BOOKS FOR PROFESSIONALS BY PROFESSIONALS™

Using and Understanding Java Data Objects

精通 JDO

- 深入分析 JDO 的规范和行为
- 涵盖主要的应用程序体系结构，包括客户 / 服务器应用程序、Web 应用程序以及使用 Enterprise JavaBeans 的应用程序
- 使用开放的源代码示例
- 涵盖 JDO 1.0 和 JDO 1.0.1

(美) David Ezzio 著
韩来彬 译

清华大学出版社

精通 JDO

(美) David Ezzio 著

韩来彬 译

清华大学出版社

北 京

EISBN: 1-59059-043-0
Using and Understanding Java Data Objects
David Ezzio
Original English language edition published by Apress L. P., 2560 Ninth Street, Suite 219, Berkeley, CA 94710 USA.

图书在版编目(CIP)数据

精通 JDO/(美)埃兹尔(Ezzio, D.)著；韩来彬译. —北京：清华大学出版社，2005.6
书名原文：Using and Understanding Java Data Objects
ISBN 7-302-10696-7
I. 精… II.①埃… ②韩… III. JAVA 语言—程序设计 IV. TP312
中国版本图书馆 CIP 数据核字(2005) 第 022893 号

出 版 者：清华大学出版社　　地　　址：北京清华大学学研大厦
　　http://www.tup.com.cn　　邮　　编：100084
　　社 总 机：010-62770175　　客户服务：010-62776969
组稿编辑：曹　康　　文稿编辑：于　平
封面设计：康　博　　版式设计：康　博
印 刷 者：清华大学印刷厂　　装 订 者：三河市李旗庄少明装订厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　印张：18　字数：461 千字
版　　次：2005 年 6 月第 1 版　2005 年 6 月第 1 次印刷
书　　号：ISBN 7-302-10696-7/TP・7234
印　　数：1～4000
定　　价：33.00 元

前　言

Java 数据对象(Java Data Objects，JDO)为 Java 对象指定了一个透明的持久性服务。此规范为存储和检索 Java 对象的持久状态描述了一个通用框架。JSR-12 专家组通过 Java 标准化组织(Java Community Process)定义了 JDO。JDO 包括规范、参考实现和技术兼容性工具包(technology compatibility kit, TCK)。这三个部分都可以从 Java 标准化组织的网站(http://www.jcp.org/jsr/detail/12.jsp)下载。许多供应商都开发了 JDO 的实现工具，并提供了其产品的多个版本。您可以从网上下载这些产品。在前言的末尾可以找到其中一些供应商的 URL。

Java 程序员使用工具成功开发他们的项目。大部分的应用程序和服务器端的组件都涉及到了内存中的对象和持久存储中的对象之间的数据移动。JDO 允许改变 Java 程序员存储和检索对象状态的方式。JDO 简化了应用程序代码，提高了应用程序的可移植性，而且还有助于分离在应用程序设计时涉及到的问题。与现有的可替代品相比，JDO 使持久性更易于理解、建模和编码。

本书读者对象

本书读者对象是 Java 程序员和应用程序架构师。本书假定您已经知道如何用 Java 编程，并假定您打算使用 JDO，也知道 JDO 的工作原理。本书重点介绍了为有效利用 JDO 您需要知道的知识。

但是，决不是说您需要理解书中的所有内容才能有效地应用 JDO。JDO 封装了数据存储的许多细节，从而易于使用。但是，因为许多为大家所熟悉的关于数据存储和提取的内容从视图中隐藏起来了，所以 JDO 可能会让人产生混淆。本书试图提供足够的信息让您能够理解 JDO 的细节及其行为。根据您的背景和当前需要，您可能会发现本书中的一些部分可以略读或跳过。

学习 JDO 和学拉小提琴一样简单。它易于上手，并几乎可以立即发出“声音”。但是如果您的目的是创作“音乐”，那就需要知识和技巧了。本书的目的就是帮助您获得成功使用 JDO 所需的知识和技巧。

