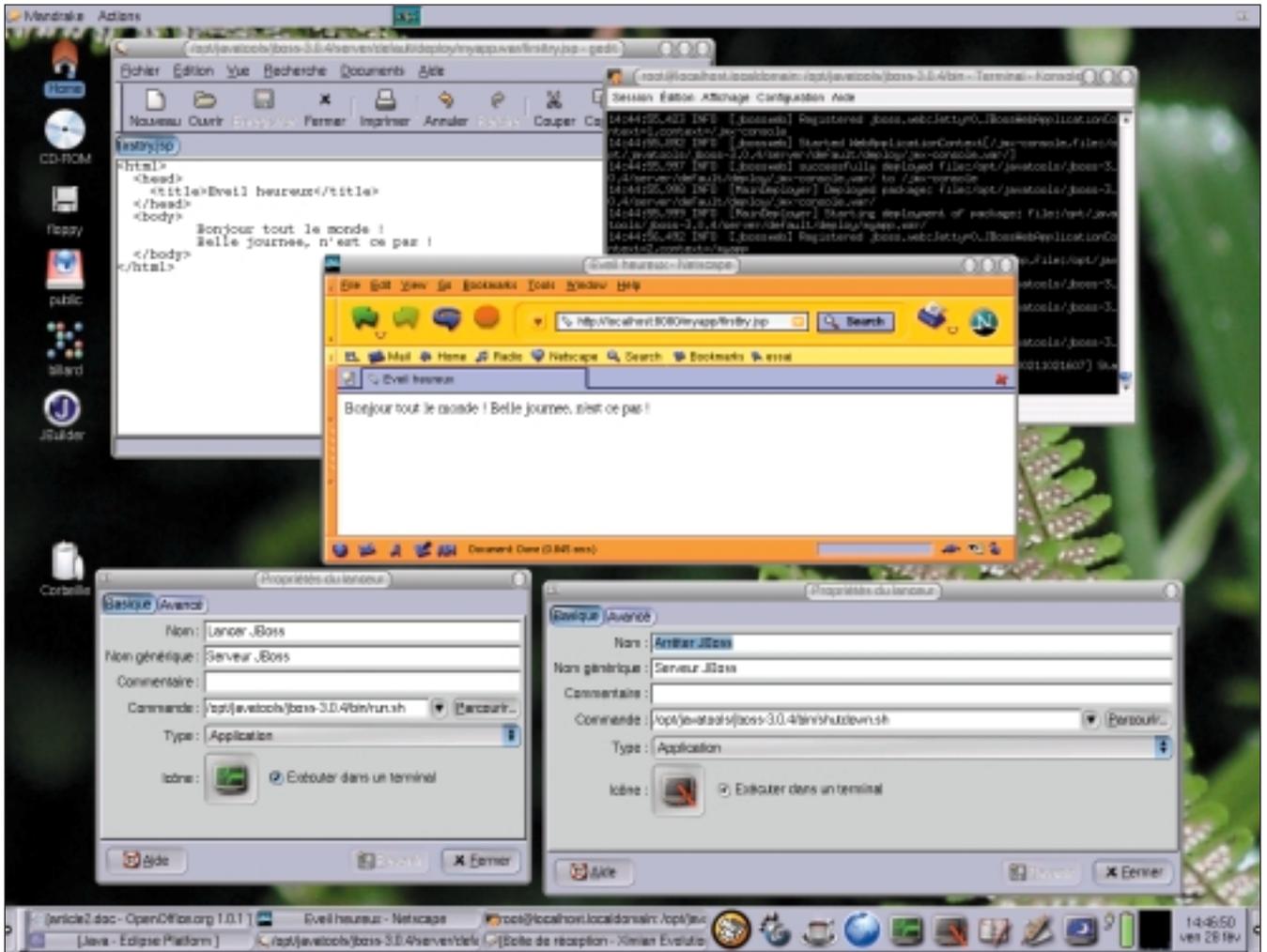
dans .cshrc :      setenv JAVA_HOME=/opt/IBMJava2-14
dans .bashrc     JAVA_HOME=/opt/IBMJava2-14
                  export JAVA_HOME
  
```

On peut aussi déclarer cette variable dans le fichier de lancement de Jboss comme nous allons le voir bientôt.

Inspectons l'arborescence

Pour nous familiariser avec Jboss, nous allons inspecter la partie de l'arborescence qui lui est propre. Sous le répertoire d'installation (`c:\javatools` ou `/op/javatools`) nous trouvons un répertoire appelé `jboss-3.0.4`. Tout jboss se trouve à l'intérieur : nul fichier n'a été semé ailleurs, nul registre du système n'a été altéré. Jboss est un produit qui ne bave pas. La désinstallation de Jboss est donc une tâche aussi sûre qu'enfantine : on prend le répertoire `jboss-3.0.4` et on le met dans la corbeille.

Mais nous n'en sommes pas là. Ouvrons le répertoire `jboss-3.0.4`. Nous découvrons cinq autres répertoires appelés "bin", "client", "docs", "lib" et "server". Le répertoire "bin" contient les fichiers et les ressources permettant de lancer et d'arrêter Jboss. Nous allons y revenir très rapidement. Le répertoire "client" contient des archives Java qu'un programme Java externe à Jboss (client swing par exemple) doit référencer dans son CLASSPATH pour



Le résultat de l'exécution de notre JSP s'affiche sur le navigateur. Le code de cette JSP est en édition sur KEdit. Au bas de la capture, on trouve les deux lanceurs, permettant de démarrer et d'arrêter JBoss

pouvoir dialoguer avec Jboss. Le répertoire "docs" contient diverses ressources documentaires : des DTDs, des exemples de fichiers de configuration, etc. Bref, beaucoup de choses, mais pas de documentation au sens où on l'entend généralement. Le répertoire "lib" contient des archives Java, à usage général, utilisé par Jboss. Le répertoire le plus important est le dernier : "server" qui contient à la fois le coeur de Jboss et nos applications. Ouvrons le répertoire "server". Nous trouvons là trois nouveaux répertoires nommés "all", "default" et "minimal". Le répertoire "all" contient une installation exhaustive de Jboss : tous les services sont présents et prêts à l'emploi. Le répertoire "minimal" contient une installation fonctionnelle mais réduite de Jboss : seuls les services indispensables ont été retenus. L'équipe qui a réalisé Jboss a visiblement compris la différence sémantique entre "exhaustif" et "indispensable": le répertoire "all" occupe 14 Mo. Le répertoire "minimal" occupe 500 Ko. L'installation qui nous intéresse est la médiane: c'est celle qui est hébergée par le répertoire " default ". Comme son nom l'indique, c'est elle qui est utilisée par défaut par Jboss. " all ", " default "

et " minimal " ont une structure similaire. Elles sont autonomes l'une par rapport à l'autre. Mais alors, peut on ne retenir que l'installation dont on a besoin et jeter les deux autres à la corbeille ? La réponse est oui. Le gain (15 Mo), au vu des caractéristiques des machines d'aujourd'hui paraît bien faible. Cela représente néanmoins près de 50% de la place occupée par Jboss sur le disque ! En taille, Jboss ne pèse pas lourd ! En valeur, c'est autre chose ... Entrons maintenant dans le répertoire " default ". Nous trouvons trois répertoires : " conf ", " lib " et " deploy ". Le premier de ces répertoires contient des fichiers de configuration finement élaborés par l'équipe Jboss et que donc nous allons laisser tranquille. Le répertoire " lib " contient les archives Java qui constituent le coeur de Jboss. Nos applications seront, elles, hébergées par le répertoire " deploy ". Ouvrons ce répertoire. Il semble régner ici une pagaille qui contraste avec l'organisation très rationnelle de toute l'arborescence que nous avons vue jusqu'ici. La plupart des fichiers ou des répertoires contenus dans " deploy " sont des applications de natures diverses. Même pour l'auteur de ces lignes, certains de ces éléments gardent encore leur mystère. Ce qui est sûr, c'est que cela n'a que peu d'importance, pour nous, développeurs ou exploitants d'applications Java/J2EE. Il nous suffira de faire une petite place à l'application que nous voulons mettre en oeuvre. Cette petite place va prendre la forme d'un nouveau répertoire que nous allons placer dans le répertoire "deploy". Dans un premier temps, cette application contiendra uniquement des composants Web. C'est pourquoi le répertoire qui

J2EE

NIVEAU : DÉBUTANT

la représente doit posséder " war " comme extension. Ce suffixe (war = Web ARchive) indique à Jboss que les composants contenus dans l'application sont des servlets et des JSP.

Créons notre premier composant

Il est temps de passer aux choses concrètes. Créons un répertoire que nous allons appeler "myapp.war". Dans ce répertoire, avec notre éditeur de texte préféré (wordpad sur Windows, Kedit ou Gedit sous Linux) nous allons créer notre premier composant.J2EE: ce sera une page JSP nommée firsttry.jsp. Je vous propose le contenu suivant :

```
<html>
<head>
  <title>Eveil</title>
</head>
<body>
  Bonjour tout le monde !
</body>
</html>
```

Sauvegardons sur disque ce contenu, sans pour autant refermer notre éditeur de texte. Nous allons maintenant nous placer dans le répertoire "bin" (situé, rappelons le, directement sous le répertoire racine de votre installation Jboss). Ce répertoire contient des fichiers appelés run.* et shutdown.*.

Le fichier de commande "run.bat" permet de lancer Jboss sur Windows. "shutdown.bat" l'arrête. Sur Linux, les fichiers de commandes correspondants sont "run.sh" et "shutdown.sh".

Lançons la commande "run" appropriée à la plateforme utilisée. Aucun paramètre n'est requis. Le plus simple est de se faire un raccourci (ou un lanceur sur Linux) qui active "run.bat" ou "run.sh". Une fenêtre terminal s'ouvre et se couvre rapidement de messages. Patientons quelques secondes. Le défilement s'arrête sur une ligne qui ressemble à celle ci :

