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

DÉVELOPPEUR RÉFÉRENCE

LA LETTRE BIMENSUELLE DU DÉVELOPPEMENT

www.devreference.net

v2.12 | 23 avril 2002

EDITO

Dans la lignée des comparatifs J2EE vs .NET, notre ami Sami Jaber de DotNetGuru.org nous offre une étude de la persistance dans les architectures n-tiers, en comparant cette fois-ci Java Data Object de Sun et ObjectSpaces, le futur framework de persistance pour .NET de Microsoft. Eh oui, .NET ne gère pas (encore) la persistance. Dans ce domaine, Java a pris une certaine avance, mais Microsoft persiste...

Pour Bjarne Stroustrup, le C++ reste encore un langage sous-estimé et mal exploité. Découvrez le futur de C++ avec le concepteur de ce langage dans une interview qu'il nous a accordée lors de la conférence OCM 2002 à Nantes, conférence sur laquelle nous faisons un petit retour.

CA-world, IBM Developer Works, Microsoft TechEd... les mois à venir seront riches en événements que nous ne manqueront pas de suivre.

Pierre Tran
Rédacteur en chef

La couche de persistance ou de données est sûrement la partie la plus stratégique et la plus complexe d'un projet. C'est à cet endroit qu'est confiné l'ensemble des entités de l'application au cœur du métier de l'entreprise (Facture, Comptes…). Mais c'est aussi et surtout dans cette couche que surviennent la plupart des problèmes liés aux performances : goulets d'étranglement, requêtes longues, transactions…

A travers les années et avec l'apparition des architecture n-tiers, cette couche a considérablement évolué pour aujourd'hui laisser place à une approche véritablement objet. Il est bien connu que le monde relationnel et le monde objet ne font pas très bon ménage, et ce pour diverses raisons. Non seulement la granularité des tables n'est pas nécessairement équivalente à celle des objets, mais l'approche et la conception orientée données d'un côté ne trouve pas d'analogie formelle de l'autre. Bref, tout cela se passe comme si SGBDR et Objets devaient coexister malgré leur différences.

[suite page 9]

Sommaire

DÉVELOPPEUR RÉFÉRENCE

LETTRE BIMENSUELLE ÉDITÉE PAR
IDG COMMUNICATIONS FRANCE
5, rue Chantecoq
92808 Puteaux Cedex
Tél. : 01 41 97 61 61
Adresse électronique :
devreference@idg.fr

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

Rédaction

Editeur
Michel Crestin
Rédacteur en chef
Pierre Tran • tran@idg.fr • 6257
Assistantes de rédaction
Kateline Renaudin
Ont collaboré à ce numéro
Jean-Louis Bénard, David Castaneira, Sami Jaber, Frédéric Milliot

Fabrication

Directeur artistique groupe
Patrice Servage
Mise en page
Pierre Tran
Photographies
Marc Guillaumot
Iconographie
Nouara Aftis
Infographies
Carole Guirrado, Pierre Tran

Publicité

Pasquale Meyer • meyer@idg.fr • 6216
Caroline Mayer • mayer@idg.fr • 6217

Service Abonnements
Développeur Référence
BP 90006 – 59718 Lille Cedex 9
Tél. 03 20 12 11 17
Fax 03 20 12 86 09
Tarif :
1 an 22 numéros : 200 euros
Prix de lancement : 150 euros

Renseignements :
Lucienne Bosser • bosser@idg.fr • 6128

Les news

Evénements | Tendances | Nouveaux produits

Retrouvez chaque jour les news sur www.devreference.net

IBM, VeriSign et Microsoft s'allient pour sécuriser XML

Microsoft, IBM et VeriSign s'associent pour sécuriser les services Web en développant WS-Security, norme fondée sur XML. La spécification doit être présentée au W3C et pourrait être intégrée à l'ensemble de recommandations XML sur le cryptage et la signature électronique.

Les trois associés prévoient la publication d'autres spécifications XML liées à la sécurité d'ici 12 à 18 mois :
- WS-Privacy pour la sécurisation des données personnelles
- WS-Secure Conversation pour la gestion d'échanges authentifiés
- WS-Policy pour les procédures et politiques de sécurisation
- WS-Trust pour les données relationnelles
- WS-Federation pour l'interopérabilité d'environnements sécurisés hétérogènes
- WS-Authorization pour la gestion des droits dans les services Web.

D.C.

IBM sécurise sa boîte à outils

IBM lance la version 3.1 de Web Services Toolkit (WSTK), sa boîte à outils de constitution de services Web.

Les améliorations concernent essentiellement la sécurité. Après avoir développé un module d'authentification en janvier, Big Blue s'attaque à la signature électronique en intégrant le jeu de spécifications WS-Security. Proposé conjointement avec VeriSign et Microsoft, ce dernier est en cours d'homologation par le W3C, chargé de normaliser le Web.

Par ailleurs, WSTK 3.1 supporte le module d'authentification Soap Security Token. Il permet l'authentification de l'émetteur du message par l'enregistrement de son identité, sa fonction et de ses droits. Selon IBM, Soap Security Token permet, de plus, l'interopérabilité de systèmes d'authentification différenciés.

Enfin, WSTK 3.1 intègre un module d'exploration de l'annuaire UDDI, le support de WSDL et un outil de gestion d'interfaces Java.

Une version d'essai est téléchargeable sur le site d'IBM.
D.C.

http://www.alphaworks.ibm.com/

Sun fédère ses applications sous la marque Sun ONE

Sun Microsystems unifie et rebaptise l'ensemble de sa gamme d'applications. Objectif : rassembler toutes les lignes de produits autour de Sun ONE (Open Net Environment), son programme en matière de services Web, concurrente de .Net de Microsoft. Du coup, les suites iPlanet (serveur d'applications et outils de constitution de portail), Forté (middleware d'intégration et outils de développement) et Chili!Soft (outil d'utilisation de modules dynamiques ASP) disparaissent tandis que Star Office (bureautique) connaît un répit et se voit simplement adjoindre la mention Sun ONE.
D.C.

HP renoncerait à Netaction pour adopter WebLogic

HP pourrait abandonner Netaction, sa stratégie logicielle fondée sur le serveur d'applications Bluestone, pour s'orienter vers un accord avec BEA, numéro un du marché avec WebLogic. Alfred Chuang, PDG et fondateur de BEA, a affirmé au cabinet d'étude Merrill Lynch que HP avait "sabordé Bluestone" et avait "adopté WebLogic". Depuis deux ans, HP a pourtant redoublé d'effort pour combler son retard commercial sur IBM et BEA qui s'accaparent le marché des serveurs d'applications avec 34% de part chacun. Du coup, Total-e-Server - nom donné à Bluestone dans le cadre de Netaction - a gagné trois points en 2001 pour atteindre prés de 7% de part de marché, à égalité avec iPlanet de Sun. En vain. Selon Mike Gillipin, analyste au Giga Group, la logique de réduction des dépenses dans le cadre de la fusion HP/Compaq conduit HP à se séparer d'une ligne de produits qui lui coûterait près de 20 M$ par an.
D.C.

Les services Web interentreprises peu fiables sans ebXML selon Sun

Marty Robins, gardien du temple Java chez Sun Microsystems, affirme qu'aucun service Web interentreprises n'est fiable tant qu'il n'intègre pas les spécifications d'ebXML. Selon lui, les fondements des corpus XML, de l'annuaire UDDI , de WSDL et de Soap sont insuffisants. Ils ne garantissent pas sécurité et qualité de service.

Marty Robins s'en prend indirectement à Microsoft, soupçonné de vouloir reléguer ebXML aux oubliettes en imposant la plate-forme de normes pour les services Web dont le WS-I, consortium créé récemment en compagnie d'IBM et de BEA, assure la promotion.

Programme de normalisation des échanges de données interentreprises, ebXML reprend les fondements développés par le consortium RosettaNet, qui se consacre aux échanges dans le secteur électronique. Il ambitionne cependant de gérer l'ensemble des procédures d'échanges électroniques entreprises.
D.C.

Improve C# plugin for Eclipse

[Vu sur application-servers] Ce plug-in, développé par Improve, permet de faire de l'édition et de la compilation de code C#, ainsi que de spécifier pour chaque fichier source les arguments à passer au compilateur C#. Son but est d'offrir un outil simple, pour découvrir et programmer en C#, mais assez évolué pour créer des assemblages (composants dans .NET).

http://www.application-servers.com/links.do?reqCode=showLink&lid=1013

Evénement

OCM 2002

Objet, Composants et Modèles

Comme chaque année depuis 1995, s'est déroulée la conférence OCM (Objet, Composants, Modèles) à Nantes, le 21 mars dernier. Cette journée de rencontres est l'occasion de faire le point sur les innovations en matière de développement logiciel.

Reportage de Pierre Tran

Cette année, OCM s'est focalisée sur l'évolution des architectures logicielles et des méthodes de développement. Organisée conjointement par l'Ecole des Mines de Nantes, l'IRISA, la MEITO et Sodifrance, cette manifestation est l'occasion d'inviter des personnalités prestigieuses du monde du développement, telles que Bertrand Meyer en 2001 ou Patrick Albert en 1999. L'édition OCM 2002 pour sa part voit la participation de Bjarne Stroustrup, inventeur du langage C++ et de Steve Cook, expert de l'ingénierie des modèles.

Redorer C++

La journée a débuté par la présentation de Bjarne Stroustrup, "Des clefs pour une utilisation efficace du langage C++ par son inventeur". L'objectif était de montrer que C++ est un langage multi paradigme, c'est-à-dire qu'il permet différents styles de programmation tels que la programmation orientée objet ou la programmation générique. C++ reste le langage généraliste le plus utilisé au monde, mais il est souvent sous-exploité. Bjarne a donc montré des exemples concrets des différents styles de programmation avec le C++ standard ISO, et comment les combiner pour produire du code plus propre et plus maintenable que s'il avait été écrit dans un seul style. Face à la popularité croissante de Java et à l'émergence de C#, le C++ avait bien besoin de redorer un peu son blason. Débuter la journée par cette présentation ultra technique était, il faut l'avouer, un peu sévère, mais les spécialistes de ce langage ont pu apprécier les concepts présentés. Nous avons rencontré Bjarne et vous trouverez plus loin le compte-rendu de l'interview qu'il nous a accordée.

Deuxième temps fort de la journée, Steve Cook est venu nous présenter l'approche MDA (Model Driven Architecture), l'architecture dirigée par les modèles. Steve a rejoint IBM en 1994. Il est co-inventeur du langage de contrainte OCL (Object Constraint Language), a contribué à la définition d'UML et du standard de processus associé SPEM (Software Process Engineering Model). Steve nous a d'abord présenté l'historique de l'abstraction dans le développement logiciel, depuis la programmation orientée objet début des années 80, jusqu'aux composants J2EE et .NET et les services Web d'aujourd'hui. Face à la complexité croissante des développements et des architectures, l'utilisation de modèles permet de définir une hiérarchie des abstractions, c'est l'objet de MDA. Steve a ensuite présenté la structure et les relations entre les modèles et les langages de programmation, et leur utilisation dans le développement logiciel, avec un focus particulier sur UML et les technologies associées : MOF (Meta-Object Facility), XMI (XML Metadata Interchange), CWM (Common Warehouse Metamodel)… A l'inverse de la précédente, cette présentation fut très conceptuelle et a permis donner un aperçu du futur dans le domaine de l'ingénierie logicielle.

L'après-midi fut consacrée à différentes sessions en parallèle portant sur les architectures et les technologies de développement : plate-forme Java, plate-forme .NET, MDA et plate-forme XML/services Web. Chaque session comprenait une présentation technologique et une présentation plus pratique (retour d'expérience, mise en œuvre). La session Java présentait le projet Eclipse, la plate-forme universelle pour outils intégrés et un retour d'expérience sur J2EE avec la plate-forme e-Commerce Smart-EC. La session .NET faisait tout d'abord le point sur cette plate-forme et offrait un retour d'expérience sur le développement d'un site Web en .Net avec une approche génie logiciel. La session MDA débutait par une présentation générale de l'approche MDA et se poursuivait par une migration vers Java avec une approche basée sur MDA. Enfin la session XML/services Web présentait la technologie XML appliquée à la télévision numérique et une mise en œuvre de services Web avec .NET.