JDO 的概念

JDO 为一些对象指定了一个持久性服务，这些对象的状态比 Java 虚拟机(Java Virtual Machine, JVM)的单个调用的生命期长。几乎每个程序都保存某些对象的状态，并且在随后的调用中恢复该状态，或者和另一个并发运行程序共享它。被保存的状态既是对象内的状态又是对象间的关联。

由于 JDO 将内存中的 Java 对象和其状态的长期存储相连接，因此它是持久性服务。通过

使用一个只关心持久性的服务，应用程序把内存和数据存储之间的状态移动工作孤立于某一段代码。如果没有持久性服务，那么用于连接数据存储的代码就会遍布整个应用程序。通过使用JDO，应用程序程序员可以使持久性成为一个模块化组件，从而提高了应用程序的可维护性。这有利于找出 bug 并修复 bug、避免产生新的 bug 以及提升应用程序的性能。

虽然大部分应用程序都需要让一些对象持久存在，但并不是所有的对象都需要持久性。程序中的许多对象都只是瞬态对象。这意味着它们的状态要根据需要创建，并在程序停止执行时丢弃这些状态(如果在这之前还没有丢弃这些状态的话)。有些对象是瞬态对象，但是要用到持久对象或者其中的信息。例如，用于计算持久性发货单的总销售额的对象本身可以是瞬态对象。

JDO 指定了一致的持久性服务

JDO 为应用程序可以持久使用的服务库指定一个应用程序编程接口(API)。JDO 还指定了持久服务，这些持久服务被透明地提供给所有持久对象。JDO 附带了可能适用于原型开发的 API 参考实现。许多供应商已经开发了新产品，或者适应已有的产品以支持 JDO 规范。这些产品将会在健壮性、性能、所支持的数据存储、对可选的 JDO 特性的支持、易于配置、灵活性等方面展开竞争。

JDO 封装并简化了程序设计问题，这些问题是持久对象所独有的。JDO 提供了一个用于查找、更新和删除现有持久对象的服务以及存储新的持久对象的服务。它封装了内存中的对象状态与数据存储中的持久状态之间的映射。它确保把对内存中的持久对象的多次改动以要么全有要么全无的方式存储到数据存储中。最后，它允许并发执行多个应用程序和线程，以共享对持久状态的访问。

JDO 提供了一组一致的持久性服务。这些服务不随所采用的数据存储和所选择的部署环境而改变。JDO 实现可以被生成用于一系列的数据存储体系结构，如文件、对象数据库、关系数据库以及通用的事务处理服务等。由于封装了数据存储，JDO 允许应用程序使用同一组服务而不必理会选择的是何种数据存储。JDO 的一致性允许应用程序忽略查询语言、数据模型以及不同数据存储间的访问接口之间的区别。当涉及到部署体系结构时，那些使用 JDO 的应用程序代码可部署在独立的应用程序、客户/服务器应用程序、servlet 以及 EnterPrise JavaBeans(EJB)中。在大部分情况下，同一组 JDO 服务在所有这些环境中都是可用的。

大部分 JDO 持久性服务都是透明的

在使用 JDO 时，程序员在应用程序数据类(这些类定义了有持久状态的对象)中不用编写与持久性有关的代码。程序员在创建应用程序数据类的时候不把持久性当作一个问题。程序员声明那些包含了对象状态的字段，并编写那些访问和修改内存中对象状态的方法。

JDO 要求有一个用于所有应用程序数据类的增强步骤。JDO 供应商提供了一个被称为 enhancer(增强器)的工具，程序员用它给应用程序数据类添加与持久性相关的代码。增强的结果是，JDO 把应用程序数据对象当作持久对象管理。由于增强通常在编译后发生，因此应用程序数据类的源代码中与持久性相关的代码通常是不可见的。

透明持久性的功能十分强大，因为它不要求程序员编写任何代码，而且它强加给设计和编码的限制很少。当一个应用程序使用了 JDO 时，它就有效地省略了应用程序数据类的源代码中