```
14:13:16,101 INFO [Server] JBoss (MX MicroKernel) [3.0.4Date:2002
11021607]Started in 0m:21s:837ms
```

Notre serveur Jboss est prêt.

Vérifions que notre composant JSP est disponible. Pour cela ouvrons un navigateur Web. Dans le champ d'adresses, saisissons l'URL suivante :

<http://localhost:8080/myapp/firsttry.jsp>

Le navigateur semble hésiter pendant quelques secondes, puis affiche une page contenant "Eveil" en titre et "Bonjour tout le monde" en contenu. Succès !

Revenons quelques instants sur ce premier essai. Commençons par inspecter l'URL que nous venons de saisir. Nous y avons indiqué successivement quel protocole est utilisé (http), quelle machine nous désirions atteindre ("localhost", nom conventionnel pour désigner notre propre machine), le port IP utilisé par défaut par Jboss (8080) pour recevoir des requêtes, le nom de l'application (myapp) contenant la ressource recherchée (firsttry.jsp). Notons que le nom de l'application est celui du répertoire de notre application, sans son extension.

Appuyez sur le bouton "rafraîchir" de votre navigateur. La page se ré affiche instantanément. Il lui avait fallu quelques secondes au premier essai. Que s'est il passé ?

Contrairement à une croyance très répandue, les pages JSP ne

sont pas interprétées par les serveurs Web dynamiques. Elles sont transformées, à la volée, en composants Java (des servlets pour être précis) qui sont ensuite compilés et chargés en mémoire. Ce procédé est exécuté au premier appel pour chaque JSP. Ensuite, l'objet Java compilé et chargé en mémoire est utilisé directement. C'est la raison pour laquelle Jboss répond plus rapidement à partir du second affichage. Néanmoins Jboss surveille le fichier d'origine. Si celui ci est modifié, le processus de génération et de compilation est renouvelé. Retournons à notre éditeur de texte et modifions notre JSP :

```
<html>
<head>
  <title>Eveil heureux</title>
</head>
<body>
  Bonjour tout le monde !
  Belle journée, n'est ce pas !
</body>
</html>
```

Sauvegarde. Ré-affichage dans le navigateur. Celui ci hésite à nouveau avant de nous fournir un titre et un contenu nouveau. Une modification a été détectée. Si vous utilisez le bouton "rafraîchir", vous constaterez que de nouveau, Jboss devient beaucoup plus réactif.

Tout ceci explique pourquoi Jboss a besoin d'un JDK (installation de développement de Java) et non pas d'un JRE (installation d'exécution). Il utilise non seulement sa machine virtuelle, mais aussi son compilateur ...

Avant de nous plonger en profondeur dans les développements J2EE, je voudrais terminer cet exposé sur l'installation et découverte de Jboss en indiquant aux lecteurs une autre manière de signaler à Jboss quel est le JDK qu'il doit utiliser. Elle consiste à préciser la valeur d'environnement JAVA_HOME dans les fichiers "run.xxx" ou "shutdown.xxx.". En première ligne de "run.bat" ou de "shutdown.bat" il faut écrire:

```
set JAVA_HOME=c:\java\jdk-1.4.1
```

Les fichiers "run.sh" et "shutdown.sh", eux, débiteront par :

```
JAVA_HOME=/opt/IBMJava2-14
export JAVA_HOME
```

Pour les lecteurs qui comme moi manquent de patience, il existe une façon plus rapide et radicale d'arrêter un serveur Jboss, que celle d'invoquer "shutdown.xxx". Il suffit de fermer la fenêtre terminal. En production ce n'est bien sûr pas recommandé (les sessions en cours sont brutalement interrompues). Mais pendant la phase de développement, je ne vois aucune contre-indication : vous ne risquez pas d'endommager votre installation Jboss. ■

Henri Darmet

Consultant/Formateur, Objet Direct.

Homsys Group

Créé en 1991, Homsys Group est spécialisé autour de trois grands pôles : la Business Intelligence, les Architectures Systèmes et Réseaux et les technologies Objet et Internet. Le groupe, qui fédère aujourd'hui 250 consultants, est implanté à Bordeaux, Grenoble, Lyon, Marseille, Paris, Rennes et Toulouse. <http://www.homsysgroup.com>



Programmons un consommateur du service Web de traduction Ba belFish (en Visual FoxPro, C# et RUBY)

PRATIQUE

- Comment traduire une interface en anglais ?
- Comment imprimer un message d'avertissement en plusieurs langues ?
- Le Service Web d'Altavista (<http://world.altavista.com/>) va s'en charger à notre place.

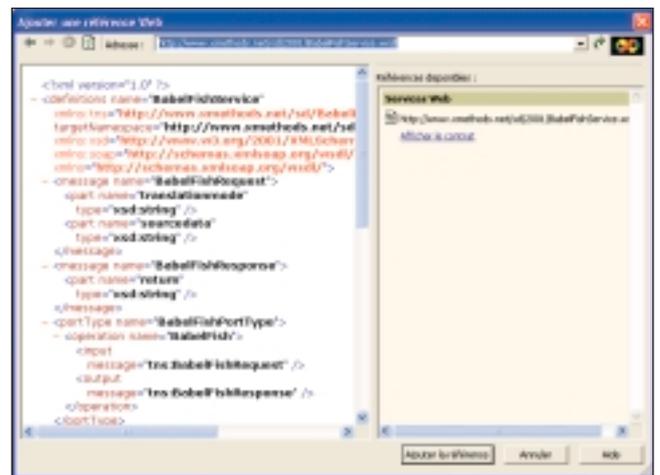


L'interface Web on line d'Altavista (<http://world.altavista.com>)

Le service Web de traduction BABELFISH est répertorié sur le site portail de XMethods (<http://www.xmethods.com>). Xmethods offre gratuitement la possibilité d'utiliser ou de proposer des services Web "d'intérêt public". Différentes implémentations de SOAP sont disponibles. Au début il n'y avait que celle en Java (ApacheSOAP), et celle en Perl (SOAP::Lite) mais aujourd'hui les implémentations sont nombreuses (<http://www.xmethods.com/ve2/ViewImplementations.po>).

Rappelons que pour être utilisable un Service Web doit fournir la description complète des éléments suivants :

- SOAP End point URL:** l'URL à partir duquel le serveur SOAP reçoit la requête ;
- SOAPAction:** l'entête SOAPAction ;
- Method Namespace URI:** l'URI (Univeral Resource Identifier) de l'objet dont la méthode est appelée ;
- Method Name:** Le nom de la (ou des) méthode(s) disponible(s) ;



Acceptez la référence au service Babelfish, pour l'incorporer à votre projet.

NIVEAU : INTERMEDIAIRE

WSDL URL: L'URL d'un fichier qui décrit complètement le service au format WSDL ;

Server Implementation: L'implémentation SOAP utilisé (comme ApacheSOAP).

Description WSDL du service Web BabelFish

WSDL (*Web Service Description Language*) est un standard qui décrit sous la forme d'une syntaxe XML un service Web. Ce document WSDL possède des sous-éléments très bien définis. Il commence par une balise décrivant le service appelé.

```
<?xml version="1.0" ?>
<definitions
  name="BabelFishService"
  targetNamespace=
    "http://www.xmethods.net/BabelFishService"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="BabelFishRequest">
  <message name="BabelFishResponse">
  <portType name="BabelFishPortType">
  <binding name="BabelFishBinding"
    type="BabelFishPortType">
  <service name="BabelFish">
</definitions
```

La balise <message> reprend la définition des données d'entrées/sorties (Payload XML).

```
<message name="BabelFishRequest">
  <part name="translationmode" type="xsd:string" />
  <part name="sourcedata" type="xsd:string" />
</message
```

La balise <PortType> liste l'ensemble des opérations supportées par une série de points d'arrêts :

```
<binding name="BabelFishBinding"
  type="BabelFishPortType">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="BabelFish">
  <soap:operation
    soapAction="urn:xmethodsBabelFish#BabelFish" />
  <input>
  <soap:body
    use="encoded"
    namespace="urn:xmethodsBabelFish"
    encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/" />
  </input>
  <output>
  <soap:body
```

```
    use="encoded" namespace="urn:xmethodsBabelFish"
    encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/" />
  </output>
  </operation>
</binding>
```

La balise <binding> définit le format des paramètres d'entrées/sorties et du protocole utilisé (connexion entre le protocole HTTP et le "Payload XML").

```
<binding name="BabelFishBinding"
  type="BabelFishPortType">
  <soap:binding
    style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
  <operation name="BabelFish">
  <soap:operation soapAction="urn:xmethodsBabelFish#BabelFish" />
  <input>
  <soap:body
    use="encoded"
    namespace="urn:xmethodsBabelFish"
    encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/" />
  </input>
  <output>
  <soap:body
    use="encoded" namespace="urn:xmethodsBabelFish" encodingStyle=
      "http://schemas.xmlsoap.org/soap/encoding/" />
  </output>
  </operation>
</binding>
```

Finalement la balise <service> reprend l'adresse du point d'arrêt, son nom et d'éventuels commentaires.

