Avant-propos

Lorsque les standards n'existent pas ou qu'ils tardent à proposer une solution universelle viable, certains outils, frameworks et projets Open Source deviennent des standards de fait. Ce fut le cas de Struts, d'Ant ou encore de Log4J, et surtout d'Hibernate ces cinq dernières années. L'originalité d'Hibernate en la matière a été de s'imposer, alors même qu'existaient deux standards, EJB 2.0 et JDO 1.0.

Deux ans après son lancement, le succès d'Hibernate est tel qu'il a amorcé la spécification de persistance des EJB 3.0. Après plusieurs années d'attente, Java SE et EE se dotent enfin de la brique critique indispensable qui lui manquait jusqu'à présent : Java Persistence.

Pourquoi revenir sur Hibernate alors que cet ouvrage est dédié à Java Persistence ? La spécification Java Persistence couvre une large majorité de cas d'utilisation. Cependant, dès que la complexité ou l'amélioration spécifique de points de performance se fait ressentir, ce sont les implémentations qui prennent le relais en proposant des solutions malheureusement non portables. Dans ce domaine, Hibernate possède la plus grande maturité et le plus large éventail de fonctionnalités supplémentaires.

Les entreprises ont un besoin crucial d'outils fiables et robustes, surtout dans le domaine de la persistance, qui est au cœur même des applications. Un produit qui implémente la spécification Java Persistence n'est pas forcément fiable et performant ; il permet simplement d'exploiter des interfaces standardisées. Il faut donc choisir avec la plus grande attention l'implémentation à utiliser. Sans préjuger de la qualité des différents fournisseurs du marché, il est indéniable qu'Hibernate, de par son histoire et les extensions qu'il propose, est un choix sûr et pérenne.

Java Persistence facilite le traitement de la persistance dans vos applications Java, lequel peut représenter jusqu'à 30 % des coûts de développement des applications écrites en JDBC, sans même parler des phases de maintenance. Une bonne maîtrise de Java Persistence, mais aussi des fonctionnalités spécifiques proposées par Hibernate, accompagnée d'une méthodologie adéquate et d'un outillage ciblé sur vos besoins peuvent vous faire économiser de 75 à 95 % de ces charges.

Au-delà des coûts, les autres apports d'Hibernate sont la qualité des développements, grâce à des mécanismes éprouvés, et le raccourcissement des délais, tout un outillage associé facilitant l'écriture comme la génération du code.

Objectifs de l'ouvrage

La gratuité d'une implémentation de Java Persistence telle qu'Hibernate ne doit pas faire illusion. Il est nécessaire d'investir dans son apprentissage puis son expertise, seuls gages de réussite de vos projets.

Pour un développeur moyennement expérimenté, la courbe d'apprentissage de Java Persistence est généralement estimée de quatre à six mois pour en maîtriser les 80 % de fonctionnalités les plus utilisées. Cela comprend non pas la maîtrise des syntaxes et API, mais bel et bien le raisonnement en termes objet ainsi que la prise en compte de la propagation et de l'interaction avec la base de données.

Dans ce domaine, il est indispensable de raisonner aussi bien globalement (macrovision applicative) qu'en profondeur (évaluation minutieuse de chaque action et paramétrage). Si vous ajoutez à cela que, tôt ou tard, il sera nécessaire d'exploiter une ou plusieurs fonctionnalités avancées d'Hibernate, l'apprentissage s'en trouve allongé d'autant.

Pour toutes ces raisons, il m'a semblé utile de fournir aux développeurs les moyens d'apprendre l'exhaustivité du produit relativement vite, de manière concrète et avec un maximum d'exemples de code. Tel est l'objectif principal de cet ouvrage.

Java Persistence with Hibernate, l'ouvrage de Christian Bauer, coécrit avec Gavin King, le créateur d'Hibernate, est une bible indispensable pour appréhender la problématique de la persistance dans les applications d'entreprise dans ces moindres recoins. Moins théorique et résolument tourné vers la pratique, le présent ouvrage se propose d'illustrer chacune des fonctionnalités de l'outil par un ou plusieurs exemples concrets.

L'objectif principal de ce livre est d'aller droit au but, en introduisant la totalité des problématiques et en détaillant de manière exhaustive la solution à mettre en application.

J'ai voulu que cet ouvrage soit aussi complet que possible, mais aussi agréable à lire, didactique et dynamique, avec des exemples parfois complexes à mettre en œuvre mais dont l'intérêt est toujours facile à cerner.

Questions-réponses

Cet ouvrage est-il une seconde édition d'*Hibernate 3.0 Gestion optimale de la persistance dans les applications Java/J2EE* ?

Non, même s'il est vrai que les deux ouvrages se ressemblent beaucoup. La raison de cette ressemblance est simple : les notions relatives à la problématique de la persistance n'ayant guère évolué, il est logique qu'en termes théoriques les deux ouvrages se rejoignent. J'ai délibérément choisi d'exploiter la même logique d'exemples et de reprendre la plupart des cas d'utilisation, compte tenu des critiques favorables reçues par mon ouvrage précédent. La mise en œuvre des exemples est cependant totalement différente. Elle repose sur un conteneur léger et permet d'aborder aussi bien des exemples en environnement autonome (Java SE) qu'en environnement d'entreprise (Java EE).

Par ailleurs, les formats et API abordés sont totalement différents, la technologie employée étant elle-même différente.