的持久性代码。但是在运行时，数据对象却是活动的，可以通过查询或标识从数据存储中提取这些数据对象。如果一个持久对象引用了另一个持久对象，那么在用到这个引用时，可以透明地实例化这个引用。当事务活动时，数据对象的持久状态和数据存储是事务一致的，而且在提交事务时，对它做的修改会在数据存储中反映出来。

JDO 的优势

程序员的工具箱包含了很多可以存储和检索对象状态的工具。Java 串行化保存一个对象图并跨越时间和空间恢复它。对象状态可以写到 XML 文本流或保存到自定义文件格式。面向对象的数据库管理系统(ODBMS)和对象/关系 (object-to-relational, O/R)映射软件可用于存储数据库中的对象状态。JDBC 提供了一个与关系数据库之间的通用 API，而且 JDBC 是当前存储持久状态的主要方法。许多可替代 JDO 的产品都提供了这种随处可用、使用广泛而且得到广泛理解的优势。那么，JDO 拥有哪些优于它们的优势呢？

JDO 和串行化、XML 和自定义文件格式的比较

串行化使用起来简单，而且对应用程序代码来说，串行化机制在很大程度上是透明的。在许多情形下它是持久性机制的选择。当在两个同位体之间或者在客户和服务器之间有单对单通信时，串行化往往是把一个 JVM 对象复制到另一个 JVM 的极好选择。在需要存储或偶尔读取少量持久状态的时候，串行化也是一个合理的选择。例如，文本编辑器或 Web 浏览器可为其用户串行化配置对象。然而，当用于通用用途的持久性时，串行化存在几个缺点。串行化会立即把整个对象图装载到内存中，从而在浏览一个很大的对象图中的一小部分时速度慢且耗费内存。串行化没有事务机制，从而不支持串行化对象的并发更新。它还缺少查询机制。最后，如果图中的任何对象改变了，串行化必须存储整个对象图。

虽然 XML 和自定义文件格式都有它们的长处，但是它们和串行化都具有许多相同的缺点，并且还有其他自身的缺点。和串行化不同，XML 和自定义文件格式必需有相当多的代码把持久状态移入移出对象。另外，应用程序负责管理对象引用。如果 A 引用 B，C 引用 B，那么以文件格式该如何表示共享使用 B 呢？应用程序必须作出决定并实现它。

与串行化、XML 和自定义文件格式相反，JDO 提供了一个事务服务用于允许和调节对同一个基础持久状态的并发访问。JDO 慢速地装载对象图。当应用程序用到对持久对象的引用时，JDO 将对象装载到内存。JDO 只为那些在事务内部已经改变的对象存储持久状态。它有一个查询机制，并管理对象引用。因此，JDO 在很多方面都提供了显著的优势。

JDO 和 ODBMS、O/R 映射的比较

由于 JDO 起源于一些在面向对象数据库管理系统的开发方面领先的概念，因此 JDO 保留了和对象数据库类似的内容。但是每个供应商提供的 ODBMS 的接口和其他供应商提供的 ODBMS 的接口都有区别。尽管对象数据管理组(ODMG)指定了一个标准接口，但是这个标准还没有为供应商和应用程序开发者所广泛采用。JDO 只提供了一个服务，这个服务能够访问所有的对象数据库，一个 JDO 实现就是为这些数据库生成的。

已经有几个对象/关系映射产品为Java应用程序提供了持久性服务。遗憾的是，这些产品都使用产品指定的接口来访问持久性服务的功能。JDO提供了这样一个希望：所有O/R映射的供应商将最终支持一个通用接口。

JDO 和带 CMP 的 EJB 相比较

在 EJB 容器中，容器管理的持久性(CMP)把实体 EJB 映射到它们的持久状态。JDO 可能会替换 CMP，但不会取代 CMP。对那些寻求比现有 CMP 提供更多灵活性的应用程序开发者来说，JDO 提供了一个更简单的方法来编写使用 bean 管理的持久性的 EJB。对那些寻求更灵活的方式以在他们的 EJB 容器内实现 CMP 的供应商来说，JDO 提供了一种方式，使他们能够把多种数据存储体系结构插入到他们的 CMP 机制中。