```
<service name="BabelFish">
  <documentation>Translates text.</documentation>
  <port name="BabelFishPort" binding="BabelFishBinding">
  <soap:address
    location=
      "http://services.xmethods.net:80/perl/soaplite.cgi" />
  </port>
</service>
```

Retenons qu'un service Web est décrit en WSDL, puis classé avec l'annuaire UDDI, et enfin est invoqué avec le protocole SOAP.

Sous Windows en Visual FoxPro (outil de développement Microsoft)

En théorie, il suffit de configurer votre poste en utilisant une implémentation appropriée, puis d'invoquer l'objet SOAP (en définissant son point d'arrêt, son URI et la méthode à appeler qui correspond au service demandé).

Par exemple en Visual Foxpro vous écrirez :

```
LOCAL x
x = CREATEOBJECT("MSSOAP.SoapClient")
x.MSSoapInit("http://www.xmethods.net/sd/2001/BabelFishService.wsdl";,
"BabelFishService", "", "")
? x.babelfish('fr_en', 'renard')
```

Ou encore :

```
LOCAL loBabel as Traductor
LOCAL loWS
loWS = NEWOBJECT("WscClient", HOME() + "\ffc\_webservices.vcx")
loWS.cWSName = "Traductor"
loBabel = loWS.SetupClient("http://www.xmethods.net/sd/BabelFishService.wsdl", "BabelFish", "BabelFishPort")
? loBabel.BabelFish("fr_en", "bonjour")
```

Seulement, il peut arriver que votre client ne puisse accéder au service, en raison des restrictions de sécurité imposées par votre pare-feu (c'est mon cas).

XMethods a prévu ce cas de figure. En effet l'ensemble des services, y compris BabelFish, sont accessibles via un mandataire (proxy) sur le port 80 (<http://services.xmethods.net:80/perl/soaplite.cgi>).

Ce qui donne :

```
CLEAR
oHTTP = create('MSXML2.XMLHTTP')

PaquetHTTP = '<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">'
PaquetHTTP = PaquetHTTP + '<SOAP:Header></SOAP:Header>'
PaquetHTTP = PaquetHTTP + '<SOAP:Body>'
PaquetHTTP = PaquetHTTP + '<m:BabelFish xmlns:m="urn:xmethodsBabelFish">'
PaquetHTTP = PaquetHTTP + '<translationmode>en_fr</translationmode>'
PaquetHTTP = PaquetHTTP + '<sourcedata>Fox</sourcedata>'
PaquetHTTP = PaquetHTTP + '</m:BabelFish>'
PaquetHTTP = PaquetHTTP + '</SOAP:Body></SOAP:Envelope>'
```

```
oHTTP.open("post", "http://services.xmethods.net:80/perl/soaplite.cgi", .F.)
oHTTP.setRequestHeader("Content-Type", "text/xml")
oHTTP.setRequestHeader("SOAPAction", "urn:xmethodsBabelFish#BabelFish")

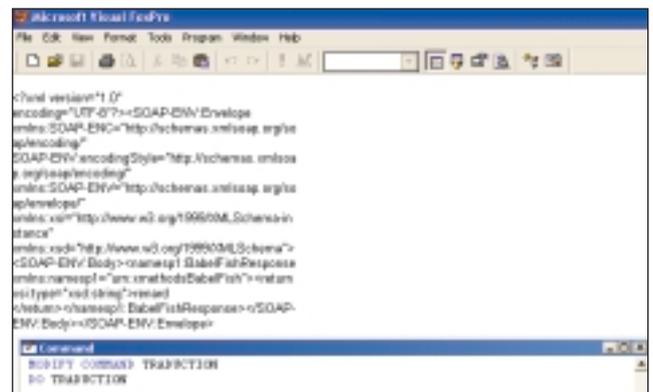
oHTTP.send(PaquetHTTP)
strReturn = oHTTP.responseText

? strReturn
```

Et cela fonctionne parfaitement. L'enveloppe renvoyée...

```
<?xml version="1.0"
encoding="UTF-8"?><SOAP-ENV:Envelope
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema">
<SOAP-ENV:Body><namesp1:BabelFishResponse
xmlns:namesp1="urn:xmethodsBabelFish"><return
xsi:type="xsd:string">renard
</return></namesp1:BabelFishResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

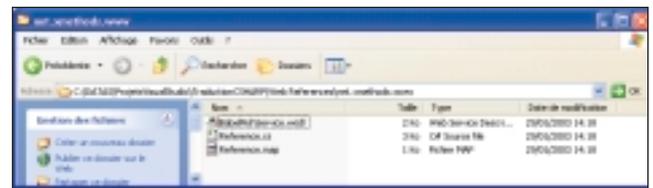
... contient bien la traduction française "renard" du mot en anglais "fox". Génial !



exécution sous Visual Fox Pro 7.0

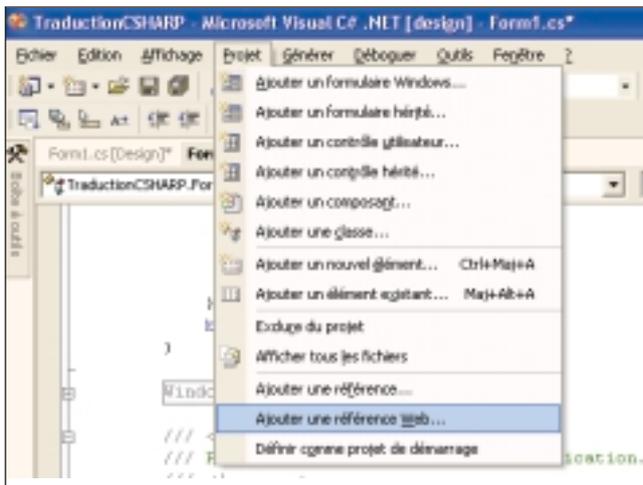
Complément d'informations pour les langages sous Windows (cas du C#)

Attention. Si vous ne programmez pas vous-même l'enveloppe SOAP comme le pratique le dernier exemple VFP, il est d'abord nécessaire de construire en local un fichier BabelFishService.wsdl (en raison du fait que l'implémentation SOAP de BabelFish est de type Apache).



Les fichiers nécessaires sont créés par VS.NET en local

Sous Visual STUDIO.NET il vous suffira de créer un nouveau projet C# (ici du nom de "TraductionCSHARP"), puis d'y ajouter la référence Web vers le service BabelFish



Ajout d'une référence web dans Visual Studio.NET

(<http://www.xmethods.net/sd/2001/BabelFishService.wsdl>). Après compilation à votre droite sous "Web references" figure bien maintenant les références au service BabelFish.



Vous voyez apparaître la référence WebBabelFish dans votre environnement VS.NET

Sur le formulaire form1, ajoutez quatre objets textbox. Le premier sera initialisé par défaut à "fr_en", le deuxième à "renard", le troisième et le quatrième seront vides.

Créez ensuite un bouton de commande, puis double-cliquez dessus.

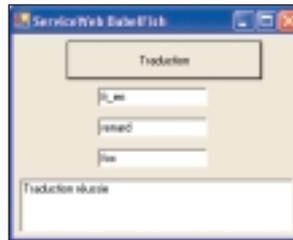
Le bout de code ajouté est celui-ci :

```
private void button1_Click(object sender, System.EventArgs e)
{
    BabelFishService fish=new BabelFishService();
    try
    {
        textBox3.Text=fish.BabelFish(textBox1.Text,this.textBox2.Text);
        textBox4.Text="Traduction réussie";
    }
    catch(System.Net.WebException )
    {
        textBox4.Text="Echec de la traduction (service Web indisponible ?)";
    }
}
```

N'oubliez pas non plus d'ajouter l'espace de noms du service de traduction :

```
...
using System.Data;
using TraductionCSHARP.net.xmethods.www;
...
```

Et c'est tout...



l'exemple final en C#

Sous Unix en RUBY

Ruby est un langage de script comme TCL ou Python (il existe une implémentation sous Windows également). Récupérez Ruby à cette adresse <http://www.ruby-lang.org/en/download.html>, décompressez et compilez l'archive. Nous allons utiliser l'implémentation soap4r

(<http://www.xmethods.com/ve2/ViewImplementation.po?xid=soap4r>).

Vous devez non seulement installer SOAP4R sous Linux, mais aussi les modules qui en dépendent (l'adresse <http://www.jin.gr.jp/~nahi/Ruby/SOAP4R>). Pour le processeur XML j'ai choisi "rexml" car il sera implémenté d'office avec la nouvelle version de RUBY (la version 1.8). Il faut aussi installer les modules date2, devel-logger, http-access2 et Uconv. Il suffit généralement de décompresser l'archive, puis d'exécuter ruby install.rb dans le sous-répertoire du module à installer.



Exécution de l'exemple RUBY

Editez le fichier traduction.rb :

```
#!/usr/bin/env ruby

text = ARGV.shift || 'bonjour'
lang = ARGV.shift || 'fr_en'

require 'soap/driver'

server = 'http://services.xmethods.net/perl/soaplite.cgi'
proxy = ENV[ 'HTTP_PROXY' ] || ENV[ 'http_proxy' ]
InterfaceNS = 'urn:xmethodsBabelFish'

drv = SOAP::Driver.new( nil, $0, InterfaceNS, server, proxy )
drv.setWireDumpDev( nil )
drv.addMethodWithSOAPAction( 'BabelFish', InterfaceNS + "#BabelFish",
'translationmode', 'sourcedata' )

print drv.BabelFish( lang, text ) + "\n"
```

Par défaut, si vous exécutez ruby traduction.rb, le service web BabelFish traduira "bonjour", sinon il prendra en compte les arguments en ligne de commande.

Pour traduire "hello" en italien :

ruby traduction.rb "hello" "en_it" ce qui donne "ciao"

Pour rappel les possibilités sont "en_fr", "en_it" (Italien), "en_de" (Allemand), "en_pt" (Portugais), "es_en" (Espagnol vers Anglais), "fr_en", "it_en", "de_en", "pt_en" et "ru_en" (Russe vers Anglais).

Bonne traduction !

Xavier Leclercq

Programmation Windows avec C#

Nous apprenons
aujourd'hui à
construire des boîtes
de dialogues avec C#

Résumons nous. Nous savons écrire une application Windows rudimentaire avec C#. Nous savons intégrer un menu à notre application. Il est clair que le menu, tout pratique qu'il soit, permet seulement à l'utilisateur d'orienter le cours du déroulement d'un programme. Il n'est pas question d'utiliser un menu pour faire des saisies. Pour ce faire, l'interface adaptée s'appelle une boîte de dialogue. Une boîte de dialogue est une fenêtre équipée de contrôles tels que boutons, boîte liste défilante, case à cocher, etc.

Une boîte de dialogue de base

Nous allons constater avec satisfaction qu'il est remarquablement simple de construire une boîte de dialogue avec C#. C'est même tellement simple qu'il n'y a pas de classe dédiée à cet usage, contrairement à Java qui offre une classe `JDialog` par exemple.