Le nouveau paradigme du génie logiciel

Pour ma part, j'ai suivi la session MDA et je n'ai pas été déçu. Jean Bézivin, de l'Université de Nantes, a présenté avec clarté et enthousiasme l'approche MDA, ses concepts, ses challenges et ses opportunités. Une présentation qui vient compléter à merveille celle de Steve Cook, car beaucoup plus pragmatique. Elle s'appuie sur le récent changement de paradigme en génie logiciel : de la technologie des objets vers la technologie des modèles. Jean n'hésite pas à piétiner au passage quelques idées reçues sur le tout-objet (abstraction, portabilité, réutilisabilité, extensibilité…) et quelques technologies passées ou émergentes : les composants, la programmation orientée aspect, la guéguerre des langages et des plates-formes… Point important, l'approche modèles n'est pas qu'une vision conceptuelle élaborée par un comité, mais un approche que réclament les entreprises qui en ont assez de payer des migrations de leur système d'information vers des plates-formes à obsolescence rapide (COM, Java, Corba, HTML, XML, .NET…). Elles veulent pouvoir construire une bonne fois pour toutes des modèles abstraits de leur processus métier et exiger de tout nouveau fournisseur de plate-forme qu'il livre les outils de transformation pour ses modèles métier. Bref, la technologie des modèles semble être en mesure de répondre à tous ces défis (gestion séparée des aspects, prise en compte du fonctionnel et du non-fonctionnel, intégration des règles, services, processus, architecture…) et offrir une voie d'évolution progressive aux objets et composants actuels. Nous aurons l'occasion de revenir en détail sur l'approche MDA dans les prochains numéros de Développeur Référence.

La journée s'est terminée sur une table ronde autour du thème "Des modèles objets vers des modèles de services", l'occasion de faire le point notamment sur la technologie émergente des services Web.

Une journée au final fort enrichissante. Et le cadre excentré de Nantes ne semble pas avoir posé un obstacle à la centaine de participants. Rendez-vous est pris pour la prochaine édition qui sera nous l'espérons toute aussi captivante.

Site de l'OCM :
http://www.ocm-ouest.org

Interview

Bjarne Stroustrup

L'inventeur du C++ évoque la standardisation du langage, le multi paradigme, C# et l'évolution de la future version du langage.

Propos recueillis par Pierre Tran
avec la collaboration de Guillaume Louel

Auteur de plusieurs ouvrages sur le langage C++ ("The C++ Programming Language", "The Design and Evolution of C++"), Bjarne Stroustrup dirige actuellement le département de programmation grande-échelle au laboratoire d'AT&T Bell. C'est à l'occasion de OCM 2002 (voir page précédente) que nous avons rencontré l'inventeur du langage le plus utilisé.

Développeur Référence : La standardisation ANSI/ISO de C++ a été un long processus qui ne s'est achevé qu'en 1998. Ne regrettez-vous pas que tout cela ne se soit pas fait plus rapidement ?

Bjarne Stroustrup : Bien sûr, cela aurait été super si le standardisation C++ s'était achevée plus tôt, mais les processus de standardisation de C++ ne prennent pas vraiment plus de temps que pour les autres standards. Les formalités de la standardisation prennent plus de temps que ce l'on pense, parce qu'il y a des tas de gens et d'organisations à rencontrer.

Pensez-vous que cela ait nuit à l'image et à la pénétration de C++ ? Dans l'enseignement par exemple, nombreux sont ceux qui continuent à enseigner C en martelant que C++ n'est toujours pas standardisé... [c'est du vécu !] ?

Beaucoup de personnes ont enseigné - et certains continuent à enseigner - le C++ soit comme un langage de bas niveau en se focalisant sur les fonctionnalités partagées avec le C, soit comme un langage pour exprimer les hiérarchies de classe. Aucune de ces deux approches ne réussit à mettre en valeur les grandes forces de C++. Pire : de telles approches perdent souvent tellement de temps sur des aspects de C++ peu gratifiants pour les programmeurs qu'elles échouent sur l'utilisation efficace de C++. Les conteneurs et les algorithmes de la bibliothèque standard, l'utilisation des exceptions dans la gestion des ressources sont des exemples des sujets clefs qui sont souvent négligés, ou considérés à tort comme difficiles.

Je pense que de tels échecs dans l'enseignement ont fait plus de tort à C++ que la lenteur du processus de standardisation ISO. Clairement, cela aurait été beaucoup mieux si nous avions rendu disponible la bibliothèque standard en 1985, mais nous ne l'avons pas fait. Et ensuite nous n'avons simplement pas su comment concevoir et mettre en œuvre quelque chose de généraliste, élégant et efficace.

Il est en effet très triste d'entendre les gens proclamer que C ++ n'est pas standardisé. La norme ISO a été ratifiée en 1998 et aucune nouvelle fonctionnalité significative n'a été ajoutée au langage ni à la bibliothèque standard depuis 1996. En informatique, c'est une longue période de temps.

Pensez-vous que cela ait ralenti l'évolution de C++ : une meilleure évolution des bibliothèques standard, la prise en compte de l'informatique distribuée... ?

Concernant la vitesse de l'évolution, c'est limité par notre compréhension des problèmes et - je dirai même plus - par notre capacité à enseigner la bonne utilisation du langage. Je ne pense pas que le langage puisse ou doive se développer plus rapidement qu'il ne le fait. Des langages expérimentaux peuvent se développer rapidement parce qu'ils ont peu d'utilisateurs. Un langage courant comme le C++ ne peut pas bouger plus vite que sa communauté d'utilisateurs et doit s'attacher à la stabilité.

Pour ce qui est des prochaines versions de C++, vous semblez privilégier une évolution des bibliothèques standard plutôt qu'une évolution de la syntaxe du langage. Pouvez vous nous expliquer cette position ? Quelles sont les bibliothèques que vous voudriez voir standardisées : multithreading, processus distribués ? Faut-il aller aussi loin que Java et proposer une bibliothèque standard de GUI ?

Ma proposition est que le travail sur la prochaine version du standard C++ ISO se concentre sur la bibliothèque standard. Le langage est déjà assez puissant pour exprimer la majorité de ce que nous avons à exprimer et la stabilité est le point le plus important. Par conséquent, je propose que nous soyons conservateurs et prudents avec les extensions du langage, pour nous concentrer sur la généralisation et des choses qui peuvent aider l'enseignement plutôt que sur des nouvelles fonctionnalités majeures. D'un autre côté, nous devons être agressifs et opportunistes quand il s'agit des extensions de la bibliothèque standard.

Je pense que des bibliothèques pour rendre C++ meilleur en tant qu'outil pour le développement de systèmes est la clé : la gestion des ressources, le threading, la distribution physique, TCP/IP, etc. sont des candidats évidents. Nous verrons aussi presque certainement des améliorations utiles telles que le matching de hash_maps et de pattern. Je crains que le GUI ne soit un sujet trop complexe et trop controversé pour le comité de standardisation. Le comité est constitué de volontaires et nous n'avons pas les ressources pour construire une bibliothèque GUI. Par ailleurs, un comité de standardisation ne peut rivaliser avec des sociétés commerciales (et non commerciales). Il doit essayer de servir toute la communauté.

Le monde Unix/Linux, et la communauté Open Source n'ont pas réellement accueilli à bras ouverts C++. La plupart des développements se font encore en C. Certains mettent en avant l'aspect performances (comme pour le kernel linux) ou réinventent la roue en ré-implémentant une couche objet dans C. Qu'en pensez vous ? Considérez-vous le succès de KDE (développé en C++), et la rapidité de son développement (face à un Gnome développé en C qui stagne) comme une démonstration de la supériorité de C++ sur de tels développements ?

Je pense que réinventer la roue est stupide et que l'utilisation de C montre trop souvent l'ignorance de ce que C++ peut faire pour construire des systèmes. C'est un signe de l'immaturité de notre domaine que les gens résistent à adopter des outils plus avancés et préférer dépenser leurs énergies à réinventer des choses en utilisant des outils primitifs plutôt que de prendre le temps d'apprendre à maîtriser des outils plus puissants.

Dans l'esprit de beaucoup de programmeurs (particulièrement les programmeurs Java), C++ reste une version orienté objet de C. Que diriez vous pour les convaincre que C++ est plus que ça ?

C'est dur de convaincre des gens qui ne veulent pas être convaincus.

```
#include<string>
#include<vector>
#include<iostream>
#include<algorithm>
using namespace std;

int main()
{
    vector<string> v;
    string s;

    // read a file of words
    while (cin>>s) v.push_back(s);

    // sort the words
    sort(v.begin(),v.end());
    ostream_iterator<string> os(cout,"\n");

    // output unique words
    unique_copy(v.begin(),v.end(),os);
}
```

Tentez d'écrire cela en C et comparez. Et faites attention à ne pas introduire de débordements de tampons ou des fuites mémoire.

En réponse à une question au sujet de C#, vous avez dit que c'était un langage propriétaire sur une plateforme propriétaire. Mais C# a été standardisé par l'ECMA et s'apprête à faire de même avec l'ISO. Et vous pouvez trouver des implémentations sur d'autres plates-formes (Mono sur Linux). Si demain C# réussit la standardisation ISO, changerez-vous d'avis ?

Probablement non, et je considère cela peu probable. Je regarderais soigneusement le processus de standardisation pour vérifier qu'il y ait une réelle participation de plusieurs parties intéressées et que l'évolution du langage ne soit pas dans les mains d'une seule société. Une vraie standardisation est plus que la production d'un morceau de papier.

C# tout comme Java a pris la voie de l'héritage simple, contrairement a C++ qui implémente l'héritage multiple. Pensez vous que l'héritage multiple est toujours la meilleure solution ?

L'héritage multiple n'est pas *toujours* la meilleure solution, mais parfois la meilleure solution d'un problème implique l'héritage multiple. Notez que même Java inclut une forme limitée d'héritage multiple : l'héritage d'interfaces. Je suis très à l'aise avec l'héritage multiple et je connais de nombreux exemples qui - à mon avis - ne peuvent pas être faits élégamment sans héritage multiple. Mon livre "The C++ Programming Language (3ème édition)" contient plus d'une douzaine de tels exemples. Une technique clé est d'hériter séparément une interface (typiquement une classe d'abstraction) et une implémentation partielle.

Vous pouvez toujours réécrire un exemple utilisant l'héritage multiple en utilisant seulement l'héritage simple (avec des fonctions de forwarding). Cependant, le résultat est souvent un programme plus long, qui reflète moins fidèlement la conception et qui est plus dur à maintenir. Notez que vous pouvez aussi réécrire tout exemple utilisant l'héritage simple en un exemple n'utilisant aucun héritage en employant la même technique et avec le même impact négatif sur la clarté de code. Un langage qui ne supporte pas l'héritage multiple est simplement moins expressif qu'un langage supportant l'héritage multiple et force parfois le programmeur à compliquer encore un peu plus son code.

Ces derniers temps on parle beaucoup plus de framework que de langages, que ce soit J2EE pour Java, où l'on lie fortement un langage à un framework, ou .NET, où Microsoft met en avant le framework au détriment des langages. Que pensez vous de cette approche ? Pensez vous qu'un framework complet soit la prochaine étape nécessaire à l'évolution de C++ ?

Les gens parlent beaucoup de frameworks, mais l'histoire est jonchée de frameworks qui n'ont pas survécu à leurs ambitions. J'ai vu des frameworks qui ont été des succès, mais ils étaient généralement limités à un domaine. Je suis sceptique quant à un framework universel, et encore plus quand de tels frameworks sont des produits issus d'un éditeur de plate-forme entrant en compétition avec des frameworks similaires d'autres éditeurs. En tant qu'utilisateur, je préfère garder mon indépendance vis-à-vis des éditeurs autant que possible.