JDO 和 JDBC 的比较

关系数据库管理系统是数据存储的主要设备，而 JDBC 是 Java 程序为访问这些数据存储而采用的主要接口。虽然 JDBC 是相当成功的，但是它还是需要相当多的代码来管理持久性。有时候程序员会编写整个持久性服务，但更经常的情况是，在类并类(class-by-class)的基础上编写代码。

程序员会发现，JDO 提供了几个优于 JDBC 的优势。首先，存储和检索对象状态需要更少的代码。JDO 完整地封装了对象构造和析构步骤。在采用了 JDBC 的代码中，这些步骤随处可见。由于程序员不再编写把持久状态移入移出对象的代码，因此需要编写的代码更少。由于 JDO 透明地提供了慢速的装载、跟踪改变和存储修改后的状态，因此应用程序数据类仍然在很大程度上忽略了持久性机制。因此，因为这些类不需要和持久性服务进行显式的交互，所以它们不会变得复杂难懂。

JDO 的另一个优势在于，JDO 减少了把对象模型映射到关系模型所需要的技巧。依靠 JDO 实现所提供的工具，有三种方法可以把持久对象的对象模型和关系模型集中到一起。或者对象模型用推导关系模型的工具构建，或者关系模型用推导对象模型的工具构建。在某种程度上，这些工具允许程序员从两个模型开始反复迭代并最终使两个模型达到一致。在这个使用 JDO 的典型开发成果中，关系模型或对象模型是驱动因素，另一个则用工具导出。

JDO 还提供了另一个优于 JDBC 的显著优势：只有在构造查询时才需要考虑对象模型。对使用 JDO 的程序员来说，不存在关系模型。同样，在查询编码期间也不使用数据库中的列名和表名。为了确定是否可以为某一组对象构造查询，只需要检查所需的对象、字段和关系是否现存于对象模型。如果对象模型中存在所需要的对象、字段和关系，那么查询就可以根据这些信息构造。因此，程序员不需要理解 SQL。用于关系数据库的 JDO 实现把 JDO 查询透明地映射到 SQL。

有人会有疑问：把程序员排除在 SQL 结构之外是否真的是个优势。为了获得更高的性能，把一些 SQL 语句融入到应用程序中去是很普通的。另外，存储过程往往用于同一目的。把 SQL 排除在程序员的领域之外会导致 JDO 在性能方面的退步吗？答案在两个方面。首先，尽管并不是 JDO 所指定的，但是提供这些特征对 JDO 实现来说是相当合理的。现在的 JDO 实现已经在提出这些特征。而且可以预期，未来的版本将会把越来越多的注意力集中在增强性能上。其次，

程序员什么都知道但什么都不精通，这是他们的天性。他们所做的工作是让应用程序运行。通常他们并不精通 SQL 编码器。这种专门技术的缺乏也是 SQL 语句首先要融入到应用程序中的部分原因。典型的应用程序很可能得益于使用那些特别关心有效地使用关系数据库的持久性软件。

与 JDBC 相比，JDO 为程序员提供了高级的持久性服务。JDO 封装了持久性的所有实现细节。程序员必须决定的只是少量的高级问题。他必须决定哪些对象类以及这些对象类的哪些特殊对象需要持久化，必须决定这些对象类中的哪些字段需要设成持久的，并在最后决定在什么地方画事务边界。这些是 JDO 没有封装的持久性的高级问题。通过暴露不可分解的持久性问题而封装所有其他问题，JDO 给予程序员提供了一个强大的、概念上的、可执行的框架。这个框架允许程序员在更短的时间内创建更健壮的应用程序。