Une boîte de dialogue peut être modale, ou non modale. Lorsqu'une boîte de dialogue est modale, l'utilisateur ne peut pas agir sur le reste de l'application tant que la boîte n'est pas fermée. Lorsqu'une boîte de dialogue est non modale, l'utilisateur peut continuer à travailler tandis que la boîte est ouverte. Bien sûr ce dernier cas oblige le programmeur à ajouter un peu de code pour gérer la communication entre la boîte de dialogue et l'application, ce que nous n'étudierons pas aujourd'hui. Dans les deux cas, une boîte de dialogue est avant tout une instance de la classe `Form`. La différence tenant seulement à la façon de faire surgir la fenêtre à l'écran. On invoquera la méthode `Show` de la classe, pour une boîte de dialogue non modale et la méthode `ShowDialog`, pour une boîte de dialogue modale.

Nous allons commencer par écrire le code pour construire une boîte de dialogue avec les deux boutons poussoirs d'usage, 'Ok' et 'Annuler'. (figure 1). Comme vous pouvez le constater en consultant le code ci-contre (encadré 1), nous dérivons une classe de la classe `Form`. Ensuite les choses se déroulent en trois étapes :

- Dans le constructeur on procède aux initialisations d'usage pour contrôler l'aspect et la position de la boîte de dialogue.
- Dans le constructeur toujours, on instancie les contrôles, ici



> Figure 1: Une première boîte de dialogue des plus rudimentaire.

encadré 1

```
// Une boîte de dialogue rudimentaire

using System;
using System.Drawing;
using System.Windows.Forms;

class Demo : Form {

    public static void Main() {
        Application.Run(new Demo());
    }

    public Demo () {
        Text = "Démon de boîte de dialogue";
        MenuItem DialogMenu = new MenuItem("Ouvrir le dialogue",
                                           new EventHandler(
OnOpenDialogClick));
        Menu = new MainMenu(new MenuItem[] {DialogMenu});
    }

    void OnOpenDialogClick(object obj, EventArgs ea) {
        string result = "Résultat de la boîte de dialogue: ";
        DemoDialog dlg = new DemoDialog();
        dlg.ShowDialog();
        result += dlg.DialogResult;
        Console.WriteLine(result);
    }
}

class DemoDialog : Form {
    public DemoDialog() {
        Text = "Ceci est une boîte de dialogue";

        // Présentation classique d'une boîte de dialogue
        StartPosition = FormStartPosition.CenterParent;
        FormBorderStyle = FormBorderStyle.FixedDialog;
        ControlBox = true; //false;
        MinimizeBox = false;
        MaximizeBox = false;
        ShowInTaskbar = false;

        // Mettre en place un bouton Ok
        Button okb = new Button();
        okb.Parent = this;
        okb.Text = "Ok";
        okb.Location = new Point(50, 220);
        okb.Click += new EventHandler(OnOkbClick);

        // Mettre en place un bouton Cancel
        Button cancelb = new Button();
        cancelb.Parent = this;
        cancelb.Text = "Annuler";
        cancelb.Location = new Point(170, 220);
        cancelb.Click += new EventHandler(OnCancelbClick);
    }

    void OnOkbClick(object obj, EventArgs ea) {
        // Ici éventuellement, on vérifie la validité des données
        // saisies par l'utilisateur
        DialogResult = DialogResult.OK;
    }

    void OnCancelbClick(object obj, EventArgs ea) {
        DialogResult = DialogResult.Cancel;
    }
}
```

deux boutons, que l'on positionne dans la fenêtre. Surtout on n'oublie pas d'attacher les boutons à la fenêtre, en affectant leur propriété 'Parent'. Sans cela les boutons resteraient invisibles. Puis on affecte à chacun des boutons le gestionnaire d'événement devant être appelé, lorsque l'utilisateur cliquera le bouton.

- On écrit les gestionnaires d'événements mentionnés plus haut. Le code en est on ne peut plus simple. On se contente d'affecter la propriété DialogResult de la Form. Le simple fait d'affecter cette propriété provoque la disparition de la fenêtre à l'écran. La fenêtre est fermée, mais pas détruite. L'instance de l'objet reste parfaitement opérationnelle et la fenêtre réapparaîtra à l'écran si on invoque à nouveau sa méthode ShowDialog. La valeur de DialogResult, prise dans l'énumération DialogResult peut être testée par l'application à la suite de l'appel à ShowDialog, ce que fait le programme d'exemple. Une autre possibilité est de tester cette valeur comme valeur de retour de ShowDialog. Le code de ce type de gestionnaire peut être enrichi. Le gestionnaire d'événement du bouton 'Ok' est un bon endroit pour tester la validité des saisies effectuées par l'utilisateur. S'il apparaît que les saisies sont incorrectes, on affichera un message, puis on rendra la main sans affecter DialogResult. Dans ce cas la fenêtre reste ouverte et l'utilisateur doit corriger sa saisie.

Soigner les apparences

Examinons de plus près les paramètres qui permettent de contrôler l'apparence à l'écran de la fenêtre et son comportement. Commençons par StartPosition qui est une valeur de l'énumération FormStartPosition.

Valeur	Description
CenterParent	La boîte de dialogue se centre au dessus de sa fenêtre parent. Cette valeur est la plus couramment employée.
CenterScreen	La boîte de dialogue se centre dans l'écran.
Manual	La position de la boîte de dialogue se centre dans l'écran par le système en fonction de sa taille.
WindowsDefaultBounds	La fenêtre est positionnée et dimensionnée à partir des critères par défaut de Windows. Cette valeur n'est pas adaptée aux boîtes de dialogue.
WindowsDefaultLocation	La fenêtre est positionnée d'après les paramètres par défaut de windows. Cette valeur présente peu d'intérêt.

Examinons ensuite l'initialisation de 4 autres propriétés de l'objet Form. Ces propriétés sont toutes à true par défaut :

Propriété	Description
ControlBox	Mettre cette propriété à false fait disparaître le bouton de fermeture en haut à droite de la fenêtre.
MinimizeBox	Mettre cette propriété à false rend inactif le bouton de réduction de la fenêtre.
MaximizeBox	Mettre cette propriété à false rend inactif le bouton d'agrandissement de la fenêtre. Si cette propriété et la précédente sont à false toutes les deux, les boutons disparaissent.
ShowInTaskBar	Mettre cette propriété à false empêche la fenêtre d'apparaître sur la barre de tâche, ce qui est en général le comportement souhaitable.

Enfin pour en terminer avec les apparences, disons que l'ordre dans lequel on crée les boutons n'est pas indifférent. Le premier bouton créé est le bouton par défaut, et comme tel entouré d'un épais rectangle noir.

Simplifier le code

Si l'on ne souhaite pas valider les saisies de l'utilisateur au sein du gestionnaire d'événement des boutons, il est possible de simplifier le code. Dans ce cas, on n'écrit pas de gestionnaire d'événement et on affecte les propriétés DialogResult des boutons eux-mêmes. Lorsque l'utilisateur clique sur un bouton, le système se charge alors de fermer la fenêtre et de transmettre la valeur DialogResult à la boîte de dialogue puis à la fonction appelante ShowDialog. Voir ci-dessous.

encadré 2

```
// Listing partiel
// Le code de notre boîte de dialogue après simplification

class DemoDialog : Form {
    public DemoDialog() {
        Text = "Ceci est une boîte de dialogue";

        // Présentation classique d'un boîte de dialogue
        StartPosition = FormStartPosition.CenterParent;
        FormBorderStyle = FormBorderStyle.FixedDialog;
        ControlBox = true; //false;
        MinimizeBox = false;
        MaximizeBox = false;
        ShowInTaskbar = false;

        // Mettre en place un bouton Ok
        Button okb = new Button();
        okb.Parent = this;
        okb.Text = "Ok";
        okb.Location = new Point(50, 220);
        okb.DialogResult = DialogResult.OK;

        // Mettre en place un bouton Cancel
        Button cancelb = new Button();
        cancelb.Parent = this;
        cancelb.Text = "Annuler";
        cancelb.Location = new Point(170, 220);
        cancelb.DialogResult = DialogResult.Cancel;
    }
}
```

Soigner le comportement

Notre boîte de dialogue se ferme bien, lorsqu'on clique sur les boutons. Mais si vous êtes un utilisateur rompu à Windows, vous savez que les boîtes de dialogue ont toutes comme comportement commun de se fermer lorsque l'utilisateur presse la touche d'échappement, ou la touche return. Or pour l'instant, notre boîte de dialogue n'obéit pas à cette règle, en ce qui concerne la touche d'échappement. Il y a deux façons de régler le problème. On peut tester directement l'appui sur la touche. Cela présente l'inconvénient d'écrire du code, mais cela peut être valable dans le cadre de gestion de dialogues très complexes. Ou alors, on utilise la méthode du paresseux. Cela consiste en l'affectation des propriétés AcceptButton et CancelButton de la fiche (l'objet Form donc) avec les boutons 'Ok' et 'Annuler' respectivement. Concrètement, cela revient à ajouter deux lignes de code, en tout et pour tout, au constructeur de l'exemple précédent:

```
AcceptButton = okb;
CancelButton = cancelb;
```

La méthode fonctionne aussi si des gestionnaires d'événements sont installés, comme nous l'avons fait dans le tout premier exemple. Avantage supplémentaire: l'ordre de création des boutons devient indifférent. Le bouton par défaut étant très logiquement celui affecté à la propriété AcceptButton.

Une boîte de dialogue complète

Nous pouvons maintenant nous lancer dans la construction d'une 'vraie' boîte de dialogue, c'est à dire une boîte de dialogue capable de récupérer des données saisies par l'utilisateur de l'application (figure 2). En outre nous devons gérer l'ordre de tabulation des contrôles et la focalisation cette fois. Rien de bien compliqué. Un contrôle fait partie de l'ordre de tabulation si et seulement si sa propriété TabStop est à true, ce qui est la valeur par défaut. L'ordre de tabulation est défini au moyen de la propriété TabIndex de chaque contrôle. Cette valeur doit avoir une valeur supérieure ou égale à zéro. L'ordre de tabulation suit les valeurs de ces propriétés en ordre croissant. Enfin pour qu'un contrôle ait la focalisation à l'ouverture de la fenêtre, il suffit d'invoquer la méthode Focus de ce contrôle.