J'aimerais voir les bibliothèques fournir un accès plus propre et plus généralisé aux frameworks, plutôt que de voir les langages intimement liés à un framework unique.

Lors d'une interview que vous avez accordée au C/C++ Users Journal, vous disiez que l'un de vos objectifs avec C++ était d'augmenter le niveau d'abstraction. Est-ce toujours votre objectif pour les prochaines versions ? Peut-on réellement implémenter un fort niveau d'abstraction sans impacter les performances (Java fait pas mal de concessions par exemple) ? Quels sont les mécanismes que vous voudriez voir intégrés à C++ ? (garbage collector...) Quels mécanismes voudriez-vous voir améliorés ?

Mon objectif a toujours été de permettre d'exprimer l'intention du programmeur plus clairement et directement dans le code. Donc oui, augmenter le niveau d'abstraction a toujours été un de mes objectifs pour C++.

Oui, vous pouvez augmenter de manière significative le niveau d'abstraction sans sacrifier sur l'efficacité. La clef est un système de typage statique (à la compilation) qui soit flexible et extensible. La partie STL de la bibliothèque standard est un bon exemple. Par exemple, un vecteur est défini de telle sorte qu'il puisse prendre les éléments de n'importe quel type (les prédéfinis comme ceux définis par l'utilisateur) sans surcharge.

Pour comparer avec Java, notez qu'un vecteur C++ prend les valeurs d'un type défini par l'utilisateur, plutôt que les références aux objets de types définis par l'utilisateur. Cela économise à la fois la mémoire (vous n'avez pas besoin de stocker les références ou des informations de gestion de la mémoire pour des éléments individuels sur le tas) et les coûts d'accès (pas de coût d'indirection à travers une référence ni aucune vérification de type des éléments extraits d'un vecteur). L'absence d'un besoin de vérification de type à l'exécution (casting) amène aussi de manière significative un code plus propre et plus concis.

Autre exemple, les algorithmes standard (comme *sort*), qui peuvent être appliqués à n'importe quel conteneur approprié et paramétré avec des critères de comparaison, sont de manière significative encore plus rapides que leurs équivalents en C (comme *qsort*).

J'aimerais que le comité de standardisation C++ reconnaisse explicitement que la ramasse-miettes est une technique d'implémentation valable pour C++, mais je ne veux pas faire que la sémantique de C++ dépende d'un ramasse-miettes. Il y a des domaines d'application (comme les drivers et autre code du noyau) pour lesquels le ramasse-miettes n'est pas approprié. Et si vous voulez du ramassage de miettes dans C++, vous pouvez utiliser un des ramasse-miettes existants. Cela marche très bien pour certains types d'applications pour lesquelles le ramasse-miettes est une technique prudente.

Vous décrivez C++ comme un langage multi paradigmes, principalement l'orienté objet et la généricité. Quels sont les paradigmes qui pourraient peut être trouver leur place dans une prochaine version de C++ ?

Je pense que le terme de paradigme est largement galvaudé, et je préfère le terme " style de programmation ". Aussi, il ne faut pas oublier l'utilisation de simples classes autonomes (c'est-à-dire qui ne font pas partie d'une hiérarchie). Elles sont essentielles pour la flexibilité et l'efficacité de nombreuses techniques de C++ modernes et sont des exemples d'abstraction de données.

Je ne pense pas que C++ supportera un nouveau paradigme dans un proche avenir, mais peut-être suis-je trop conservateur dans ce que j'appelle un paradigme. J'espère que le prochain standard C++ supportera la programmation distribuée, mais qu'il le fera principalement à travers la bibliothèque standard.

Avez-vous l'intention d'intégrer la Programmation Orientée Aspect ?

Non. Je ne suis pas encore certain que la "Programmation Orientée Aspect" puisse s'accorder avec mes principes qui consistent à représenter indépendamment les concepts et le code et à les combiner librement (et seulement) quand nécessaire. Je ne suis certain non plus que la notion de programmation orientée aspect soit générale et généralement utile. Rappelez-vous qu'on n'ajoute de nouvelles particularités à un langage de programmation largement utilisé qu'après avoir considérer les alternatives.

Merci, Bjarne, de votre contribution.

Notez que les questions fréquemment posées sur C++ et leurs réponses peuvent être consultées sur ma homepage. Vous y trouverez également de nombreux articles et des liens utiles sur C++.

http://www.research.att.com/~bs ●

Méthodes agiles (7)

Unified Process

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

Voici le dernier volet de notre dossier consacré aux méthodes agiles. Nous allons voir comment Unified Process (UP) se positionne par rapport aux autres méthodes agiles.

Rappelons pour commencer, de manière synthétique, les principales caractéristiques d'Unified Process.

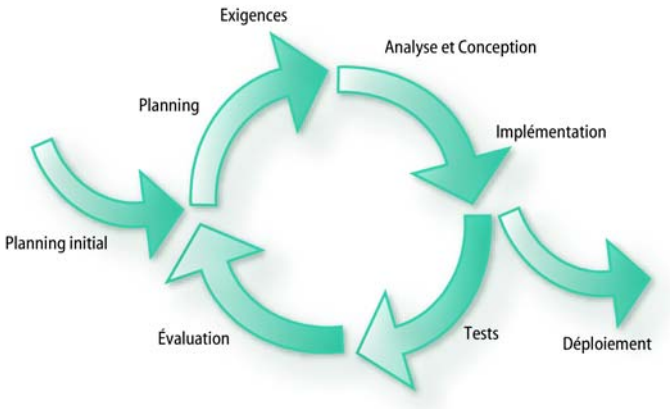
UP est pilotée par les cas d'utilisation

Les cas d'utilisation (Use Cases) sont les véritables pilotes du projet, quelle que soit l'activité et quelle que soit la phase. En effet, le système est tout d'abord analysé, conçu et développé pour des utilisateurs. Tout doit donc être fait en adoptant le point de vue utilisateur. Les cas d'utilisation sont l'outil de modélisation des besoins en termes de fonctionnalités : c'est sous cette forme que sont exprimées les exigences. Les cas d'utilisation servent aussi de base pour l'analyse, le modèle logique, ainsi que de base pour les tests fonctionnels.

UP est centrée sur l'architecture

UP intègre très tôt l'architecture dans le cycle de vie d'une application. L'architecture du système est décrite à l'aide de différentes vues. Alors que les modèles cas d'utilisation, analyse, conception, implémentation, déploiement, sont complets et exhaustifs, les vues définies par l'architecte ne reprennent que les éléments significatifs. L'architecte procède de manière incrémentale : il commence par définir une architecture simplifiée qui répond aux besoins classés comme prioritaires avant de définir à partir de là les sous-systèmes de manière beaucoup plus précise.

Figure 1 - UP est une méthode itérative et incrémentale (source : Rational).



UP est itérative et incrémentale

UP procède par itérations par opposition aux méthodes en cascade qui sont plus coûteuses et ne permettent pas une bonne gestion du risque. En procédant de manière itérative, il est possible de découvrir les erreurs et les incompréhensions plus tôt. Le feedback de l'utilisateur est aussi encouragé et les tests effectués à chaque utilisation permettent d'avoir une vision plus objective de l'avancement du projet. Enfin, le travail itératif permet à l'équipe de capitaliser à chaque cycle les enseignements du cycle précédent (figure 1).

UP gère les besoins et les exigences

Les exigences du client changent au cours du projet. UP recommande de découvrir les besoins, de les organiser et de les documenter de manière à avoir une approche disciplinée de l'étude des exigences et de pouvoir les filtrer, les trier par propriété et réaliser leur suivi pour pouvoir estimer de manière objective les fonctionnalités et les performances du système à développer.

UP est fondée sur la production de composants

UP recommande de structurer l'architecture à base de composants (COM, CORBA, EJB, etc.). Les architectures ainsi construites sont de fait plus robustes et modulaires. De plus, cette approche permet la réutilisation de composants existants si ceux-ci ont été développés de manière assez générique et dans une approche de Framework.

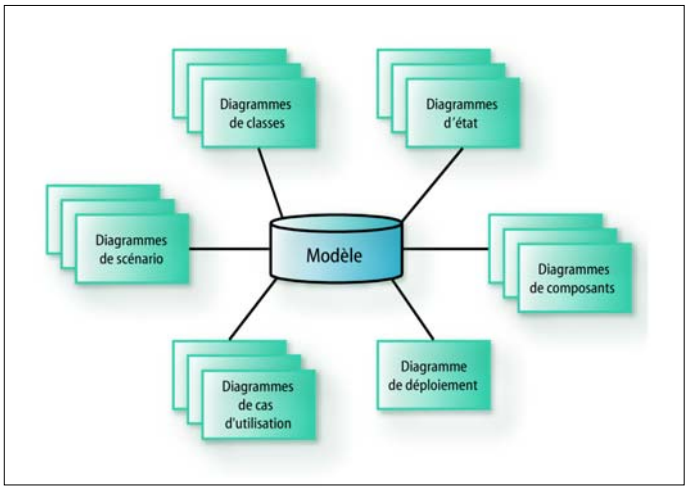


Figure 2. Modélisation visuelle selon différentes vues (source : Rational)

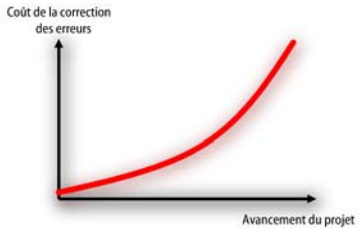


Figure 3.

UP pratique la modélisation visuelle

UP préconise la modélisation visuelle du système. Un modèle est une simplification de la réalité qui décrit le système selon un certain point de vue. La modélisation sous différents points de vue permet une meilleure compréhension globale du système à concevoir. L'utilisation d'outils visuels augmente encore les capacités de l'équipe à gérer la complexité des systèmes (figure 2).

UP se soucie en permanence de la qualité

Partant du constat que le coût de la correction des erreurs augmente exponentiellement au fur et à mesure de l'avancement du projet, il est absolument nécessaire d'avoir un contrôle permanent et rigoureux des fonctionnalités, de la fiabilité, des performances de l'application et des performances du système. Pour cela, une automatisation des tests tout au long du cycle de vie s'impose. Ainsi, les défauts sont détectés très rapidement et le coût de leur correction est minimisé, l'estimation de l'état d'avancement du projet est plus objective et la qualité et les performances des parties "à risque" est améliorée.

UP gère les risques de façon systématique et permanente

La phase de pré-étude a pour but d'identifier les risques qui pourraient mettre le projet en péril. Tout au long du projet, il est recommandé de maintenir une liste de risques, issus de la phase de pré-étude, classés selon leur danger pour l'équipe afin d'avoir une vue plus explicite des choses.

Les activités dans la méthode UP

UP définit les activités essentielles et propose, pour chacune d'entre elles, une liste d'intervenants (architecte, analyste...) et des travaux associés.

Expression des besoins

UP distingue deux types de modèles : "Domain Model" et "Business Model". Le modèle de domaine ne dépend pas de l'application : il a pour but d'apprendre un domaine, un métier jusque là méconnu. Le modèle "business" est plutôt une modélisation des procédures en vigueur dans l'entreprise et s'intéresse directement aux collaborateur et à leurs divers travaux. C'est à partir de ce modèle que peut alors démarrer l'analyse des cas d'utilisation du système. Il s'agit de mieux connaître le domaine et ses activités avant de s'intéresser aux fonctionnalités du système par une étude par les cas d'utilisation.

Analyse

Comprendre et structurer le logiciel à développer sont les objectifs de l'analyse. On construit dans cette activité une représentation interne quasi-idéale du système, sans trop tenir comte des exigences et contraintes de conception. L'analyse utilise le langage du développeur alors que l'expression des besoins se fait du point de vue utilisateur.

Conception

La conception consiste à définir l'architecture du système. La conception n'est non pas une phase du process UP mais une activité qui trouve sa place dans toutes les phases. La conception est ainsi réalisée de manière incrémentale. Dans une phase de pré-étude elle consiste par exemple à maquetter l'architecture, tandis qu'en phase de construction la conception de grossière devient beaucoup plus détaillée. Si l'analyse est une vue purement logique, la conception est plus physique et se soucie des contraintes que l'analyse avait laissées de côté.