JDO 有普遍应用的潜力

由于 JDO 提供了在封装性、一致性、透明性和可移植性等方面的优势，因此这个标准的创建者希望它能够得到普遍应用，同时也希望为 Java 平台而创建的 JavaServer Page、Java servlet、EnterPrise JavaBeans 和 JDBC 也能得到普遍应用。通过使用 JDO 而不是其他现有的替代品，应用程序开发人员在选择适合应用程序的数据存储方面有更大的灵活性，在各个相互竞争的实现中有更多的选择，在选择部署环境时有更大的灵活性，有更大的力量建立健壮的设计，而且开发速度更快。如果 JDO 像创建者期望的那样得到了普遍应用，那么所有的应用程序开发人员将更多产、更有价值，因为他们获得了使用 Java 工具箱中另一个强大工具的技能。

JDO 的简史

全球有大量的事务数据存在于关系数据库中。关系数据库使用用行列表示的数据表来建模。SQL 语言用于操作和询问这个关系模型。另一方面，面向对象的程序设计语言，如 Java，利用对象图建模。通过修改成员变量的值，在内存中完成对对象的操作。

Java 程序通常把关系数据库用作数据存储，但是困难和效率低下却是个普遍的问题。一个困难是，必须编写相当多的代码从对象向关系数据库移动数据并从关系数据库向对象移动数据。这些代码往往是重复的，并依赖于关系模型，从而引起了剪切-粘贴错误。因为程序员面对的是两个持久状态模型，一个在数据库内产生的关系模型中，另一个在应用程序的对象模型中，这导致了很大的麻烦。程序员必须决定在这两个模型之间是实现完全的映射还是只实现部分映射。第一个选择工作量大，而且只用到其中一部分。第二个选择导致模型同步和映射代码混成一团，映射代码随程序设计的进展而不断增加。不管作何种选择，为了实现在这两个模型之间的映射，程序员都必须维护这两个模型和代码。整件事特别让人困惑，因为程序员可以看出：必须有某个通用的解决办法来替代正在编写、调试和维护的代码。效率低下和错误混进映射且遍及整个映射，但是在很大程度上，这是因为典型的面向对象程序员不精通 SQL 和关系数据建模而造成的。

创建对象数据库是为了更加容易和有效地用面向对象语言操作持久数据。在 20 世纪 90 年代初期，面向对象数据库管理系统的供应商们希望通过采用与对象数据库的公共接口来增加市

场份额。基于这个期望成立了对象数据管理组(ODMG)。尽管 ODMG 规范已经修订了三次，但是到了 20 世纪 90 年代末期，对象数据库很明显没有放松掌握事务数据的关系数据库。

尽管对象数据库的成功让一些人失望，但是工程师们仍然可以看到用通用用途的持久性服务处理对象模型和关系数据库间的映射的价值。因此，在 20 世纪 90 年代期间产生了对象关系映射工具。这些 O/R 映射工具为程序员提供了一种方法来使用一个打包的多用途持久性服务，而不用再自己编写持久性服务。虽然编程小组已经在享受使用这些工具的成就，但是这些工具的应用并不广泛，它们的接口仍然是个别的和专用的。

1999 年 6 月，当 Java 标准化组织采用 JDO 的 Java 规范需求时，最初的设想，即关心从数据存储到面向对象程序的数据移动的最佳方法，已经在一端缩小，而在另一端扩展了。在面向对象语言方面，这个设想只在 Java 缩小了。在数据存储方面，这个设想扩展了，包括了关系数据库、对象数据库，并真正包括了任何类型的事务数据存储。基本的想法是：Java 程序员应该有一个事务持久性 API，市场给不同的数据存储提供了事务持久性 API 的不同实现。

随着 JDO 在专家组的发展，供应商和开发团体采用了 EnterPrise JavaBeans 规范，并把它作为在 Java 平台上建立和部署分布式对象的主要模型。对 JDO 演化的回应是，找到了能够把 JDO 应用于 EJB 组件开发的方法。EJB 开发人员将会发现，可以用 JDO 替代 JDBC。这种替代可能发生在使用 JDBC 的会话 bean 中，也可能发生在开发人员正使用 JDBC 实现 bean 管理的持久性的实体 bean 中。

