

**Tendance**

# EJB 3 et JDO 2 pour unifier la persistance en Java

Après l'adoption de JDO 2, le standard permettant de stocker des objets Java dans des bases transactionnelles, EJB 3, attendu pour 2006, va-t-il enfin réconcilier les deux approches ?

**C'**est l'un des principaux défauts de la plate-forme J2EE, et bizarrement, il reste sans véritable réponse formelle de Sun à ce jour. La persistance des données dans une architecture d'entreprise Java est certes un problème technique restreint, mais de sa mise en œuvre peut dépendre la réussite ou l'échec d'un projet entier. Alors que le vote de JDO 2, un standard de persistance des données en environnement Java, aurait dû signifier la fin de cet épineux problème, la polémique sur le sujet a pris de l'ampleur avec le projet de spécification de la troisième version des EJB (composants Enterprise Java Beans) attendue pour 2006.

Une application J2EE est constituée d'objets qui manipulent des données. Celles-ci doivent pouvoir être extraites de différentes sources (généralement des bases de données, éventuellement mises à jour, puis réintégrées en base. Ce mécanisme, dit de persistance, est un processus sans souci avec une base de données objet. Mais dans la réalité, les SGBDO n'ont pas décollé, et l'immense majorité des projets recourt à une base relationnelle... se confrontant ainsi au hiatus objet-relationnel : il faut découper les objets en données à stocker dans des tables puis les recomposer. La plupart des développeurs codent tout simplement cette gymnastique d'extraction et de mise à jour au sein même de l'applica-

## OUTILS DE GESTION DE LA PERSISTANCE EN ENVIRONNEMENT JAVA

Logiciel	Editeur	Commentaires	Tarifs
<b>IMPLÉMENTATIONS DE JDO</b>			
FrontierSuite	ObjectFrontier	Basé sur JDO 1. La version Entreprise offre une fonction de cache distribué.	n. c.
JCredo JDO	Riflexo	L'équipe italo-bulgare de Riflexo s'attache à soigner l'intégration de JCredo au sein des ateliers Eclipse, JBuilder et JDeveloper.	De 0 à 100 euros pour l'édition standard, 500 euros pour l'édition professionnelle, 1 500 euros pour l'édition entreprise.
JDO Toolkit	MVCSoft	Offre aussi une possibilité de transformation des données.	139 dollars par développeur.
JPox	Sourceforge/ Apache	Sélectionné par Sun en décembre dernier comme implémentation de référence. Adopté par le projet Apache DB.	Gratuit (licence Open Source Apache).
Kodo JDO	Solarmetric	La version Entreprise, pour serveurs J2EE, comporte des améliorations comme la gestion des transactions distribuées. A été proposé en OEM par Versant.	De 600 à 1 100 dollars par développeur pour la version standard, de 3 000 à 4 000 dollars pour la version Entreprise.
Lido	Xcalia	Ancienne référence sur le marché français, Xcalia étant le nouveau nom de Libelis.	Gratuit pour le développement. Editions Professional et Enterprise payantes (prix n. c.).
TriActive JDO	Sourceforge	Basé sur JDO 1. La dernière version stable date d'un an.	Gratuit (licence Open Source GPL).
VOA	Versant	Issu du rachat de JDO Genie en juin 2004. L'édition Entreprise concerne les déploiements J2EE.	995 euros par développeur pour l'édition professionnelle, 3 000 euros pour l'édition Entreprise.
Xorm	Sourceforge	Basé sur JDO 1. La dernière version stable date de dix mois.	Gratuit (licence Open Source GPL).
<b>OUTILS DE CORRESPONDANCE RELATIONNEL-OBJET</b>			
Cocobase	Thought	Prévoit d'implémenter JDO 2.	6 000 dollars par développeur.
EdgeXtend	ObjectStore	Spécialisé dans la gestion de cache distribué des données, repris par Progress.	n. c.
Hibernate	JBoss	Proche des spécifications retenues pour EJB 3.	Gratuit (licence Open Source LGPL).
OJB	Apache	Prévoit d'être totalement compatible JDO dans sa version 2.	Gratuit (licence Open Source Apache).
Toplink	Oracle	Le produit, acquis par Oracle en 2002, a plus de dix ans d'existence.	77 euros par développeur, 3 862 euros par processeur (ou inclus dans OracleAS) au déploiement.

tion – un spécialiste du domaine estime ainsi que le recours au codage manuel en JDBC pour accéder aux bases de données représente 60% des projets existants (\*). Cette façon de faire offre un bénéfice hypothétique – du code parfaitement optimisé, à condition que le développeur domine son sujet – et des inconvénients tout à fait réels : temps de développement beaucoup plus long, et surtout une maintenance plus complexe qui obère la pérennité du projet. "Si vous changez le schémas d'une table, tout le code est à réécrire", explique Frédéric Lejeune,

directeur associé de la SSII Sysdeo et responsable des formations. "Il faut automatiser, c'est le premier conseil que je donne, renchérit Benoît Moussaud, architecte J2EE chez Unilog. *Quelle que soit la solution technologique retenue ensuite, il faut un outil, ne serait-ce que pour encapsuler le JDBC.*"

### Les EJB ont échoué

La première réponse des éditeurs au besoin d'automatisation de cette étape a été de créer une variété d'EJB. Dite CMP (Container Managed Persistence), elle était censée prendre

en charge cet aspect persistance, mais il s'est vite avéré que son utilisation grevait considérablement les performances de l'application. "La CMP avait l'air séduisante sur le papier, mais presque tous les projets l'utilisant en sont revenus, note Benoît Moussaud. La programmation était très complexe et ne fonctionnait que pour un volume faible."