Implémentation et Test

Cette activité consiste à créer à proprement parler les divers composants : sources, scripts, puis exécutables... Les tests se basent sur les cas d'utilisation.

Les phases du cycle de vie

Les différentes activités sont mises en œuvre lors des différentes phases du cycle de vie ; l'importance des différentes activités dans les phases du cycle est illustrée dans la figure 4.

Etude d'opportunité

C'est durant cette phase qu'il faut se poser la question de la faisabilité du projet, des frontières du système, des risques majeurs qui pourraient mettre en péril le projet. A la fin de cette phase, est établi un document donnant une vision globale des principales exigences, des fonctionnalités clés et des contraintes majeures. Environ 10 % des cas d'utilisation sont connus à l'issue de cette phase. Il convient aussi d'établir une estimation initiale des risques, un "Project Plan", un "Business Model".

Elaboration

Cette phase a pour but de préciser l'architecture. A ce stade, 80 % des cas d'utilisation sont obtenus. Cette phase permet aussi de connaître des exigences supplémentaires, en particulier des exigences non fonctionnelles, de décrire l'architecture du système, de le prototyper, de réviser la liste des risques et de mettre en place un plan de développement. En théorie, c'est à ce stade qu'on est à même de faire une offre de réalisation du système.

Construction

L'objectif est de fournir une version beta de la release en cours de développement ainsi qu'une version du manuel utilisateur.

Transition

Cette phase a pour objectif de peaufiner le système et de corriger les derniers bugs pour pouvoir passer de la version beta au déploiement. Comme le nom de la phase l'indique, il s'agit aussi de préparer la release suivante et de boucler le cycle soit sur une nouvelle étude d'opportunité soit une élaboration ou construction.

UP est elle une méthode agile ?

Pour la plupart des défenseurs des méthodes agiles, UP n'est pas à proprement parler " agile ". Le cycle de vie UP est souvent considéré comme trop lourd, et le nombre de livrables (souvent appelés Artefacts) associés est considéré comme trop important. Les détracteurs ajoutent que la mise en œuvre d'UP est d'ailleurs impossible sans l'utilisation de la suite d'outils Rational.

Pourtant, par certains côtés, UP n'est pas totalement dénué d'agilité : le caractère très itératif du cycle, l'importance donnée à la gestion des risques, la proximité avec les besoins utilisateurs (par le biais des cas d'utilisation) en sont l'exemple. Il n'en reste pas moins que l'image d'UP ne s'inscrit pas aujourd'hui dans la mouvance

des méthodes de type Extreme Programming. La sortie d'outils comme Rational XDE constitue de ce fait une avancée très importante de la part de l'éditeur, au même titre que ses outils de tests automatisés : en positionnant la modélisation au cœur du code par le biais d'un environnement totalement intégré à celui d'IBM ou de Microsoft, assurant une bi-directionnalité entre les modèles et le code source, Rational brise l'image de Rose, AGL souvent considéré comme trop éloigné du développement.

Car le compromis entre les méthodes agiles, très centrées sur le code, et l'exigence de la part des grands comptes de livrables complémentaires (documentation notamment) passe nécessairement par la mise en œuvre d'outils de ce type.

Prendre en compte l'agilité

Cette mise en perspective d'Unified Process conclut notre série d'articles sur les méthodes agiles. Les signes de Rational en faveur de l'agilité sont assez symptomatiques de ce qui est en train de se produire : les méthodes traditionnelles ne sont pas abandonnées, mais peu à peu transformées pour prendre en compte le signal fort envoyé par le taux d'échec important des projets informatiques et par des méthodes telles que Extreme Programming.

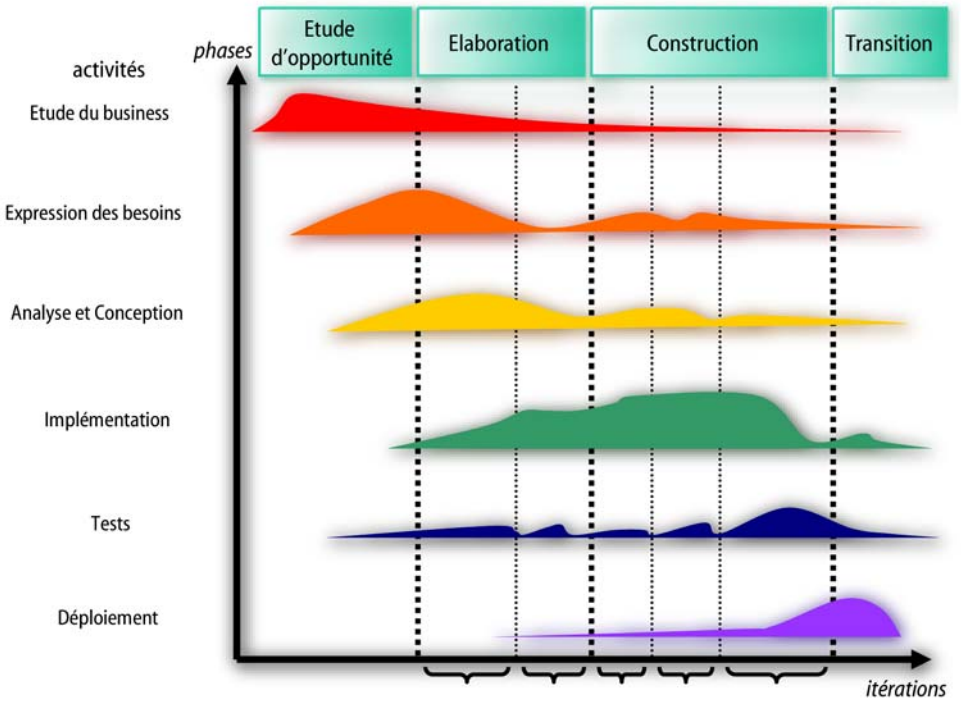


Figure 4. Importance des activités au cours des différentes phases (source : Rational)

Il est probable que, pour la plupart des grandes entreprises françaises, il n'y aura pas de "grande rupture" sur les méthodes mises en œuvre, même si un certain nombre de comptes lancent des projets pilotes autour d'Extreme Programming. Beaucoup adoptent — ou ont déjà adopté — des pratiques agiles sans remettre en cause l'existant global. Pour les défenseurs des méthodes agiles, cette approche reste trop réductrice et n'offrira pas des résultants probants. Il n'en reste pas moins que la mise en œuvre progressive de l'agilité semble bien être un passage obligé.

Sami Jaber,
consultant et formateur chez Valtech sur J2EE et .NET depuis trois ans, spécialisé dans l'architecture des systèmes fortement distribués (EJB, MTS/DCom, Corba, RMI...) orientés multi-niveaux et internet/intranet. Anime des séminaires autour d'XML, EJB, .NET. Webmaster du site DotNetGuru.org

[suite de la page 1]

Avec l'expérience, nous savons aujourd'hui qu'il est possible de masquer à travers des objets toute la structure d'une base relationnelle. Cette approche a pour nom le mapping Objet/Relationnel. Pour ce faire, il existe trois types d'approche :

• Réaliser cette opération manuellement de manière spécifique en utilisant des modèles de conception éprouvés (Design Patterns)

• Utiliser des outils du marché tels que les EJB (Enterprise JavaBean) ou les outils de mapping objet/relationnel (JDO, ObjectSpaces, TopLink, CocoBase…)

• Faire un mélange des deux premières approches en optimisant manuellement les parties sensibles (gros volumes...)

Nous ciblerons dans cet article la deuxième et la troisième approche à travers deux frameworks promis à un bel avenir : Java Data Objects (JDO) et Microsoft ObjectSpaces. Leurs modèles de développement sont très proches, mais aussi et surtout parce qu'ils représentent chacun à leur manière les deux architectures prédominantes du moment : Java et .NET. Dans un premier temps, nous présenterons ObjectSpaces, le framework de persistance encore en préparation chez Microsoft. Puis nous comparerons les deux approches, JDO et ObjectSpaces, de manière technique.

La problématique de la persistance sous .NET

Microsoft reste très discret sur le sujet, mais la plate-forme .NET actuellement ne gère pas la persistance des objets. Tout d'abord, .NET n'intègre pas la notion de synchronisation en mode objet/relationnel. Le framework .NET ne fournit pas de mécanisme en standard permettant de gérer automatiquement la synchronisation d'un cache d'objets sur le principe des EJB Entités. Plus précisément, ObjectSpaces est le seul framework de mapping Objet/Relationnel de Microsoft à proposer ce type de fonctionnalité. La différence essentielle entre ObjectSpaces et les EJB se situe au niveau de l'intervention de l'utilisateur. Celui-ci doit spécifier explicitement dans ObjectSpaces qu'il désire synchroniser une instance, alors que les EJB s'en chargent automatiquement en cas d'appel à un modificateur d'accès, et ce, dans le cadre d'une transaction. C'est donc une démarche active et non passive. Voilà pourquoi ObjectSpaces s'inscrit plus dans la logique des JDO que dans celle des EJB.

Ensuite, il n' y a pas d'entités persistantes gérées par conteneur. Dans ObjectSpaces, il n'existe pas à proprement parler de conteneur dans le sens "service" ou "serveur". Il y a bien un cache, mais qui n'est pas distribué comme c'est le cas dans les EJB.

Pour autant, la persistance ne constitue pas une priorité pour Microsoft. L'éditeur a toujours argumenté sur le fait que les EJB persistants ne sont pas efficaces en terme de charge. D'ailleurs, le Java PetStore qu'ils ont réécrit fait la part belle aux procédures stockées.

L'avenir sera riche en enseignements, car ObjectSpaces sera disponible d'ici quelques mois. La position de Microsoft au sujet de la persistance sera surtout radicalement différente. Si l'on jette un coup d'œil aux présentations d'ObjectSpaces, les concepteurs de ce framework critiquent vivement l'utilisation des DataSet, des procédures stockées et prônent la mise en place d'une couche d'objets persistants.

L'architecture ObjectSpaces

L'architecture Objectspaces a pour objectif principal de permettre d'exposer des données sous la forme d'objets et de liste d'objets indépendamment de leur représentation physique (tables, colonnes, lignes ou éléments XML). Ces objets sont appelés "objets persistants" et sont définis en

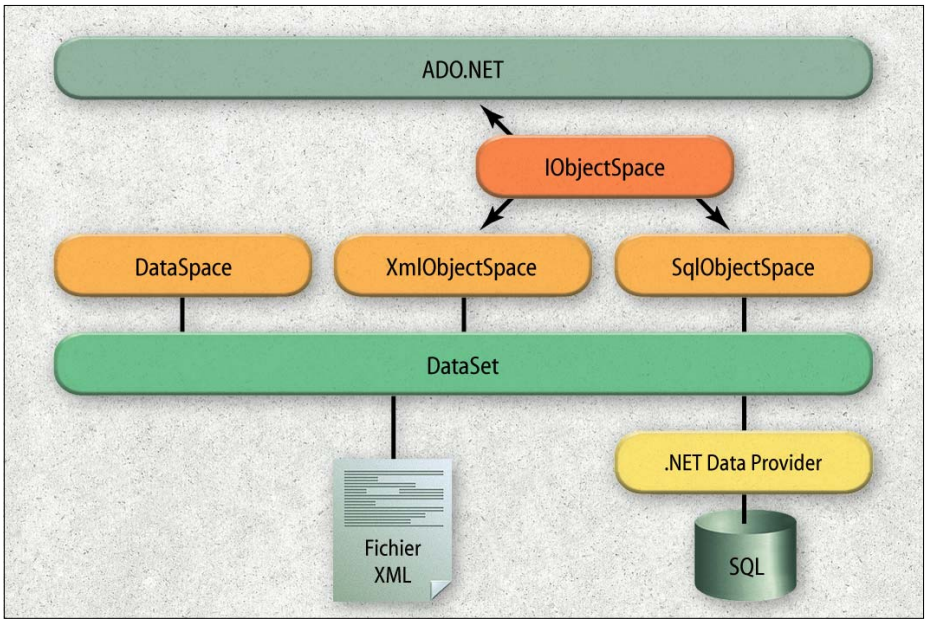


Figure 1. Les classes principales du framework ObjectSpaces.

utilisant les conventions du framework .NET. L'objet le plus important dans le framework est l'usine de classe ObjectSpace (pattern Factory) dont le rôle est le chargement, la création et la suppression d'instances d'objets persistants dans la base de données. Le terme base de données doit être pris au sens large, il comprend l'ensemble des bases relationnelles mais aussi les fichiers XML à plat.