Pour initialiser et récupérer les valeurs maintenues dans les contrôles, le plus simple et le plus élégant est de définir nous-mêmes des propriétés additionnelles à l'objet de type Form. Une



> Figure 2: Une boîte de dialogue complète et de bon goût ;-)

C#

NIVEAU : DÉBUTANT

telle chose se fait en déclarant une méthode publique, contenant elle-même deux méthodes invariablement baptisées 'get' et 'set'. Ces deux méthodes incluses sont des accesseurs, invoqués automatiquement par le compilateur. La méthode get est invoquée lorsqu'on lit la propriété et la méthode set est invoquée lorsqu'on l'affecte. Comme vous pouvez le constater dans l'exemple ci-contre (encadré 3) la méthode set reçoit automatiquement, bien que cela ne se "voit pas", un paramètre de même type que la valeur de retour de la méthode de propriété. Ce paramètre est systématiquement rangé dans une variable du nom de 'value'. Ceci fait, l'utilisation est toute simple. Via les propriétés ainsi définies, on initialise les contrôles de la boîte de dialogue depuis la partie principale de l'application, juste avant d'invoquer la méthode ShowDialog, puis on récupère le résultat des courses juste après. Nous avons fait le tour de la question en ce qui concerne les boîtes de dialogues modales. C# est un langage doté de qualités incontestables. Son maniement facile permet d'envisager des tâches habituellement fastidieuses comme la création manuelle d'interfaces utilisateur. Pour un débutant qui n'a pas les moyens de s'offrir un Visual Studio .Net, c'est une perspective plutôt sympathique.

encadré 3

```
// Une boîte de dialogue complètement opérationnelle.

using System;
using System.Drawing;
using System.Windows.Forms;

class Demo : Form {

    public static void Main() {
        Application.Run(new Demo());
    }

    public Demo () {
        Text = "Démo de boîte de dialogue";
        MenuItem DialogMenu = new MenuItem("Ouvrir le dialogue",
        new EventHandler(OnOpenDialogClick));
        Menu = new MainMenu(new MenuItem[] {DialogMenu});
    }

    void OnOpenDialogClick(object obj, EventArgs ea) {
        DemoDialog ddlg = new DemoDialog();
        ddlg.Saisie = "Python ;-);";
        if(ddlg.ShowDialog() == DialogResult.OK) {
            string result = "Votre langage préféré = ";
            result += ddlg.Saisie;
            Console.WriteLine(result);
        }
        else {
            Console.WriteLine("Python rules ! ;-);");
        }
    }
}

class DemoDialog : Form {

    private TextBox tb;

    public DemoDialog() {
        Text = "Ceci est une boîte de dialogue";

        // Présentation classique d'un boîte de dialogue
        StartPosition = FormStartPosition.CenterParent;
        FormBorderStyle = FormBorderStyle.FixedDialog;
        ControlBox = true; //false;
```

```
        MinimizeBox = false;
        MaximizeBox = false;
        ShowInTaskbar = false;

        // Mettre en place un bouton Ok
        Button okb = new Button();
        okb.Parent = this;
        okb.Text = "Ok";
        okb.Location = new Point(50, 220);
        okb.DialogResult = DialogResult.OK;
        okb.TabIndex = 1;

        // Mettre en place un bouton Cancel
        Button cancelb = new Button();
        cancelb.Parent = this;
        cancelb.Text = "Annuler";
        cancelb.Location = new Point(170, 220);
        cancelb.DialogResult = DialogResult.Cancel;
        cancelb.TabIndex = 2;

        AcceptButton = okb;
        CancelButton = cancelb;

        // Mettre en place le Label
        Label label = new Label();
        label.Parent = this;
        label.AutoSize = true;
        label.Text = "Quel est votre langage préféré ?";
        label.Location = new Point(50, 85);
        label.TabStop = false;

        // Mettre en place le TextBox
        Rectangle rect;
        tb = new TextBox();
        tb.Parent = this;
        tb.Location = new Point(50, 100);
        rect = tb.Bounds;
        rect.Width = 200;
        tb.Bounds = rect;
        tb.TabIndex = 0;

        tb.Focus();
    }

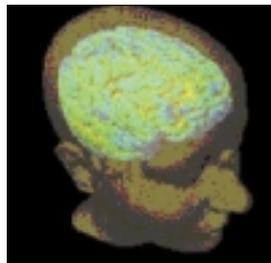
    public string Saisie {
        get {
            return tb.Text;
        }
        set {
            tb.Text = value;
        }
    }
}
```

Frédéric Mazué

frederic.mazue@programmez.com

Wiki : chaînon manquant entre "Knowledge Management" et Internet ?

Le Knowledge Management permet de gérer (dans le sens du partage et de la circulation) les connaissances au sein de votre entreprise. Si vous ne savez pas de quoi il est question, c'est le moment de se poser les bonnes questions.



PHP

NIVEAU : INTERMÉDIAIRE

Prenons le cas d'un chef de projet et de ses développeurs. Si l'un d'entre eux quittait l'équipe, est-ce que son départ mettrait en danger le projet ? Autrement dit, les connaissances qu'il détient sont-elles conservées quelque part ? L'information circule-t-elle bien entre le client et votre équipe de développement ? Les développeurs ont des idées innovantes : que deviennent-elles ?

Les développeurs sont-ils uniquement responsables du dévelop-



pement, ou sont-ils aussi à l'écoute des desiderata des clients (car sans leur aide, ces clients n'arriveraient même pas à formuler leurs plaintes).

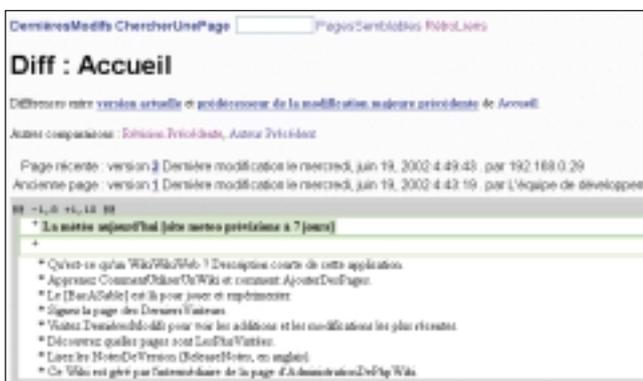
La gestion des connaissances est, comme on le voit, un sujet très large. Chaque éditeur propose ses solutions. Par exemple pour Microsoft, les outils pour structurer la connaissance dans l'entreprise sont : Microsoft Exchange Server, Office XP et SQL Server.

Nous allons vous présenter phpwiki, qui se situe en dehors des produits logiciels verticaux commerciaux spécialisés dans la gestion des connaissances. Il répond pourtant à bon nombre de principes et d'objectifs du Knowledge Management (lire encadré). Alors pourquoi s'en priver ? Surtout qu'il est gratuit...

Mais qu'est-ce au juste que le Wiki ?

Le WIKI peut améliorer l'accessibilité de ces connaissances. Il pourra aider l'entreprise à créer des "communautés de partage" des connaissances. Et si le WIKI est en libre accès depuis Internet, des personnes externes aideront gratuitement à entretenir les connaissances nécessaires au développement de l'entreprise. Le WIKI est à considérer comme une base de données "dynamique" de ces connaissances. Un outil informatique ultra simple d'emploi au service de tous.

En quelques mots, le Wiki autorise tout visiteur à modifier des pages en ligne !



Ne vous étonnez pas de tomber un jour sur une page d'accueil d'un site Wiki déclarant :

"Tout le monde peut y écrire n'importe quoi. Je n'aime pas la censure, mais je me réserve le droit de suspendre une page à contenu délictueux. Le bac à sable sert à faire des essais. Les critiques ouvertes à l'encontre d'autres personnes sont bien entendu autorisées, à la condition qu'elles soient signées : pas de diffamation anonyme (je veux bien mettre mon nom sur certaines critiques si je les trouve justifiées et si leur auteur initial ne souhaite pas les signer, mais ça n'est négociable qu'au cas par cas). Effectivement, n'importe qui peut "tout effacer".

Le Wiki est comme une grande ardoise (ou un grand tableau blanc pour faire plus moderne) pour laquelle chacun dispose d'une craie et peut effacer ce qu'il veut. Mais l'intérêt est de construire, pas de détruire. La meilleure protection d'un Wiki, c'est le bon sens et le respect mutuel. Cependant, si cela peut en rassurer certains, n'ayez pas peur, il y a des sauvegardes régulières utilisables en cas d'"accident".

Un wiki est un outil "d'interaction collective du web". Il est pos-



Knowledge Management : 7 principes et 7 objectifs

7 principes :

- Les connaissances de l'entreprise situées du côté des clients ;
- Les connaissances se trouvant dans les produits et les services ;
- Le capital humain ;
- Les connaissances contenues dans les processus ;
- La mémoire organisationnelle ;
- La mémoire transactionnelle ;
- Les connaissances en tant que biens immatériels.

7 objectifs :

- Reconnaître et protéger le savoir de l'entreprise ;
- Identifier les connaissances et les compétences en dehors de votre entreprise ;
- Apprendre à la mémoire transactionnelle ;
- Reconnaître les éléments responsables du succès de votre entreprise ;
- Créer les connaissances nécessaires au développement de l'entreprise ;
- Développer les connaissances et les compétences existantes ;
- Améliorer l'accessibilité de ces connaissances.

sible de modifier ("EditThisPage") chaque page du site.

"Sur ce site, toi, tu peux modifier les pages, en ajouter, (en supprimer), et réellement participer à sa construction. Ce site n'appartient à personne et n'est dirigé par personne. C'est toi qui le fait vivre par ta contribution. C'est toi l'acteur !"

Un excellent exemple de site Wiki est la première encyclopédie universelle basée sur ce principe de contribution collective : Wikipedia (<http://www.wikipedia.com/>). Son équivalent francophone est <http://fr.wikipedia.com/>. Pour la petite histoire, "Wiki wiki" signifie "vite" en hawaïen...