Toplink, précurseur des outils de mapping (correspondance relationnel-objet-ORM), a construit son succès en offrant un moyen beaucoup plus simple de gérer la persistance : le développeur établit

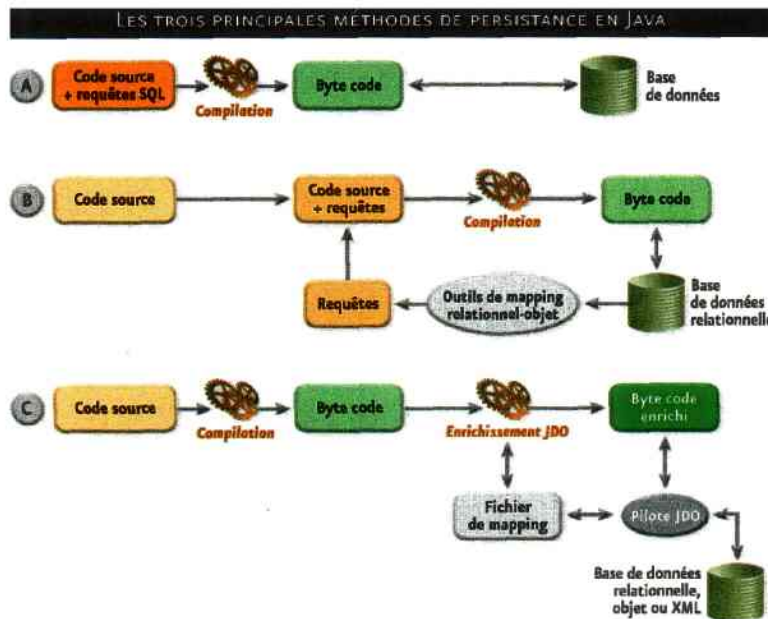
un fichier de correspondance entre le schéma de la base de données et les objets de l'application, l'outil générant les requêtes JDBC ad hoc. Ainsi, le code de l'application elle-même et le code d'accès aux bases peuvent évoluer de façon indépendante. Et le risque de requêtes JDBC incorrectes est moindre. Quelques outils, notamment en Open Source, ont cherché à concurrencer Toplink. Avec le rachat de cedemier par Oracle et l'intégration du projet Open Source Hibernate au sein des solutions de JBoss, qui s'est taillé une part de marché conséquente dans les ser-

veurs d'applications, le marché aujourd'hui se focalise autour de ces deux produits. "Le choix de Toplink peut paraître naturel pour les clients d'Oracle, mais cela peut aussi représenter un point bloquant pour certains clients, qui ont peur de se retrouver pieds et poings liés avec Oracle, note Frédéric Lejeune. Côté Open Source, le produit en vogue aujourd'hui est Hibernate : nos demandes de formation explosent. C'est un bon produit, mature, que nous proposons spontanément aux clients qui démarrent."

## JDO émerge lentement

Entretemps, une autre voie a émergé sous l'impulsion de Sun, qui est allé chercher un spécialiste de l'objet chez Versant, Craig Russell, pour travailler sur une façon standard de stocker des objets Java dans des bases transactionnelles : JDO (Java Data Objects). Objectif : construire une application Java sans savoir a priori quelle base sera utilisée. "Avec un outil de mapping, le modèle Java colle au plus près la base, alors que JDO part du modèle objet, remarque Benoît Moussaud. C'est une vision de plus haut niveau, où l'on part du modèle objet, que l'on cherche ensuite à rendre persistant, dans un SGBD relationnel, objet, XML..." Avant de travailler pour Unilog, Benoît Moussaud avait ainsi travaillé sur une application pour Cril Telecom Software, basée sur le SGBD de Versant. Face aux réticences des clients vis-à-vis des SGBD, Cril Telecom était passé à un pilote JDO, Lido, le rendant capable de s'adapter au SGBD de chacun de ses clients. Lors de la conception de son progiciel e-Citiz, Genigraph a choisi JDO pour les mêmes raisons.

Le JCP (Java Community Process) étudie rapidement la question : JDO est sa douzième demande de spécification (JSR12 - Java Specification Request). Mais la version 1 du standard ne voit le jour qu'en 2002, trois ans après les premières discussions. Et encore, il est restreint. Il ne régit pas, par exemple, la façon dont la correspondance doit être établie entre une base relationnelle et un objet. Pour les experts du sujet, il s'agit d'une omission volontaire, pour ne pas froisser les éditeurs d'ORM. Vu la guérilla de



**A] A la main :** l'ensemble du code est écrit manuellement, y compris les requêtes pour accéder à la base de données cible. Il est possible d'optimiser les performances, mais toute évolution de la base nécessite de revoir la majeure partie du code applicatif.