ObjectSpaces se sert des classes ADO.NET pour implémenter son mécanisme de persistance. Un cache local est utilisé autour de l'objet DataSet permettant de charger des données à partir d'une source quelconque (XML, SGBD...) afin de répercuter en base toute modification du DataSet. Il est important de comprendre que le framework ObjectSpaces fonctionne dans un mode totalement déconnecté lui permettant de se re-synchroniser avec la source de données en cas de modification. Plusieurs alternatives sont offertes pour la gestion des objets :

- Implicitement laissez ObjetSpaces gérer le chargement et la persistance de ses données
- Gérer soi-même explicitement les interactions avec la base

La gestion implicite se fait par l'intermédiaire des objets XMLObjectSpace et SQLObjectSpace, la gestion explicite se fait par l'intermédiaire de la classe DataSpace en fournissant un objet DataSet.

La figure 1 illustre les classes principales du framework. IObjectSpace est utilisé lorsqu'il est nécessaire d'être indépendant du type de données manipulé (XML ou SQL). Le mapping entre modèle objet et modèle physique se fait via un fichier XML. Le but affiché par les équipes d'ObjectSpaces consiste à permettre toute modification du fichier XML de mapping sans avoir à recompiler le code client.

Voyons de plus près les différentes étapes nécessaires pour créer un objet persistant.

Implémenter l'objet

Un objet persistant est défini de la manière suivante : tous ses accesseurs sont abstraits. Le Framework se charge de générer le code nécessaire à leur implémentation en fonction du type de données manipulé. De plus, tout objet ObjectSpaces est caractérisé par un identificateur unique représenté par l'attribut de Runtime UniqueId. Cet attribut est en lecture/écriture lorsque le développeur choisi de l'initialiser ou en lecture seule lorsqu'il laisse le framework s'en charger. Enfin, certaines méthodes sont liées à l'API : OnCreate(..) représente le constructeur de la classe (rappelez-vous, c'est le framework qui a la responsabilité de gérer la création et la destruction de votre objet).

```
[C#]
public abstract class Consultant
{
    // This property contains a unique value.
    [UniqueId] protected abstract string Id { get; set; }

    public abstract string Name { get; set; }
    public abstract string CompanyName {get;set;}
    public abstract string Phone { get; set; }
    public abstract string Fax { get; set; }

    // OnCreate is the ObjectSpaces "constructor"
    public void OnCreate(string newId)
    {
        Id = newId;
    }
}
```

Il existe d'autres méthodes de ce type décrites dans la matrice suivante :

Méthode	Description
OnCreate()	Appelée à la création de l'objet
OnMaterialize()	Appelée la 1ère fois après l'appel de OnCreate
OnDelete()	Appelée à la destruction de l'objet

Définir les relations

L'étape suivante consiste à créer les relations entre les classes. Rajoutons dans la classe précédente Consultant une relation une relation bidirectionnelle 1-1 avec la classe Course : un consultant donne un cours et un cours ne peut être donné que par ce consultant (il n'a pas intérêt à tomber malade celui-là). La classe modifiée nous donne :

```
[C#]
public abstract class Consultant
{
    // This property contains a unique value.
    [UniqueId] protected abstract string Id { get; set; }

    public abstract string Name { get; set; }
    public abstract string CompanyName {get;set;}
    public abstract string Phone { get; set; }
    public abstract string Fax { get; set; }

    public abstract Course MyCourse { get; set; }

    public void OnCreate(string newId) {...}
}
```

Voici la classe Course associée. Remarquez la relation bidirectionnelle :

```
public abstract class Course
{
    [UniqueId] public abstract int CourseId { get; set; }
    public abstract Consultant MyConsultant { get; set; }

    public void OnCreate(int Id)
    {
        CourseId = Id;
    }
}
```

Le fichier XML de mapping sera le suivant :

```
<map
 xmlns="http://www.microsoft.com/ObjectSpaces-v1">
    <type name="Consultant">
        <property name="Id" />
        <property name="myCourse" relationship=
          "TeachThisCourse" side="Parent" />
    </type>
    <type name="Course">
        <property name="CourseId" />
        <property name="myConsultant" relationship=
          "ImTaughtByThisConsultant" side="Child" />
    </type>
    <relationship name"TeachThisCourse"
     type="OneToOne" parenType="Consultant"
     childType="Course">
        <key parent="Id" child="CourseId"/>
    </relationship>
</map>
```

Il existe à l'heure actuelle deux types de relations différentes prises en charge par le framework : 1,1 (OneToOne) et 1,n (OneToMany), avec un concept de lien d'association LinkedBy ou de composition ComposedBy. Ainsi, lors d'une relation de composition, toute suppression d'un objet parent entraînera la suppression des objets fils.

Attribut XML	Oblig	Description
type	oui	OneToOne ou OneToMany
parentType	oui	Se réfère au parent de la relation
childType	oui	Se réfère au fils de la relation

Utilisation de l'objet et synchronisation du cache

Pour utiliser un objet ObjectSpace, commencez par récupérer une référence sur la Factory avec la méthode CreateObjectSpace. Ensuite, la méthode CreateObject() renvoie des références d'objets persistants en assurant leur création en base. La mise à jour et la synchronisation se fait manuellement par l'appel aux méthodes os.Update() ou

```
os.ReSync() pour assurer la synchronisation entière du cache.
// Fichier de configuration de la base de données
// et du mapping
IObjectSpace os =
    ObjectSpaceFactory.CreateObjectSpace("source.xml");
Consultant theConsultant =
    (Consultant)os.CreateObject(typeof(Consultant), 1);
theConsultant.CompanyName = "Valtech" ;
theConsultant.Phone = "123456" ;

// Or os.UpdateAll() for updating all objects
os.Update(theConsultant) ;

// Synchronise all the cache with the underlying storage
os.ReSyncAll() ;

Course theCourse =
    (Course)os.CreateObject(typeof(Course), 1);

// Establish the one-to-one relationship.
theConsultant.MyCourse = theCourse;

Console.WriteLine("theConsultant.MyCourse.Id: {0}",
theConsultant.MyCourse.Id);
```

Liste d'objets à partir de critères

OPath est un langage de requête créé spécialement pour ObjectSpaces. Il permet de récupérer un graphe d'objets persistants selon des critères spécifiés en OPath. Le code suivant nous illustre quelques exemples de requêtes :

```
// Renvoie la liste des consultants chez Valtech
os.GetObjects(GetType(Consultant), "CompanyName =
'Valtech'") ;

// Renvoie un consultant
os.GetObjects(GetType(Consultant), "") ;

// etc ...
os.GetObjects(GetType(Customer), "Orders.OrderDate >=
'5/1/1998' and Orders.OrderDate <= '5/31/1998'");
os.GetObjects(GetType(Customer), "Orders[Freight >
10].Details.Quantity > 50");

// Utilisation dans le cadre d'une relation 1,n :
// Consultant donne n Courses
// Recherche et Parcours tous les consultants dont le nom
// commence par F travaillant chez Valtech
foreach( Consultant theConsultant in
    os.GetObjects(typeof(Consultant),
    "CompanyName  ='Valtech' and Name < 'F*'") )
{
    Console.WriteLine("\nConsultant Name: "+
        theConsultant.Name  );
    foreach(Course theCourse in theConsultant.Courses){
        Console.WriteLine(
            "Consultant: Id: " + theConsultant.Id +
            ", Company Name :" + theConsultant.Company +
            ", Phone : " + theConsultant.Phone +
            ", Fax : " + theConsultant.Fax +
            ", Email: " +  theConsultant.Email);
    }
}
```

Pourquoi un n-ième langage après OQL (Object Query Langage), SQL (Simple Query Langage), XPath... ? C'est une question que nous sommes en droit de nous poser. Mais il faut bien l'avouer, dans ce domaine, Microsoft n'est pas le seul éditeur à s'être confronté au problème. Sans aller très loin, le monde Java contient une multitude d'exemples où la standardisation du langage de requête objet est un vain mot. Ce sujet a toujours été source de moult discussions interminables, que ce soit dans la spécification EJB (EJBQL) ou dans des API plus spécifiques telles que JDO (http://www.castor.org). Certains utilisent EJBQL (WebLogic), d'autres préfèrent OQL et finalement on en revient toujours à SQL...

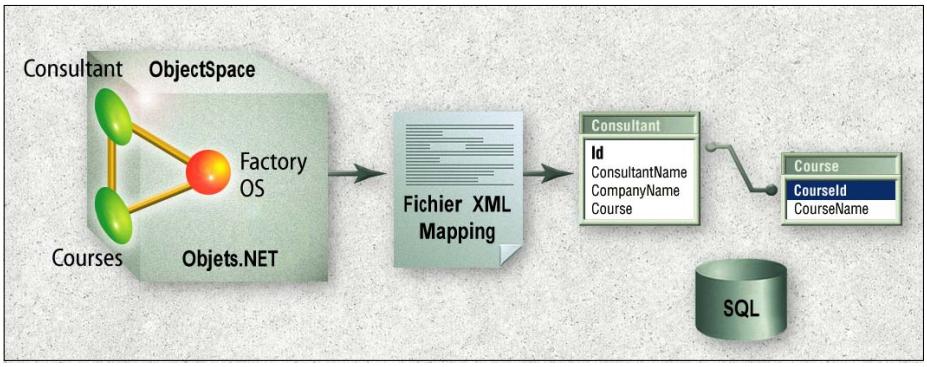


Figure 2. Mapping objet/relationnel : principe du stockage en base d'objets.

Mapping Objet/Relationnel

Nous avons gardé le meilleur pour la fin : le mapping Objet/Relationnel. Comment configurer notre base existante (SQLServer, Oracle, ...) pour continuer à utiliser les objets précédents ?

Rappelez-vous ce qui a été dit en début d'article, le but d'ObjectSpaces est de masquer au client le modèle de stockage physique. Le fait de passer d'une gestion sous forme de fichiers XML à une gestion sous la forme d'une base doit-il changer la vision que le client a de cet objet ? Bien sûr que non ! C'est pourquoi, au niveau des API, vous ne verrez rien qui fasse intervenir une éventuelle requête SQL, ni même une table. Tout se passe à l'intérieur des fichiers de configurations : Map.xml, Connect.xml et DbSource.xml. Vous spécifiez les sources de données vers vos bases, les tables et colonnes à mapper sur les objets et le tour est joué !

La figure 2 illustre le principe du stockage dans une base SQL Server de nos deux objets Consultant et Course avec deux tables.Pour accomplir cette tâche il suffit de créer le fichier de mapping XML suivant :

```
<map
 xmlns="http://www.microsoft.com/ObjectSpaces-v1">
   <type name="Consultant" dataSource="Consultant"
    source="ConsultantDS">
      <property name="Id" dataSource="Id" />
      <property name="Company"
       dataSource="CompanyName" />
      <property name="Consultant"
       dataSource="ConsultantName" />
      <property name="Courses" relationship="Teach"
       side="Parent" />
   </type>
   <type name="Course" dataSource="Course"
    source="ConsultantDS">
      <uniquekey name="Id" mappedBy=
       "autoIncrement" dataSource="CourseId"/>
      <property name="CourseId"
       dataSource="Name"/>
      <property name="Consultant" relationship=
       "Teach" side="Child" />
   </type>
   <relationship name"Teach" type="OneToMany"
    parenType="Consultant" childType="Course">
      <key parent="Id" child="Courses"/>
   </relationship>
</map>
```

Ce fichier contient l'association table(colonne) => champs de l'objet ainsi que les relations entre ces objets. A l'avenir, nous pouvons imaginer que Microsoft fournira des IHM permettant de sélectionner des graphes d'objets et de réaliser le mapping graphiquement. Biztalk Mapper ne propose-t-il pas déjà quelque chose de semblable ?