PhpWiki

Vous pouvez récupérer PHPWIKI sur le site <http://sourceforge.net/projects/phpwiki/>. Phpwiki une fois décompressé occupe près de 32 Mo sur disque, ce qui n'est pas négligeable (surtout si vous n'êtes pas votre propre hébergeur).

Votre version de PHP sera au minimum la 4.0.5. PhpWiki fonctionne par défaut avec des "fichiers plats", autrement dit sans avoir recours à un SGBD. Mais il est possible de l'intégrer à un SGBD comme MySQL ou PostgreSQL. Pour y parvenir, les concepteurs font appel à une couche d'abstraction PEAR (des bibliothèques spécialisées pour PHP).



Pour simplifier nous allons ici employer "des fichiers plats".

Configuration nécessaire

La seule étape préliminaire à l'utilisation de phpWiki est l'édition du fichier index.php.

```
// Baptisez votre site (par exemple un pop-up reprenant ce texte apparaît lorsque
// vous pointez votre souris sur l'icône du coin supérieur droit "PHP WIKI")
define('WIKI_NAME', 'Mon site Wiki');
```

```
// Définissez un mot de passe pour l'administrateur
define('ADMIN_USER', "admin");
define('ADMIN_PASSWD', "motdepasse");
```

```
// Si true (vrai) seul l'administrateur peut réaliser des sauvegardes
define('ZIPDUMP_AUTH', true);
```

```
// La taille maximale d'un fichier en upload (ici 16 Mo)
define('MAX_UPLOAD_SIZE', 16 * 1024 * 1024);
```

```
// Vous pouvez choisir un thème
define('THEME', 'MacOSX');
```

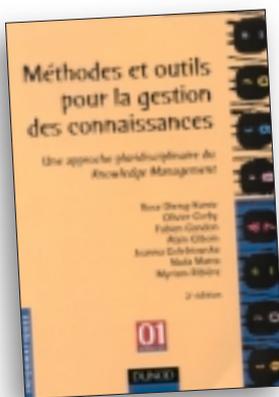
```
// Votre langue sera le français (fr_FR)
$LANG='fr_FR';
```

Le principe

Cliquez sur le bac à sable et éditez celui-ci :

<http://www.votresite.com/phpwiki/index.php/BacASable?action=edit>

Cette zone a été conçue pour qu'un nouveau venu "s'entraîne" à éditer une page. Si vous tapez "Je suis en train d'éditer cette page CeciEstUnLien", puis si vous cliquez sur "prévisualiser", le lien



Méthodes et outils pour la gestion des connaissances

Une approche globale du Knowledge management
Edition DUNOD

PHP

NIVEAU : INTERMÉDIAIRE

CeciEstUnLien apparaîtra avec un point d'interrogation à sa droite (vous arriverez au même résultat en tapant un lien [qqchose] entre crochets).

En cliquant sur le point d'interrogation vous créez une nouvelle page. Si le lien est direct <http://www.qqchose.com> il n'y a pas de point d'interrogation car la page existe déjà (et pas besoin de crochets non plus).

Autrement dit, si vous retournez à la page d'accueil (en cliquant sur l'icône PHP WIKI située en haut à droite), et éditez la page pour y ajouter par exemple le texte :

"La météo aujourd'hui [site meteo prévisions à 7 jours]", un point d'interrogation apparaîtra à droite du lien "site meteo prévisions à 7 jours". En cliquant sur celui-ci une nouvelle page apparaîtra avec comme première ligne :

"Décrivez [site meteo prévisions à 7 jours] ici."

Ajoutez le texte :

"Le site est le suivant <http://www.bdb.be/weer/weerfr.html>". Sauvegardez.

Etudiez la documentation et entraînez-vous pour insérer des images, rédigez en caractères gras ou italiques, etc.

Les opérations de maintenance

Si vous voulez apercevoir l'historique des modifications de votre page d'accueil, rien de plus simple, il suffit de cliquer sur "historique". Vous pourrez passer en revue la première version initiale, la deuxième version (avec le point d'interrogation) et enfin la troisième version sans point d'interrogation (puisque la page lien existe). En cliquant sur "diff" apparaîtront exactement le texte en plus ou en moins d'une version à l'autre...



Il est possible de rechercher des pages semblables, de visualiser les pages les plus visitées, de visualiser les modifications depuis 1, 3, 7, 30 ou 90 jours ; de signer un "livre d'or des derniers visiteurs".

L'administrateur peut réinitialiser le bac à sable ("ratisser le bac à sable"), réaliser une sauvegarde (dump) des pages ou une restauration.

Pour conclure

Mettre en place un serveur PHPWiki est un jeu d'enfant. Ce projet formidable prendra sans aucun doute à l'avenir de plus en plus d'importance. En tous cas le WIKI peut sans problème faire partie de votre arsenal logiciel d'aide au "Knowledge Management". Et comme il ne coûte pas grand chose il serait dommage de s'en passer.

Par Xavier Leclercq

ANALYSE et CONCEPTION: LES DIAGRAMMES UML

“Power AMC et l’ULM ne font plus qu’un”
PowerAMC V8, de Sybase, supportait la conception et l’analyse orientées objet (OOAD) grâce aux diagrammes UML de cas d’utilisation, de séquence et de classes. La version 9.5 de PowerAMC renforce le support du langage UML en intégrant l’ensemble des diagrammes UML : Diagrammes d’objets, de collaboration, d’états, de déploiement, d’activité et de composants. Il améliore de ce fait l’analyse et l’intégration au développement. D. Dichmann, de Sybase, nous présente cette initiation aux diagrammes UML.

UML

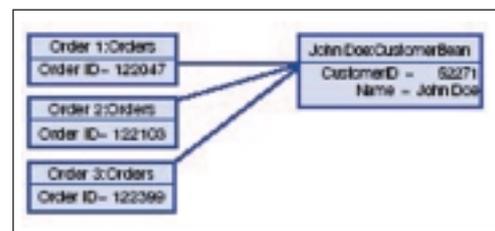
NIVEAU : INTERMÉDIAIRE

Aujourd’hui, nous utilisons des langages objet, tels Java, de nouvelles technologies, telles XML, et nous consolidons, physiquement ou virtuellement, nos bases de données. Nous utiliserons de nombreuses techniques d’analyse et de conception, qu’il s’agisse de la modélisation des processus métiers, de l’analyse orientée objet basée sur l’UML ou de la modélisation Merise des données.

Le diagramme de cas d’utilisation

Le diagramme de cas d’utilisation UML permet de modéliser le contexte du système. Il nous aide à identifier les éléments externes au système/à l’application, les éléments internes, ainsi que la portée du système. En tant que modèle graphique, il est également très utile pour communiquer les spécifications système recueillies. Il est utile aux équipes de développement chargées d’élaborer le produit final, mais aussi aux responsables métiers et aux personnes qui utiliseront le logiciel et qui, selon toute vraisemblance, n’auront pas un profil d’informaticien. Les diagrammes de cas d’utilisation représentent graphiquement la manière dont le système respecte les règles de gestion collectées et répond aux exigences spécifiées par l’utilisateur.

Les diagrammes de cas d’utilisation UML ont également leur utilité à une étape ultérieure du cycle de vie du projet. En effet, les spécifications définies dans un diagramme de cas d’utilisation peuvent servir lors des tests fonctionnels.



Cas d'utilisation d'un client institutionnel passant une commande

Le diagramme de séquence

Le diagramme de séquence UML affine les spécifications et nous permet de commencer à relier les éléments de conception entre eux. Les diagrammes de séquence nous permettent essentiellement de documenter les interactions entre les objets de niveau supérieur et ceux de niveau inférieur. Il est possible de représenter l’interaction entre les acteurs (les mêmes que ceux des diagrammes de cas d’utilisation) et les instances de classe (objets de la mémoire de l’ordinateur — techniques et détaillés).

Les diagrammes de séquence nous permettent de décrire le comportement du système sur une période donnée, en suivant un

ordre précis d'événements (messages) qui constituent un scénario spécifique pour l'utilisation du système. Ils s'appuient sur l'analyse des cas d'utilisation et préparent la conception — ils nous permettent de décider quels objets du système seront en charge de la gestion des tâches définies par les diagrammes de cas d'utilisation, et également quels objets géreront les "conversations" ou communications au sein du système ou avec l'extérieur. A mesure que nous découvrons les messages dans les diagrammes de séquence, nous pourrions décrire quelques-unes des opérations majeures (méthodes) à exercer sur les classes et interfaces (les éléments du code qui seront compilés et déployés au final).

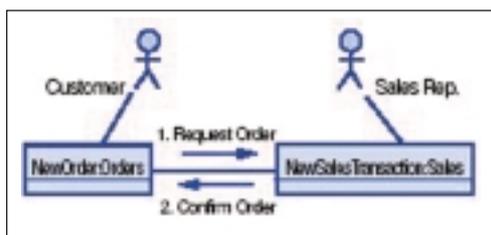


Diagramme de séquence pour une demande de prêt

Le diagramme d'activités

Le diagramme d'activités UML est conçu pour nous aider à comprendre la dynamique des objets de notre système. Ces diagrammes décrivent le comportement d'une classe spécifique, ou d'un petit groupe de classes et leurs interactions. Il est différent du diagramme de séquence en ce sens qu'il ne représente pas une série de communications temporisées mais un transfert de contrôle d'une tâche à une autre, avec l'identification du responsable (objet) des tâches lorsqu'elles se produisent.

Le diagramme d'activités UML est une vue technique des processus métiers. Il peut être dérivé d'une analyse des flux métiers, à la suite peut-être de la modélisation des processus métiers.

Les diagrammes d'activités permettent également d'avoir une vue dynamique détaillée des éléments internes du système (interopération des EJBs, etc.)