Cet ouvrage porte-t-il sur Hibernate ?

Oui et non. Non, car le noyau d'Hibernate a toujours reposé sur une définition des métadonnées écrite au format hbm.xml et sur l'utilisation de sessions Hibernate. Nous ne détaillons ni l'un ni l'autre dans ce livre, mais nous focalisons sur le style Java Persistence, à savoir la définition des métadonnées par les annotations et l'utilisation de l'API principale EntityManager, en passant en revue de manière pratique l'intégralité de la spécification. En complément, nous fournissons le détail des fonctionnalités supplémentaires spécifiques fournies par Hibernate, mais toujours en exploitant le style Java Persistence.

Cet ouvrage détaille-t-il les descripteurs de déploiement d'entités orm.xml et les fichiers de mapping Hibernate hbm.xml ?

Non. Concernant les fichiers de mapping Hibernate hbm.xml, la réponse est fournie à la question précédente. Si toutefois ce format vous intéresse, l'équivalent des chapitres 3 et 4, dédiés aux métadonnées, est fourni au format hbm.xml sur la page dédiée au livre du site Web d'Eyrolles. Il s'agit de deux des chapitres de mon précédent ouvrage, peu de chose ayant changé depuis lors concernant ce format.

En ce qui concerne le format standardisé orm.xml, les raisons de préférer ce format aux annotations sont peu nombreuses. L'important est de comprendre une fonctionnalité. Si vous comprenez un paramétrage *via* les annotations, il vous est facile de vous référer à la spécification pour la traduire au format orm.xml.

Où peut-on trouver les exemples de code ?

Les exemples de code sont disponibles sur la page dédiée à l'ouvrage du site Web d'Eyrolles, à l'adresse *www.editions-eyrolles.com*. Ils ont été conçus comme des tests unitaires afin que vous puissiez les exécuter facilement et y insérer des assertions.

L'intégration avec Spring ou Struts est-elle abordée ?

J'ai reçu une critique particulière concernant le fait que mon précédent ouvrage n'abordait pas en détail l'intégration avec Struts. Aujourd'hui, Struts est beaucoup moins utilisé qu'il y a quelques années et continuera de perdre des utilisateurs au profit notamment de JSF.

Cet ouvrage porte sur Java Persistence et non sur Struts ni Spring.

Comment devenir contributeur du projet Hibernate ?

Il n'y a rien de particulier à faire. Hibernate est le fruit d'une interaction intense entre les utilisateurs, les contributeurs et l'équipe d'Hibernate.

Si vous êtes motivé pour participer à l'évolution d'Hibernate, plusieurs axes peuvent vous intéresser, notamment les suivants : développement de nouvelles fonctionnalités (généralement réservé aux développeurs expérimentés), évolution des outils ou des annotations, documentation, etc.

Organisation de l'ouvrage

La structure de cet ouvrage a parfois été un casse-tête. Il a fallu jongler dès le début entre la configuration de la persistance *via* les annotations et l'utilisation à proprement parler des API de Java Persistence, le tout sans répéter le guide de référence de l'outil, qui est sans doute le plus complet du monde Open Source.

- **Chapitre 1.** Propose un historique et un état des lieux de la persistance dans le monde Java ainsi que des solutions actuellement disponibles sur le marché. Il présente un exemple très simple d'utilisation de Java Persistence.
- **Chapitre 2.** Décrit le raisonnement à adopter lorsque vous utilisez un mappeur objet-relationnel. Le vocabulaire est posé dès ce chapitre, qui montre également comment installer JBoss intégré, la base d'exécution des exemples qui illustrent ce livre.
- **Chapitre 3.** Montre comment annoter vos entités et propose un référentiel des métadonnées.
- **Chapitre 4.** Apprend à maîtriser les notions abordées dans les trois premiers chapitres. À ce stade de l'ouvrage, vous commencez à entrer dans les fonctionnalités avancées de mapping. Dans ce chapitre, vous découvrirez certains principes avancés de modélisation et les indications indispensables pour mapper vos choix de modélisation.
- **Chapitre 5.** Dédié aux techniques de récupération d'objets. Vous verrez qu'il existe plusieurs méthodes pour interroger le système de stockage de vos objets (la base de données relationnelle).
- **Chapitre 6.** Décrit en détail comment considérer la création, la modification et la suppression des entités. Vous y apprendrez comment prendre en compte la concourance dans vos applications et aborderez la notion de persistance transitive.
- **Chapitre 7.** Présente les techniques les plus répandues pour manipuler le gestionnaire d'entités et propose plusieurs best practices permettant de mettre en œuvre une gestion simple et optimale du gestionnaire d'entités, que votre environnement soit autonome (Java SE) ou d'entreprise (EE).
- **Chapitre 8.** Revient sur certaines fonctionnalités très poussées, spécifiques de l'implémentation de Java Persistence fournie par Hibernate.
- **Chapitre 9.** Se penche sur l'outillage disponible autour d'Hibernate.
- **Chapitre 10.** Traite de la problématique d'intégration des caches de second niveau et des pools de connexions. Il détaille aussi l'utilisation du projet annexe Hibernate Validator.

Organisation des exemples

Vous pouvez télécharger trois projets depuis la page Web dédiée au livre :

- java-persistance : ce projet couvre 90 % des exemples du livre. Le nommage des packages vous permettra très facilement de cibler les sources à analyser.