Ensuite, il ne faut pas oublier d'associer le fichier XML (config.xml) contenant les DataSource ADO.NET (lien vers les bases).

```
<sources
 xmlns="http://www.microsoft.com/ObjectSpaces-v1">
    <source name="Example" adapter="sql"
     connection="Data Source=LocalHost; Integrated
     Security=SSPI; Database=Consultant"/>
</sources>
```

Enfin, le code source du client une fois modifié doit juste se contenter de charger les fichiers de configuration de la manière suivante :

```
IObjectSpace os =
ObjectSpaceFactory.CreateObjectSpace("config.xml");
```

Vous l'aurez compris, ObjectSpaces est un énorme bond en avant vers l'objet de Microsoft. La couche de persistance est la seule brique qui manquait à .NET pour devenir une véritable architecture 3-tiers. Les lecteurs ayant des connaissances J2EE et EJB ont dû sourire discrètement à la lecture de cet article. Et oui, ces concepts existent déjà dans les EJB à travers les composants Entity. Oui, ces notions existent aussi dans la multitude d'outils de persistance Java tels que JDO, TopLink, CocoBase, Oracle9i et bien d'autres ! Microsoft ré-écrit un n-ième Framework de persistance et a un retard important à combler dans ce domaine, il faut bien l'avouer. D'un autre coté, est-ce vraiment le but d'ObjectSpaces de vouloir contrer Sun ? A suivre les discussions émanant du forum microsoft.public.dotnet.objectspaces, cela reste moins sûr...

Au delà des discussions d'ordre stratégique, il se pose la question fondamentale : qu'est-ce que JDO ou les EJB Entity font-ils de plus qu'ObjectSpaces et inversement ?

Java Data Objects vs ObjectSpaces

Toute comparaison n'a de pertinence que lorsqu'elle basée sur un ensemble de critères communs. Nous avons choisi pour ce comparatif les critères qui avaient le plus d'intérêt pour les applications d'entreprise et qui impactent directement la qualité d'un produit :

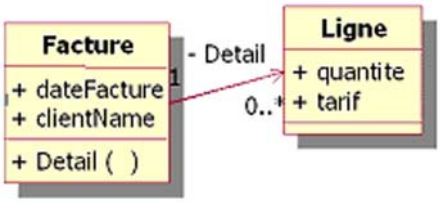
- Le modèle de développement
- Le mapping objet/relationnel
- La gestion des transactions
- Le langage d'interrogation
- Le cache d'objets

Pour chaque critère nous analyserons de part et d'autre l'implémentation tout en focalisant notre attention sur les différences respectives. Toutefois, il est important de prendre en compte le fait qu'ObjectSpaces est un produit en cours de développement. Il sera mené probablement à évoluer, il convient donc de prendre avec précaution les conclusions qui suivront. Enfin, JDO est une spécification émise par Sun et destinée à être implémentée. C'est pourquoi, dans cet article, nous avons réalisé nos tests avec une de ces implémentations : Sun JDO-Ri qui est loin de représenter la qualité d'outils tels que Castor (ExoLab), Kodo ou LiDO (Libelis).

Le modèle de développement

Le modèle de développement consiste à proposer un socle technique pour l'implémentation des objets persistants. Cela comprend la manière d'écrire un objet persistant mais aussi les éventuelles contraintes liées aux interfaces du framework.

Sur cet aspect, JDO et ObjectSpaces adoptent la même démarche générale. Prenons un exemple concret avec le célèbre exemple de la Facture constituée de ses lignes :



JDO :

```
public class Facture {
    private String _ClientName = null;
    private List _detail = new LinkedList ();
    private java.util.Date _date = null;

    public Facture (String clientName, java.util.Date date)
    {
        _ClientName = clientName;
        _date = date;
    }
    public void addLigneFacture (Ligne 1)
    {
        _detail.add(1);
    }
    public void getClientName ()
    {
        retrun _ClientName;
    }
    public LinkedList detail()
    {
        return _detail;
    }
…
}
```

ObjectSpaces (C#) :

```
public abstract class Facture
{
    public abstract int Id {get;}
    public abstract DateTime DateFacture {get; set;}
    public abstract float Total {get; set;}
    public abstract string ClienName {get; set;}
    [Composite(typeof(Ligne), "facture")]
    public abstract IList Detail {get;}
}
```

Première remarque : ObjectSpaces impose une abstraction de la classe métier (mot clé abstract) alors que JDO, lui, autorise le client à fournir une implémentation des méthodes. En d'autres termes, le framework ObjectSpaces génère l'implémentation et JDO réalise une introspection du code client afin d'extraire les informations dont il a besoin. Nous pourrions à priori penser que JDO ne génère aucun code, mais détrompez-vous. Il faut bien comprendre que le rôle d'un framework de persistance est avant tout de stocker des objets suivant un type prédéfinit et contraint. Or, le type Facture précédent est inutilisable tel quel car il n'est pas dissociable techniquement des autres objets. C'est pourquoi JDO, mais aussi ObjectSpaces, auront pour tâche de générer à l'exécution toute l'implémentation de ces objets afin de les insérer dans le socle technique du framework. Cette opération est appelée enhancement ou extension. L'intérêt de cette technique consiste à masquer totalement au client le fait qu'il manipule une base de données ou un fichier XML et à générer de manière transparente toute la tuyauterie nécessaire à l'invocation des requêtes liées au type de stockage (SQL, XML...).

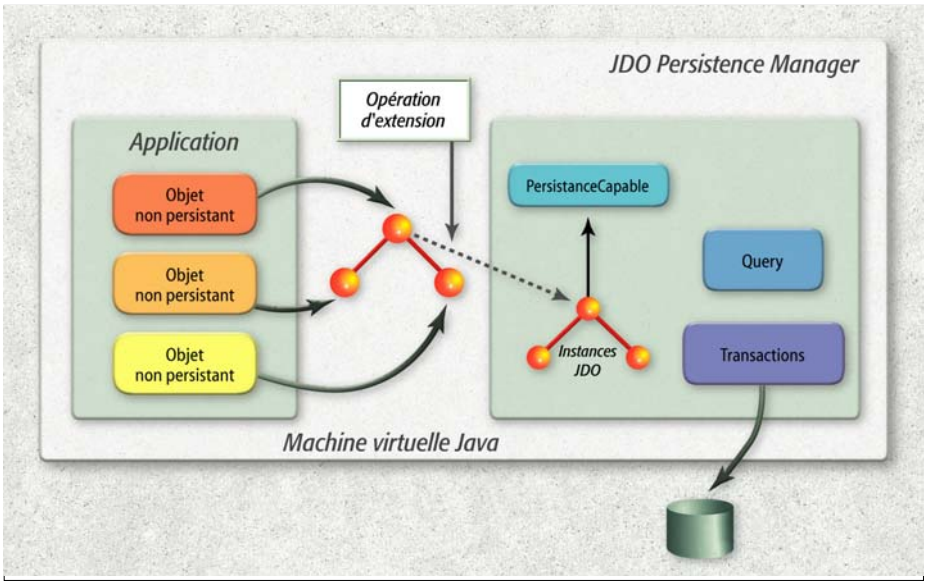


Figure 3. L'API JDO.

Voyons de plus près le type d'objet manipulé par JDO :

```
public class Facture implements PersistenceCapable {
    private String _ClientName = null;
    private List _detail = new LinkedList ();
    private java.util.Date _date = null;

    // Champs générés par l'extenseur
    protected transient javax.jdo.StateManager
        jdoStateManager;
    protected transient byte jdoFlags;
    private static String[] jdoFieldNames;
    private static Class[] jdoFieldTypes;
    private static Class jdoPersistenceCapableSuperClass ;
(…)

    public Facture (String clientName, java.util.Date date)
    {
        _ClientName = clientName;
        _date = date;
    }
    public void addLigneFacture (Ligne 1)
    {
        _detail.add(1);
    }
    public LinkedList detail()
    {
        return _detail;
    }

    // Méthodes générées par l'extenseur
    public boolean jdoIsPersistent() { // implementation
        générée}
    public boolean jdoIsTransactional() { //
implementation
        générée}
    public boolean jdoIsNewt() { // implementation
        générée}
    public boolean jdoIsDirty() { // implementation
        générée}
    public boolean jdoIsDeleted() {
        return jdoStateManager.isDeleted(); }
    public void jdoMakeDirty(){
        jdoStateManager.MakeDirty(); } (…)
}
```

Vous remarquerez que l'objet généré n'a plus rien à voir avec l'objet initial. Vous comprendrez aisément qu'il est peu souhaitable d'imposer à un développeur d'implémententer une telle classe pour chaque objet métier. L'intérêt de l'approche par génération de code est de faire prendre en charge cette lourde tâche par le framework .

Mais attardons-nous sur les API JDO une fois les objets étendus. La figure 3 illustre la philosophie générale de l'API. Tous les objets persistants sont du type PersistenceCapable. Chaque entité est gérée à l'aide d'un StateManager chargé de mettre à jour l'objet en cas de modification mais aussi de synchroniser son état avec la base sous-jacente. Le StateManager est de toute évidence l'un des objets les plus importants du framework. Il possède toute l'intelligence liée au chargement et à la sauvegarde des données en mémoire. L'ensemble de ces classes est orchestré par le PersistenceManager dont le rôle est d'intéropérer avec le type de stockage (SQL, XML, …).

Après extension, toutes les méthodes nécessaires au cycle de vie de l'objet sont créées. Lors d'une opération quelconque de l'utilisateur visant à manipuler l'état d'une entité, le conteneur JDO ou ObjectSpaces prendra soin d'invoquer au préalable les différentes méthodes de notification présentes dans l'objet (onCreate()…).

Pour changer le type de stockage, il suffit d'appliquer une extension destinée à un autre

Persistence Manager (XML), et le tour est joué !. Pour résumer, le JDO Persistence Manager joue le rôle de Factory d'objets du type PersistenceCapable que le client manipule directement de manière transparente.

```
PersistenceManager pm = pmf.getPersistenceManager();
Facture p = new Facture("Dupont", new java.util.Date());
pm.makePersistent(p);
```

Concernant ObjectSpaces, la démarche est légèrement différente même si la philosophie générale reste très proche. Contrairement à JDO qui mentionne dans la spécification officielle ce que doit contenir l'objet généré, ObjectSpaces est une implémentation et, à ce titre, l'investigation s'avère plus délicate. N'en déplaise à Microsoft et pour la bonne cause (;-)), nous avons décompilé les sources du framework ObjectSpaces à l'aide de l'outil Salamander (www.remotesoft.com). Les résultats ont confirmé le contenu de la documentation du produit et nous a permis d'en savoir beaucoup plus sur le contenu du framework.

Tout d'abord, l'extension d'un objet métier est réalisée à l'aide du compilateur C# suivant le principe de la génération dynamique de code (cf article "génération dynamique de code" sur www.dotnetguru.org). Une fois la classe étendue, elle devient du type IObjectSpace qui peut être comparée à l'interface PersistenceCapable de JDO. Le Design Pattern utilisé ici ne fait aucun doute, c'est la Factory (Fabrique de classes), autre point de raccordement. La figure 4 résume le principe.