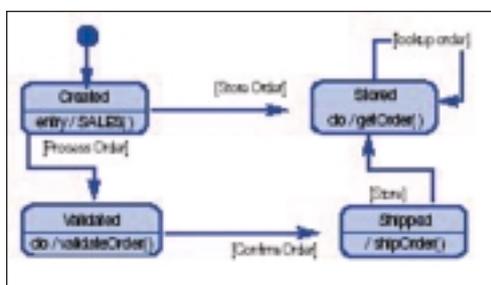


Diagramme d'activités - traitement d'une commande

Analyser pour concevoir

Les modèles d'analyse représentent à la fois la finalité et le comportement attendu du système, indépendamment des modalités de sa mise en oeuvre. A ce stade il n'est pas nécessaire de tenir compte des langages qui seront utilisés, du serveur d'applications qui sera déployé et des EJBs qui seront employés, ou même de savoir si l'architecture retenue sera de type J2EE ou .NET. En revanche, dès que l'on rentre dans la phase de conception, il n'en va plus de même. Il devient alors nécessaire de connaître les détails de l'environnement de production et de construire le logiciel en fonction d'une architecture spécifique. La conception est la définition de la mise en oeuvre du système.

En faisant reposer la conception sur les informations fournies par l'analyse, on s'assure que le comportement du système qui est en cours de codage est exact, et que le développement sera en mesure de produire un système en parfaite adéquation avec les besoins.

Que nous apporte donc l'analyse? **Les diagrammes de cas d'utilisation** permettent d'identifier les objets et conduisent à la création des classes et des interfaces.

Une ou plusieurs classes ou interfaces peuvent mettre en oeuvre un même acteur, et nous pouvons créer directement les classes et les interfaces dans la définition de l'acteur. Il nous est également possible de relier un acteur à une classe ou une interface existante, montrant comment une partie du code peut servir à différents éléments de l'analyse.

Les **diagrammes de séquence** permettent d'identifier les méthodes et de guider la création d'opérations sur une classe. S'il est nécessaire d'envoyer un message à une classe, il est possible d'appeler une méthode sur cette classe. Le message du diagramme de séquence peut être utilisé pour créer automatiquement une opération ou relier à une opération existante.

Quelles sont les tâches incluses dans la conception ?

La phase de conception consiste à définir, après avoir identifié les besoins, la façon dont sera construit le système. Il est à ce stade nécessaire de décider de la façon dont sera exprimée la logique applicative: elle peut prendre la forme de composants, tels qu'un EJB dans un serveur d'applications; il peut s'agir d'une classe ou d'un composant faisant partie d'une application client utilisant un langage tel que VB ou PowerBuilder, ou bien encore être traduite au sein d'une base de données relationnelles par des triggers et des procédures. Des choix doivent alors être faits en fonction des besoins identifiés et des spécifications définies (le système doit-il être évolutif, sécurisé, axé principalement sur les performances,

dans quelle mesure doit-il être accessible, etc...). Les diagrammes de classes et de composants UML permettent de définir les structures détaillées, techniques et statiques de notre système.

Le diagramme de classes

Le diagramme de classes UML permet de définir l'intégralité de la structure du code, en intégrant la logique applicative et toutes les structures nécessaires. Dans la mesure où ce diagramme modélise le code tel qu'il est utilisé en développement, il est clairement une représentation abstraite d'un langage objet tel que Java ou PowerBuilder. Nous pouvons également utiliser le diagramme de classes UML pour représenter de façon abstraite des structures complexes rédigées en XML, en les rendant ainsi plus simples à développer et à comprendre.

Le diagramme de classes UML peut être utilisé comme source pour générer du code. Ce code est modifiable lors du développement afin de compléter, tester et déployer une application finale. Dans la mesure où la capacité de mise en correspondance est de 1:1 entre les langages objet et le diagramme de classes UML, il est également possible de rétro-concevoir le code, lire les fichiers source et de les utiliser pour créer un nouveau diagramme de classes. Cette façon de procéder permet de mieux comprendre les systèmes existants et de rationaliser les efforts d'intégration et de maintenance.

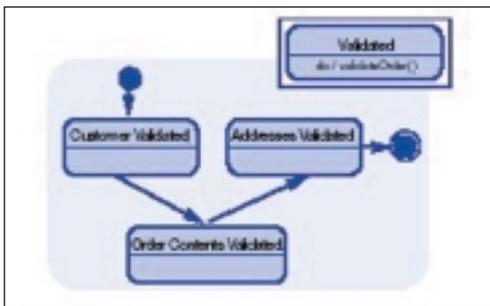
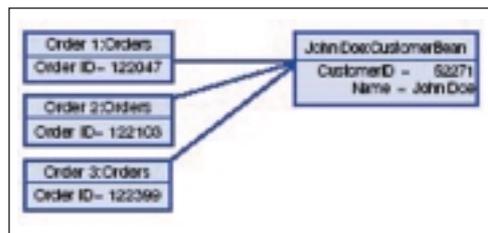


Diagramme de classes pour un système de saisie de commandes

Le diagramme de composants

Le diagramme de composants UML est utilisé pour décrire les définitions et les dépendances des objets de niveau supérieur, à partir d'une vue élargie et sans s'encombrer de détails techniques superflus à ce stade. Il s'agit toujours d'un modèle de conception et d'une représentation abstraite directe du code. Par exemple, le symbole de composant d'un EJB est directement relié aux classes et aux interfaces nécessaires à sa mise en œuvre et il générera le code nécessaire pour conduire le développement du bean achevé. Le diagramme de composants est censé être plus facile à comprendre et à gérer qu'une vue de niveau code pour les architectures de composants. Il améliore également le partage et la

réutilisation du code — en documentant les interfaces des composants, il n'est pas nécessaire de comprendre l'implémentation d'un composant pour l'utiliser dans d'autres projets et systèmes.



Modèle d'un EJB pour un bean d'une entité cliente

David Dichmann

Directeur Produit, PowerAMC, Sybase, Inc.

L'INGENIERIE INTERACTIVE PAR VA-ET-VIENT

Vos projets informatiques, pas plus que le monde, ne sont statiques. Le scénario classique de l'ingénierie par va-et-vient implique une modification du code en cours de développement qui doit être répercutée dans le diagramme de classes :

1. Création d'un diagramme de classes et ajout d'éléments de la logique applicative au modèle
2. Génération du code d'application dans le système de fichiers
3. Edition du code dans un IDE ou un éditeur de texte
4. Modification de la conception, sans tenir compte des modifications apportées au code généré
5. Rétro-conception des modifications dans le diagramme de classes existant
6. Synchronisation (fusion) des modifications apportées à la conception et les modifications effectuées lors du développement
7. Génération du nouveau code qui est le résultat des modifications apportées à la conception et par le développeur.

L'utilisation de la synchronisation/fusion prémunit contre le risque de perte du travail effectué par un développeur lorsque des modifications sont apportées au diagramme de classes afin de répercuter un nouveau concept de conception, tout en permettant au concepteur d'accepter ou de refuser les modifications apportées en phase de développement.

Certaines modifications sont à ce stade obsolètes en raison des changements effectués dans la conception. Ce type de politique permet aux services informatiques de conserver un contrôle total sur l'architecture, et c'est ce contrôle qui constitue l'une des clés du succès.

Mais, là ne s'arrêtent pas les possibilités d'un outil comme PowerAMC. Une fois les modèles de conception mis à jour, il est possible de répercuter ces changements sur l'analyse. Plus vraisemblablement, il est toujours possible d'identifier un nouveau besoin dans le cadre de l'analyse et à partir de sa définition pour modifier la conception et le code en conséquence. Grâce à un outil tel que PowerAMC, et à ses fonctionnalités de comparaison/fusion innovatrices, il est réellement possible de synchroniser par "va-et-vient" tous les modèles à toutes les phases du cycle du développement.

SeaSide : Des applications web complexes très simplement...

SeaSide est un puissant framework pour développer des applications web complexes. La fonctionnalité la plus remarquable et unique de SeaSide réside dans son approche de la gestion de sessions.

Contrairement aux modèles *servlets*, qui nécessitent un handler pour chaque page ou requête, SeaSide représente l'entière session, comme un morceau de code avec un *flot de contrôle linéaire* et naturel : les pages s'appellent les unes les autres comme des invocations de méthodes, les formulaires sont de simples objets, les objets sont passés comme des références, plus besoin de les tronçonner en Urls ou champs cachés, et tout ceci avec la possibilité de retour en arrière et la concurrence inhérente aux browsers webs et du bouton de retour en arrière.

SeaSide offre aussi un modèle d'événements basés sur des *callbacks*, la gestion des pages devant s'autodétruire, une approche simple de la génération d'HTML, un ensemble de composants réutilisables, une gestion des templates pour designers, et un ensemble d'outils de développement.

Dans cet article nous allons faire nos premiers pas en SeaSide en montrant comment un simple jeu peut être porté en SeaSide. Dans l'article suivant, nous aborderons l'architecture de SeaSide et son modèle de composants, ainsi que la possibilité de *backtrack*, des sujets d'un niveau plus avancé.

Obtenir et démarrer SeaSide

Vous devez avoir une version récente de Squeak (3.2 ou 3.4) www.squeak.org, le serveur web Comanche 5.0 pour Squeak <http://fcs.cc.gatech.edu/~bolot/kom/kom-5-0-02May1824.cs.gz> et la dernière version de SeaSide (2.21) disponible à <http://beta4.com/seaside2/downloads/latest.html>. Chargez Comanche et SeaSide dans Squeak (open File List...). Pour démarrer Comanche configuré pour SeaSide, exécutez l'expression `WAKom startOn: 9090`. SeaSide doit maintenant écouter des requêtes sur le port 9090. Notez que si vous sauvez votre image, le service sera automatiquement lancé. Pour vérifier que SeaSide fonctionne bien ouvrez la `http://localhost:9090/seaside/counter` à votre navigateur. Vous devez arriver sur un exemple très simple: un compteur, comme montré par la figure 1. Cliquez sur les "+" et "-". Si SeaSide vous demande un mot de passe, donnez : `seaside/admin`.