**B] ORM :** le développeur établit à l'aide d'un outil une correspondance entre le code applicatif et la base de données relationnelle cible. L'outil génère le code de requête. Une évolution de la base se traduit par un travail moindre et automatisé.

**C] JDO :** l'applicatif délègue la gestion de la persistance au pilote JDO. C'est lui qui génère le code d'enrichissement de l'applicatif. En théorie, l'application devient très facilement portable d'une base à une autre.

JBoss et Oracle face à la version 2 du standard (voir l'encadré ci-dessous), qui inclut cette notion de mapping, l'analyse doit être juste...

## EJB 3, la réconciliation ?

Côté ORM, on explique que le discours des partisans de JDO manque de pragmatisme. "Ce que recherchent les développeurs et le marché, c'est de la persistance relationnelle, indique ainsi Emmanuel

Bernard, membre de l'équipe Hibernate et du groupe d'expertise EJB3. Le développement d'applications performantes en faisant totale abstraction (dans sa connaissance) de la technologie sous-jacente (les bases de données relationnelles) n'est qu'une chimère." Emmanuel Bernard voit dans la troisième version des EJB, en cours de définition, la possibilité de réconcilier les différentes approches, comme Sun l'a indiqué dans une lettre

d'intention. De fait, des membres du groupe de travail JDO se retrouvent dans le groupe EJB, et les spécifications pour la persistance sont très proches des mécanismes utilisés par les ORM. "EJB3 a toutes les cartes en main", dit Emmanuel Bernard, à commencer par le soutien de Sun (sans doute lassé du jusqu'au-boutisme objet de Craig Russell) et des grands éditeurs de serveurs d'applications J2EE. A terme, explique Eric Samson, directeur

technique d'Xcalia, JDO devrait continuer d'exister en tant que complément des EJB 3 : "D'ores et déjà, EJB 3 apparaît comme une remise en cause totale des EJB et une reconnaissance des idées de JDO. Et comme EJB 3 ne s'occupera que du problème relationnel, il y aura besoin d'un standard pour les autres sources de données." ●

OLIVIER RAFAL

(\*) Voir <http://roller.com/page/jizhou>

## JDO 2, le vote tourne au psychodrame

La deuxième version du standard JDO (Java Data Object), qui vise à rendre le développement d'applications Java indépendant de la technologie retenue pour la base de données sous-jacente, a été approuvée le 28 février dernier, au terme d'un mois de tensions. Le drame se noue le 19 janvier dernier, quand le standard est rejeté par la grande majorité des votants. La communauté des utilisateurs de JDO s'émeut. Jean-Claude Bellando, directeur commercial de Versant France, est un des rares à s'agiter en France : "Les gens veulent un standard de persistance tout de suite, sans attendre les EJB 3, le tout étant

de trouver une solution de convergence vers EJB 3 pour la suite." Il se veut toutefois confiant dans l'issue du second vote, programmé un mois plus tard : "JBoss et Oracle seront les plus durs à convaincre, mais HP, Intel ou SAP sont plutôt de bonne foi dans leurs demandes d'explication." Ces derniers reprochent en effet à Sun de ne pas être clair sur la façon dont les mécanismes de persistance des EJB 3 et de JDO pourraient converger. Google, qui s'est abstenu, évoque un risque de fracture de la communauté J2EE. Le 26 janvier, les chefs de projet EJB 3 et JDO 2 publient une lettre ouverte (\*) expliquant que les EJB 3 fourniront un modèle de persistance unifié, tant

pour les architectures J2EE que J2SE, et que JDO sera rendu compatible avec ce modèle. La plupart des opposants jugent ces explications satisfaisantes, BEA estimant même qu'il existe "une communauté JDO qui a besoin de cette spécification", qu'il ne faudrait pas "tenir en otage" dans l'attente de l'évolution des EJB. Lors du second vote, Oracle et JBoss finissent par s'abstenir à l'instar d'IBM, non sans manifester leur mauvaise humeur. Pour JBoss, le rapprochement entre JDO 2 et EJB 3 "pourrait donner le sentiment d'un coup pour mentir ce dernier". O. R.

(\*) A lire sur <http://java.sun.com/j2ee/letter/persistence.html>

## POUR EN SAVOIR PLUS

- [www.jdoentral.com](http://www.jdoentral.com)  
Portail d'infos anglophones.
- <http://jcp.org/en/jsr/detail?id=243>  
Page consacrée à JDO 2 sur le site du JCP (=12 pour la page sur JDO 1, =220 pour EJB 3).
- <http://bmoussaud.developpez.com/tutoriel/java/jdo/>  
Mise en œuvre de JDO.
- <http://javarome.free.fr/net/JDO.html>  
Explications et FAQ en français.
- <http://c2.com/cgi-bin/wiki?ObjectRelationalToolComparison>  
Tableau comparatif en anglais.
- <http://incubator.apache.org/ibatis/site/about.html>  
Alternative à Hibernate et JDO.