La première étape consiste à récupérer une instance de la Factory d'objets métier étendus de type IObjectSpace :

```
IObjectSpace os =
ObjectSpaceFactory.CreateObjectSpace("source.xml");
```

Notez que le fichier source.xml sert à spécifier le type de Factory utilisée : XML, SQL, OleDb, Spécifique, etc … Ensuite, il nous suffit de demander la création d'un objet métier étendu à l'aide de la méthode CreateObject() prenant en paramètre la clé primaire définie comme suit :

```
Facture theFacture =
(Facture)os.CreateObject(typeof(Facture), "1");
```

Contrairement à JDO, ObjectSpaces utilise des classes abstraites. L'utilisateur doit donc aussitôt après la création de l'objet initialiser les différents champs :

```
theFacture.ClientName = "DotNetGuru" ;
theFacture.Date = System.DateTime.Now ;
```

A ce stade, l'objet possède une identité et une instance en mémoire mais n'est toujours pas synchronisé avec la base. Il convient donc d'invoquer la méthode UpdateAll() du cache de la manière suivante :

```
os.UpdateAll() ;
```

Ce scénario, trivial en apparence, fait appel à l'ensemble de la chaîne d'exécution du Framework technique, de la création des instances à la mise à jour des données.

En résumé, vous remarquerez que ObjectSpaces et JDO adoptent une démarche relativement similaire à travers une conception centrée sur le Design Pattern Factory et une approche par génération dynamique de code.

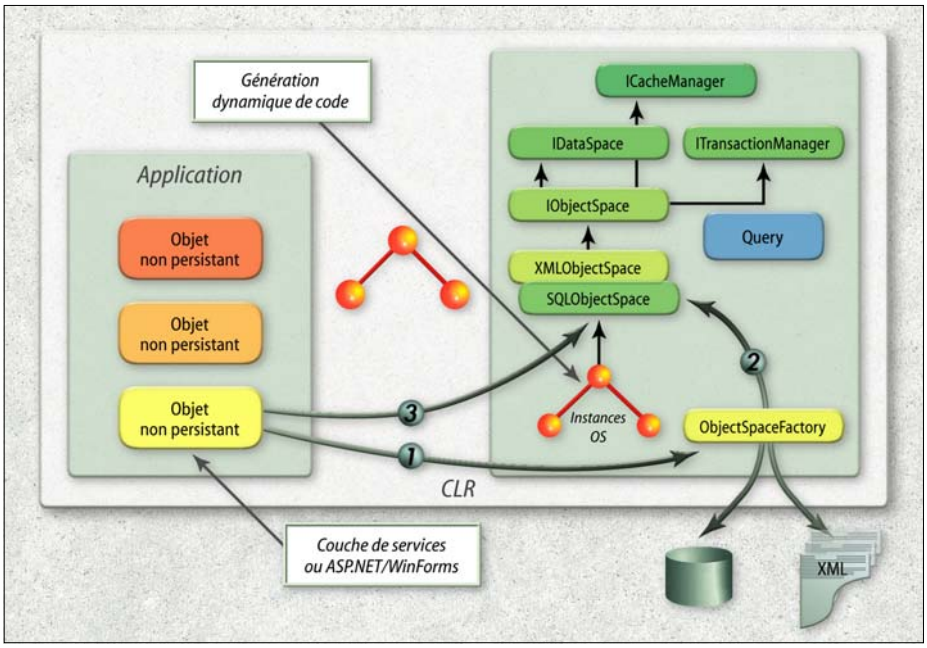


Figure 4. L'API ObjectSpaces.

Le Mapping Objet/Relationnel

Le mapping est la brique essentielle assurant la représentation objet du modèle physique. ObjectSpaces et JDO sont encore une fois très proche dans leur manière d'aborder les choses.

Concernant le mapping des types, JDO, de part sa nature, utilise les types et collections Java alors qu'ObjectSpaces se base sur le CTS (Common Type System) de .NET. Si en règle générale, ces frameworks proposent un mapping "un objet représente une table", dans la pratique cela tend à dégrader les performances et impose pour référence le modèle physique (granularité différente…). C'est pourquoi, JDO et ObjectSpaces à travers l'interface DataSpace proposent un mécanisme d'extension de la persistance afin d'enrichir le comportement de base du framework. Attardons-nous sur les mécanismes et les outils proposés de part et d'autre pour concevoir ce mapping.

JDO et ObjectSpaces utilisent tous les deux XML comme format du fichier destiné à décrire le mapping Objet/Relationnel. Chacun, à sa manière, formalise ce mapping sous la forme de DTD pour JDO et de Schemas pour ObjectSpaces.

Concernant les outils, la différence fondamentale provient du fait que ObjectSpaces propose à travers Visual Studio .NET un mapper graphique permettant de générer le fichier XML précédent. Ce mapper est en fait un plug-in tirant partie de l'interface de conception XML déjà existante afin de générer un schéma au format ObjectSpaces.

La gestion des transactions

La gestion des transactions prend une part importante dans chaque framework. Ainsi, les objets peuvent être modifiés ou supprimés dans le cadre d'une seule et même transaction globale avec ou sans intervention de l'utilisateur. JDO dissocie deux types de gestion :

- Transactions manuelles : le client spécifie explicitement la démarcation via les ordres pm.begin(), pm.commit() et pm.rollback()
- Transactions gérées automatiquement : lorsqu'une méthode d'un objet métier est appelée, le conteneur JDO effectue les appels commit() ou rollback().

De plus, JDO utilise deux modes de verrouillage à travers les transactions :

- Verrouillage optimiste (optimistic transaction management) : aucun verrouillage d'enregistrements n'est opéré. Au moment du commit() une comparaison des instances actives en mémoire et de la base de données est réalisée et en cas d'incohérence, une exception est levée. Le client décide ensuite de la marche à suivre.
- Verrouillage pessimiste (pessimistic transaction management) : utilise le verrouillage classique d'enregistrement lors des accès concurrents.

ObjectSpaces adopte une démarche similaire concernant le mode de verrouillage du fait de l'utilisation d'un objet DataSet sous-jacent. L'utilisateur a aussi la possibilité de gérer explicitement la démarcation via les ordres ObjectSpace.BeginTransaction(), ObjectSpace.CommitTransaction() et ObjectSpace.RollbackTransaction().

En résumé, JDO et ObjectSpaces proposent les mêmes fonctionnalités avec un léger avantage pour JDO qui met l'accent sur une gestion plus poussée des transactions avec le support du protocole XA dans le cas de conteneurs distribués.

Les langages d'interrogation

JDO adopte à travers l'interface Query le langage JDOQL (JDO Object Query Language) suivant plusieurs objectifs :

- La neutralité par rapport au langage de la base : le langage sous-jacent peut être du SQL, du OQL ou n'importe quel système spécifique (mainframe…)

• L'optimisation : il doit être possible de profiter de la puissance des langages sous-jacents tels que SQL pour étendre l'interface Query (requêtes complexes, jointures…).

• L'adéquation avec les environnements n-tiers : il doit être possible d'effectuer des requêtes entièrement en mémoire ou déléguer leur exécution à un système tiers.

• Le support des gros volumes de données : L'interface Query doit gérer le fait qu'un résultat renvoie de gros volumes d'enregistrements de manière optimale, c'est à dire sans instancier au préalable d'objets.

• Les requêtes pré-compilées : il doit être possible de pré-compiler une requête afin d'optimiser les appels ultérieurs.

Quant à ObjectSpaces, le langage OPath est à l'honneur. Ce langage propriétaire, tout comme JDOQL d'ailleurs, permet d'effectuer des recherches en utilisant un formalisme proche d'OQL. Nous vous invitons à parcourir l'article de DotNetGuru à ce sujet.

Le cache d'objets

Le cache d'objets constitue la plus value d'un framework de persistance. L'avantage du cache est de proposer à un utilisateur la manipulation des instances en mémoire afin d'éviter le surcoût lié à l'exécution d'une requête. Cependant, le cache d'objets s'avère pertinent dans un cas bien précis : lorsque les données sont en lecture seule où lorsque la majorité des instances ne sont pas amenées à être modifiées.

ObjectSpaces et JDO proposent tous les deux un cache transparent avec, pour JDO, la possibilité de spécifier explicitement l'éviction de certaines instances non utilisées dans le cache à l'aide de la méthode pm.evict(Object persistenceCapable). A l'heure où nous écrivons ces lignes, ObjectSpaces ne fournit pas ce type de méthode, l'interface IObjectCache comprend uniquement les méthodes Update() et Resync().

La spécification de JDO ne mentionne pas explicitement si le cache doit être distribué ou local, cette opération est laissée à l'appréciation du provider JDO. Toutefois, le développeur a la possibilité de bénéficier des services du framework EJB (Distribution, Sécurité, Synchronisation...) en association avec le cache d'instances JDO. D'ailleurs, la spécification décrit de manière bien précise la démarche à suivre.

Quant à ObjectSpaces, le produit actuel ne propose pas de cache distribué. Les équipes de Microsoft ont déjà fort à faire pour stabiliser le produit et décliner des versions compatibles avec d'autres bases de données (Oracle, Sybase, MySQL...).

JDO et ObjectSpaces sont des produits très similaires à plusieurs égards. L'approche utilisée de part et d'autre consiste à masquer l'ensemble des opérations de création, modification et suppression des instances à l'aide du Design Pattern Factory. De plus, l'impact du framework concernant les entités manipulées est très faible du fait de la transparence induite par l'extension de code (génération dynamique). L'objet Facture initial se transforme en objet technique PersistenceCapable sous JDO et en IObjectSpace sous ObjectSpaces. Aussi, cet article nous prouve bien qu'il existe de multiples façons de concevoir un framework de persistance, encore faut-il s'entendre sur un langage de requête commun. Enfin, ObjectSpaces et JDO nous démontrent une chose, l'avenir appartient aux frameworks qui feront la part belle au couplage faible entre objets métier et API de persistance. Finalement, tout l'inverse de ce que propose aujourd'hui les EJB entités… ●

ALLER PLUS LOIN

Spécifications officielles de JDO par Sun : JSR 000012 (release)
http://access1.sun.com/jdo

Génération dynamique de code (DotNetGuru)
http://www.dotnetguru.org/

Mapping objet relationnel avec Castor par Didier Girard :
http://www.devreference.net/article.php3?id=41

L'annuaire Web 2002 du développeur (4)

Ressources Web : HTML, CGI, JavaScript

Frédéric Milliot
est architecte logiciel et consultant en communication. Il collabore à la presse française et américaine depuis 1989.

Quatrième partie de notre dossier consacré aux meilleurs sites Web pour le développeur.

Ressources HTML/XHTML

Dynamic Drive

Présentation : ***
Contenu : ***
Niveau : **
http://www.dynamicdrive.com/

C'est sur ces pages qu'est rassemblée la plus complète des collection de scripts (D)HTML disponible sur le Web. Un exemple : dans la section Jeux, une des douze catégories de scripts disponibles, pas moins de dix applications ludiques complètes sont proposées. Avec bien sûr les incontournables, comme Tetris (deux versions avec et sans images) ou un casse-briques toutes options, mais également Phong, un ping-pong contre l'ordinateur avec intelligence artificielle, un testeur de réflexes ou un simulateur de tremblements de Terre. Bien sûr, il y a aussi du sérieux. Vous pourrez ainsi appliquer des effets de frame à tout document affiché, automatiser l'utilisation de la loupe d'IE 5.5+, ou déclencher toutes les opérations possibles en matière de contenu dynamique. En résumé, si vous avez aimé un effet ou une fonction DHTML sur un site, c'est ici que vous les trouverez.