Un simple jeu de devinettes

Nous allons implémenter un sujet de devinettes qui fonctionne de la manière suivante : (1) le système demande à l'utilisateur de penser à un fruit, (2) l'ordinateur pose une série de questions pour identifier le fruit. Quand l'ordinateur a cerné le fruit en question il propose une solution. (3) Si c'est le bon fruit, le jeu est fini. Si ce n'était pas le bon fruit, il demande à l'utilisateur le nom du fruit ainsi qu'un indice qui lui permettra d'identifier le fruit. Ces informations sont ajoutées dans une base de données pour utilisation future. L'idée n'est pas de faire un cours sur les jeux de devinettes,

donc voilà la logique interne du jeu. Les indices et les fruits sont représentés comme les noeuds d'un arbre binaire étiqueté. Chaque indice est à la tête d'un arbre, dont les branches représentent les réponses positives ou négatives de l'utilisateur. En partant de la racine de l'arbre, l'ordinateur va donc traverser l'arbre, poser les questions basées sur les indices et naviguer dans l'arbre, en suivant les réponses de l'utilisateur. Lorsqu'il arrive dans une voie terminale, soit il a trouvé le bon fruit, soit il doit insérer un nouveau noeud contenant un fruit et indice.

Comme Squeak ne propose pas d'arbre binaire dans sa distribution, nous allons en définir un. Il a besoin d'une étiquette et des deux branches. Comme nous ne le définissons que pour ce jeu, appelons les keys, yes et no. Définissez la classe FruitTree comme suit :

```
Object subclass: #FruitTree
  instanceVariableNames: 'key yes no '
  classVariableNames: ''
  poolDictionaries: ''
  category: 'Seaside Game'
```

Définissez les accesseurs key: aString, key, yes: aTree, yes, no et no: aTree, sur le modèle :

```
FruitTree>>key: aString
  key := aString

FruitTree>>key
  ^ key
```

Quand on insère un nouvel indice et un fruit, nous voulons insérer l'indice en premier avec la branche yes de l'indice pointant sur le nouveau fruit. Pour nous faciliter la tâche, nous définissons donc la méthode de classe newFruit:withClue : définie comme suit :

```
FruitTree class>>newFruit: aFruitString withClue: aClueString
  ^ self new
    key: aClue;
    yes: (self new key: aFruitString)
```

Pour commencer le jeu, nous avons besoin d'une racine que nous définissons ici comme une variable globale pour plus de simplicité. Évaluez dans un workspace le code suivant. Bien sûr une aubergine n'est pas un fruit mais c'est le symbole de SeaSide. Le terme SeaSide (Squeak Enterprise Aubergines Server) vient d'une caricature des Beans.

```
FruitRoot := FruitTree newFruit: 'an eggplant' withClue: 'purple'.
```

Maintenant nous devons écrire la logique du jeu. Nous l'avons défini d'un seul bloc, de façon à ce que l'on puisse l'exécuter dans un workspace. Ouvrez un workspace et tapez le code suivant :

```
I node response next!
Object inform: 'Think of a fruit.'.

node := FruitRoot.
[response := Object confirm: 'Is it ', node key, '?!'.
 next := response ifTrue: [node yes] iffFalse: [node no].
 next notNil]
  whileTrue: [node := next].

response

  ifTrue: [Object inform: 'Got it!']
  iffFalse:
    [I fruit clue!
     fruit := FillInTheBlankMorph request: 'Give the name of your fruit'.
     clue := FillInTheBlankMorph request: 'What makes it different from
     other fruits?'.
     node no: (FruitTree newFruit: fruit withClue: clue)].
```

Voilà une implémentation assez simple de la logique du jeu telle que nous l'avons décrite plus avant: on traverse l'arbre jusqu'à ce que l'on arrive à une branche terminale. A ce point, si la réponse est vraie, on a trouvé le fruit, sinon on crée un nouveau noeud dans l'arbre. Exécutez cette expression plusieurs fois de suite pour voir l'arbre grandir. Vous pouvez ensuite essayer FruitRoot explore, pour le visualiser.

Pour le web

Ce jeu, somme toute insignifiant, 15 lignes de code dans un workspace, l'est beaucoup moins quand il s'agit de l'implanter, pour obtenir une application web avec des moyens traditionnels. Si vous avez déjà développé des applications web, prenez un moment pour imaginer comment vous vous y prendriez. En premier lieu, vous devriez transformer l'algorithme sous sa forme actuelle itérative, en une forme récursive, liant chaque réponse aux questions yes/no à la suivante. Vous devriez penser aussi aux appels à la widget FillInTheBlankMorph à la fin: Peut-être pourriez vous utiliser la même page pour les deux appels, comment savoir ou rentrer l'information? Peut-être que vous devriez paramétrer la page avec la suivante? Ect... Bref un enfer pour un jeu aussi ridicule donc, imaginez pour une véritable application ! En SeaSide, toutes ces questions, qui ont de l'importance dans d'autres systèmes, n'en ont aucune. Sans rentrer dans les détails que nous aborderons dans l'article suivant, nous allons transformer notre jeu. Tout d'abord, nous créons une nouvelle sorte de composant web, que nous appelons FruitGuessing, comme suit:

```
WAComponent subclass: #FruitGuessing
  instanceVariableNames: ''
  classVariableNames: ''
  poolDictionaries: ''
  category: 'Seaside Game'
```

Ensuite nous définissons, comment le composant doit s'afficher dans le navigateur web, en définissant la méthode renderContentOn :

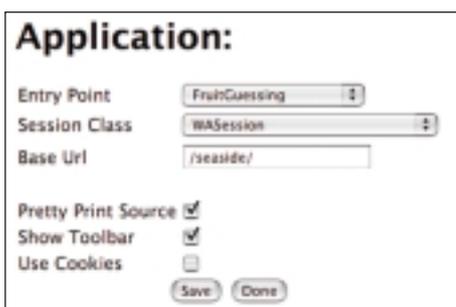
```
FruitGuessing>>renderContentOn: html
html form: [html submitButtonWithAction: [ self play ] text: 'Play' ]
```

Ici, cette méthode crée simplement un bouton ayant comme étiquette 'Play' (Figure 1) et comme action, d'invoquer la méthode play que nous allons définir plus loin.



> Figure 1: L'interface ultra minimaliste de notre jeu.

Nous allons enregistrer notre application (notez que nous pourrions le faire en exécutant l'expression `FruitGuessing registerAsApplication: 'fruit'`) en utilisant la page de configuration de SeaSide. Dans le navigateur, allez sur la page <http://localhost:9090/seaside/config> (le login est seaside/admin) vous devez alors arriver sur une page contenant la liste des applications démos. Entrez le nom de votre application, par exemple 'Fruit', dans le champ et cliquez sur le bouton add et sélectionnez la classe que vous venez de créer, comme montré dans la Figure 2.



> Figure 2: configuration de l'application

Une fois ceci fait, vous devez pouvoir lancer votre application, soit depuis le panneau de configuration en cliquant sur l'élément de la liste, soit en allant sur la page <http://localhost:9090/seaside/fruit>. Vous devez obtenir la Figure 1. Si vous pressez sur le bouton, vous allez invoquer une méthode non définie, donc vous allez obtenir la trace de l'erreur. Il nous faut donc convertir notre script en une méthode. En SeaSide rien de plus simple. Copiez le code du workspace dans l'éditeur, changez Object et FillInTheBlankMorph par self, comme suit :

```
FruitGuessing>>play
| node response next |

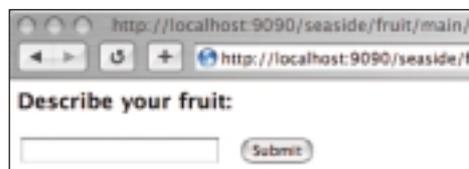
"Initialize the game"
node := FruitRoot.
self inform: 'Think of a fruit!!'.

"Walking down the tree"
```

```
[ response := self confirm: 'Is it ', node key, '?' .
  next := response
  ifTrue: [ node yes ]
  ifFalse: [ node no ].
  next isNil not ] whileTrue: [ node := next ].

"Update the tree"
response
ifTrue: [ self inform: 'Got it!! ]
ifFalse: [ | fruit clue |
  fruit := self request: 'Describe your fruit:'.
  clue := self request: 'What makes it different form other fruit?'.
  node no: (FruitNode newFruit: fruit withClue: clue) ]
```

Et voilà, votre application fonctionne. Cliquez sur le bouton play et jouez. Comme vous le voyez, vous n'avez pas eu à transformer votre application, ni à changer de paradigme.



Objets et Continuations

Une application en SeaSide est un ensemble d'objets se passant le contrôle de l'application et ayant la responsabilité de gérer leur affichage et leurs entrées. Comme nous le verrons dans le prochain article, cela n'est pas anodin en présence du bouton de retour des navigateurs. Contrairement à la plupart des autres frameworks pour le développement d'applications webs, SeaSide intègre le cycle requete/reponse des applications web, avec le flot de contrôle illimité offert par Smalltalk. Ainsi les boucles, les conditions, les appels et retours de méthodes peuvent tous créer de multiples pages web indépendantes. A titre d'exemple, notez que l'outil de configuration, ainsi que tous les dialogues, sont des composants SeaSide. Essayez le browser de code en cliquant sur le Lien "Browse". Il est aussi développé en SeaSide de la manière la plus simple possible. L'idée est que SeaSide utilise la notion de continuation des langages fonctionnels, ce qui lui permet de conserver des points dans l'exécution d'un programme, pour potentiellement y revenir. Bien sûr, SeaSide offre des abstractions sur ce modèle et le programmeur n'a pas à connaître la logique interne de SeaSide, il doit juste utiliser les abstractions offertes. Dans l'article du mois prochain, nous détaillerons le modèle de SeaSide, ainsi que la manière dont le flot de contrôle est géré. Nous présenterons les messages spécifiques qui offrent à SeaSide ce confort d'utilisation et ce pouvoir d'expression aussi puissant et naturel qu'inattendu. ■

Références

Squeak: <http://www.squeak.org/> - SeaSide: <http://beta4.com/seaside2/> - ESUG: <http://www.esug.org/> - Wiki Francophone: <http://www.iutc3.unicaen.fr:8000/fsug/>
Livres Gratuits :
<http://www.iam.unibe.ch/~ducasse/WebPages/FreeBooks.html>
ESUG CD : <http://www.ira.uka.de/~marcus/Esugcd.html>
Squeak CD : <http://www.ira.uka.de/~marcus/SqueakCD.html>