DHTML Shock

Tout de même, on peut en faire des choses surnaturelles avec (D)HTML. Sur ce site, très bien tenu tant du point de vue graphique que de celui du contenu, un grand nombre d'effets spéciaux vous attendent, notamment des tracé de déplacement à la souris, des tooltips pour les mouseovers, une horloge dynamique qui fonctionne sans image, juste avec des couches (layers), des tickers d'actualité, des effaceurs de texte fonctionnant sans aucune surcharge pondérale à la Flash, ou encore des scripts de traitement d'images généralement liés à la taille de la zone active du navigateur. Bref, de tout un peu, avec en plus le plaisir de la découverte. Soit dit en passant, les scripts ici rassemblés son " crossbrowsers ", c'est à dire qu'ils fonctionnent avec tous les navigateurs un tant soit peu actuels. Terminons sur une critique : une fonction de prévisualisation en direct serait la bienvenue…

Présentation : ****
Contenu : ****
Niveau : **
http://www.cgi-resources.com/

WebFX

On ne trouvera que peu de scripts sur cette URL, mais la qualité de ce qui y est proposé est maximale. C'est bien simple : les résultats obtenus sont à la fois innovants et dépassent ce qu'on aurait crû possible en HTML. Ainsi, par exemple, vous pourrez agrémenter vos pages d'une horloge multiple (comme dans les salles de Bourse avec l'heure de Tokyo, New-York, Londres, etc.). Autre exemple, un bureau actif (active desktop) sur lequel on peut s'ajouter des Post-It électroniques. Les technologies mises en œuvre ici sont étendues, puisqu'elles touchent à la gestion de fichiers, à l'interface utilisateur (dialogues et paramétrages), ainsi qu'au skins

(quatre étant distribuées dans le package téléchargeable gratuitement). Dernier exemple pour la route : un Démineur pour IE5 et Mozilla qui constitue en soi un site de moins de 5 Ko (ce code a participé au concours annuel des sites Web de moins de 5 Ko http:// www.the5k.org). Bref, un peu de futile, mais il y a là beaucoup à apprendre en termes d'élégance de codage.

Présentation : ***
Contenu : ****
Niveau : ****
http://webfx.eae.net/

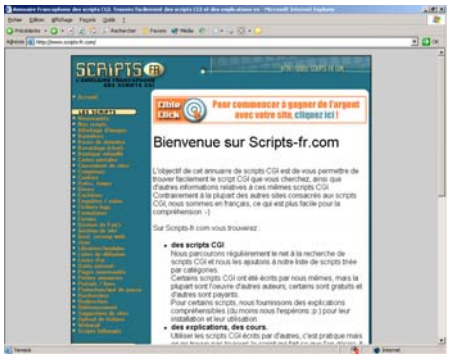
Cross-Browser.com

Cross-Browser vous offre CBE, une API DHTML conçue pour étendre les fonctionnalités DHTML à tous les navigateurs et, surtout, à en uniformiser les résultats. Après l'avoir téléchargée en local, on utilise CBE (aujourd'hui en version 4b10) pour développer les applications. Il suffit ensuite de copier les fichiers résultant et certaines librairies CBE sur son serveur pour que tout fonctionne à merveille. Comme son nom l'indique, Cross-Browser fonctionne sur à peu près tous les navigateurs du marché : IE, Netscape, Gecko, Opera, Mozilla et AOL - dès lors qu'il se fondent sur un modèle objet. Il faut également signaler que le site Cross-Browser peut servir d'IDE en ligne, avec sa propre fenêtre applicative et ses fenêtres de référence et de débogage. Pour vous convaincre de la pertinence de ce développement, n'hésitez pas à visiter les sites indiqués comme utilisant CBE, notamment la modélisation du système solaire avec informations astronomiques en popup par simple clic.

Présentation : ***
Contenu : ***
Niveau : ***
http://www.cross-browser.com

Ressources CGI / PERL

Scripts.fr



Ce site se présente légitimement comme "L'annuaire francophone des scripts CGI". S'il est moins exhaustif que ses homologues américains (voir plus bas), il n'en compte pas moins les mêmes fonctionnalités : date de MAJ, cotation par les internautes, indication de l'origine, du langage, de la plateforme. Mieux, une description sommaire accompagne les listings catégoriels, et se double d'explications très complètes, dans des pages à part, sur l'auteur, sur la structure du script, sur les pré-requis techniques, sur son implémentation et sur les éventuelles précautions à prendre. Cela ressemble à un travail de fourmi, mais le résultat est remarquable. Nous avons également apprécié la simplicité d'accès aux ressources. Les catégories (une petite quarantaine) sont précises et listées dès la page d'accueil, si bien qu'on accède aux scripts plus facilement qu'ailleurs. Ajoutons pour finir que les auteurs sont en train de préparer un cours de CGI en ligne, qui viendra s'ajouter au 10+ tutoriels déjà disponibles en français. Souhaitons-leur que les leçons sont d'aussi bonne qualité que le site.

Présentation : *****
Contenu : ****
Niveau : **
http://www.scripts-fr.com/

Perl-Gratuit

Le nom du site annonce la couleur, qui n'est donc pas celle de l'argent. Sur Perl-gratuit, vous trouverez bien sûr des scripts peu onéreux, notamment ceux de l'auteur. Ils sont accompagnés de commentaires et d'instructions d'implémentation, et des exemples sont fournis pour que vous puissiez évaluer avant de télécharger. Mais c'est surtout par son annuaire des scripts gratuits en français (plus de 330 au total) que ce site mérité un signet. Tous ont en effet été testés avant que d'être listés, si bien qu'on est certain d'un minimum de qualité. Les tests en question ne se résument d'ailleurs pas à un simple compte-rendu. Pour chaque script, une fiche technique est livrée, qui comporte des indications sur le temps d'installation, la facilité d'utilisation, les fichiers à configurer, les points forts et les points faibles et surtout, surtout, des copies d'écran (une ou plusieurs selon la nature du code). Comme sur Scripts.fr, le travail d'édition est impressionnant. Si vous passez par ce site, n'oubliez pas de féliciter l'auteur !

Présentation : ***
Contenu : ****
Niveau : **
http://www.perl-gratuit.com

CGI Extremes

A cette URL, c'est l'exhaustivité qui prévaut ! Si vous cherchez un CGI spécial, extraordinaire, atypique, exotique ou complexe, il y a des chances que vous le trouviez parmi les 2000+ qui vous sont proposés ici. Sept grandes catégories (Statistics, File Management, Site Management, Commerce…), elles-mêmes divisées en une quarantaine de sous-catégories (pour Commerce : enchères, petites annonces, paniers d'achat, transactions et financement…), orientent le développeur vers le sacré Graal avec une vraie logique. Un système de vote a été mis en place, mais son utilité dépend évidemment du nombre de votants (lui aussi indiqué). Les scripts proposés sont en grande majorité gratuits ; ceux qui ne le sont pas ont au moins le mérite d'être mentionnés, sachant qu'ils sont parfois les seuls à faire ce qu'ils font. Signalons enfin que le téléchargement ou l'affichage s'effectuent depuis les sites des auteurs, avec évidemment un risque de broken link - que vous pourrez signaler aux auteurs du site pour qu'ils mettent la page consultée à jour.

Présentation : ****
Contenu : ****
Niveau : **
http://www.cgiextremes.com/

CGI Resources

Dans la guerre des chiffres, ce site pourrait être vainqueur avec, le jour de notre dernière consultation, 4287 " ressources " CGI disponibles. C'est plus que sur CGI Extremes, mais toutes ces ressources ne sont pas nécessairement des scripts et, s'ils en sont, pas nécessairement en Perl. Car l'intérêt de ce site réside dans son approche " transverse " du scripting en CGI, comme on dit dans les grands comptes. Pour (continuer à) apprendre, il met à votre disposition près de 200 articles, tutoriels ou leçons sur tous les sujets désirables : variables d'environnement, cookies, headers HTTP, sécurité ou pré-requis du côté serveur… Si vous vous sentez noyé par la masse d'information offerte, un moteur de recherche est là pour parcourir toutes les sections à votre place. Vous pourrez même ajouter votre propre catégorie dans les différentes arborescences, au cas assez improbable où ce que vous cherchez ou développez n'y serait pas déjà.

Présentation : ****
Contenu : ****
Niveau : **
http://www.cgi-resources.com/

Ressources JavaScript/JScript

The JavaScript Source

Près de 1000 scripts Java à copier et coller directement depuis le site, voilà ce que vous trouverez chez JavaScript Source, hébergé par Internet.com. Classés en 23 sections (dont les ajouts les plus récents), les scripts sont accessibles à partir d'une liste à la Yahoo, présente également dans les pages constituant les rubriques, ce qui est on ne peut plus pratique. Pour chaque rubrique, la liste des contenus est assez détaillée, et inclut l'historique des mises à jour. Enfin, chaque script bénéficie d'une " page perso ", avec un descriptif parfois assez détaillé de l'objet du script, un mode d'emploi et d'implémentation, ainsi qu'une zone de copiage. Si vous préférez, le code pourra vous être e-mailé directement. Si vous laissez votre adresse électronique, vous pourrez également en profiter pour vous abonner à la newsletter JavaScript Source - en version texte ou HTML, ou en version résumée mensuelle.

Présentation : ***
Contenu : ***
Niveau : **
http://javascript.internet.com

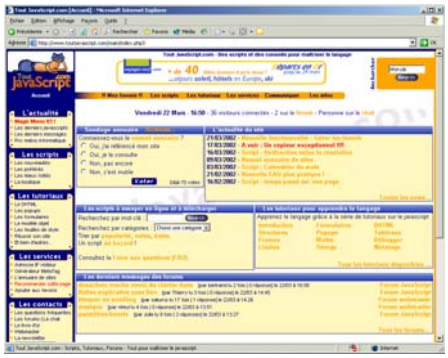
JavaScripts.com

Abondance de biens ne saurait nuire. C'est pourquoi nous vous présentons également ce site, en complément de The JavaScript Source. Les deux ont des vocations équivalentes : vous offrir autant de scripts gratuits et prêts à l'emploi que possible, pour que vous n'ayez plus rien à faire ou presque. Ici, un moteur de recherche en page d'accueil complète l'accès direct par rubriques (16 au total, y compris Mathematics, Business et Security). Dans chaque section, un descriptif sommaire donne accès à la page du script, laquelle est succincte au possible : le nom de l'auteur, la date de mise à jour, le prix (gratuit sauf exceptions) et la plate-forme requise (versions de Netscape ou de IE), ainsi que deux liens : l'un pour tester le script, l'autre pour en voir ou copier le code. Plusieurs contributeurs célèbres dans la communauté collaborent parfois au site en lui offrant des articles techniques dont certains sont excellents (même s'ils débordent le script cadre du Javascript). On pourra donc aussi apprendre à cette adresse, et avec les experts, s'il vous plaît !

Présentation : ***
Contenu : ***
Niveau : **
http://www.javascripts.com

Tout JavaScript

Premier site français de cette catégorie, Tout JavaScript n'est pas le plus exhaustif de la Toile en matière de Javascripts. Cela étant dit, ceux qui y sont ont été choisi pour leur utilité. Ainsi, entre autres, citons un redirecteur vers d'autres pages Web en fonction de la résolution d'écran de l'internaute, un détecteur de vitesse de connexion, un extracteur d'URL ou encore un moteur de chat (celui du site, en l'occurrence). Particularité de cette URL, tous les scripts sont testables en direct, si bien que l'on sait tout de suite à quoi on a affaire. Tout Javascript vaut également pour les tutoriels qu'il propose, plus de 25 traitant chacun d'un sujet précis (les cookies, les tableaux Javascript, Javascript et PHP) et agrémentés de codes et d'exemples complets.



Présentation : ***
Contenu : ***
Niveau : **
http://www.toutjavascript.com

JavaScriptFR

Voilà un site à l'esthétique particulière, mais que nous trouvons réussie. C'est toutefois pour son français qu'il prend place lui aussi dans ces colonnes. Moins riche que ses équivalents américains, il permet néanmoins un accès multi-critérisé à ses contenus, par ailleurs classés par niveau (1 = débutant, 2 = intermédiaire, 3 = expert). Quelques exemples de scripts : un jeu de 421, un menu arborescent 100 % JavaScript, des barres de défilement en couleur ou la détection automatique d'une adresse IP. Communautaire, JavaScriptFR héberge également un forum, destiné à offrir à chacun un coup de main plus ou moins immédiat, mais dont les questions et contributions sont pour la plupart de niveau débutant, ainsi qu'un newsgroup où, cette fois, la lecture est un peu plus enrichissante. Si vous parlez peu l'anglais et que JavaScript est votre lot quotidien, vous y trouverez peut-être votre bonheur.

Présentation : ****
Contenu : ***
Niveau : **
http://www.javascriptfr.com