在JDO Java服务需求提交超过两年半的时间以后，Java标准化组织于2002年3月发布了JDO 1.0。

JDO 1.0.1

2003 年，JDO 维护先导者发布了 JDO 1.0.1。1.0.1 版比 1.0 版有较小的改进。它澄清了某些晦涩的地方，解决了某些含糊和矛盾，并在几个地方有较小的增强。在讨论 1.0.1 规范不同于 1.0 规范的问题时，本书描述了 1.0.1 版所做的改进。在非常重要的地方，本书也注重了 1.0.1 版和 1.0 版的区别。

谁在实现 JDO 1.0

JDO 标准在世界范围内引起了关注。许多供应商都在出售以关系数据库为目标的实现。这些实现通常和任意一个有 JDBC 2.0 驱动程序的关系数据库一起使用。本书附带的开放的示例源代码展现了三家瞄准关系数据库的供应商的产品特点。这三家供应商是：SolarMetric (http://www.solarmetric.com)，其产品是 Kodo 实现；LIBeLIS (http://www.libelis.com)，其产品是 Lido 实现；Signsoft (http://www.signsoft.com)，其产品是 IntelliBO 实现。之所以选择这三家供应商，是因为他们都有 JDO 的早期版本，都定期更新其版本，而且都有免费的测试版以及很好的产品支持。

JDO 实现也开始在 ODBMS 一方出现。Poet(http://www.poet.com)正在为其 FastObjects 数据库提供 JDO 接口。Poet 和其他供应商正在支持 JDOCentral 网站(http://www.JDOCentral.com)。该网站有大量的关于 JDO 的信息和可用的实现。

作为规范过程的一部分，Sun Microsystems 用 b-tree 文件数据存储提供了一个 JDO 参考实现。Sun 还主办了非正式专家组的网页 http://access1.sun.com/jdo。

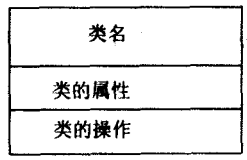
JDO 的开放的源码实现开始出现。Apache 的 Jakarta 项目(http://jakarta.apache.org/ojb)中的 ObJectRelationalBridge 是和 ODMG 3.0 兼容的对象数据库。该对象数据库正在实现 JDO 接口。另外，TriActive JDO 和 XORM 可以在 http://sourceforge.net 中找到。

担心、不确定性和疑问(FUD)

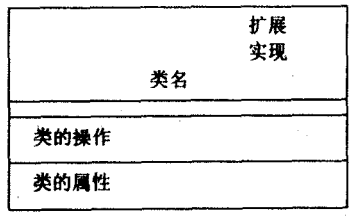
JDO 表现出了一场破坏性的革新，这场革新威胁到在某些软件产品上的已有的势力范围及其竞争的市场动态。那些发现他们已有的产品正在和完全不同的价值主张相竞争的供应商们，他们的一个标准反应就是把 JDO 这个暴发户扔到反对的目光中去。任何混乱(或 FUD)都会拖延用户向这个新选择的转移。值得赞扬的是，JDO 已经从一些已有产品的供应商那里引出了 FUD。在这些环境下，FUD 是正常的，而且最好的办法就是忽略它。

对本书中用到的 UML 图的说明

在本书中，统一建模语言(Universal Modeling Language, UML)类图被用于介绍 JDO 的各种类和接口。这些类和接口对程序员来说是非常重要的。标准的 UML 类图由三个逻辑单元组成。顶部的逻辑单元包含类名。中间的逻辑单元包含类的属性，其中包含的是类的静态变量或成员变量。底部的逻辑单元包含类操作。当用图表示接口时，顶部的逻辑单元包含字符串“<<Interface>>”，用于表明该图的主题是接口。



在本书中，为了提供更多的信息，已经修改了标准的 UML 类图。本书采用的 UML 类图有四个逻辑单元，以取代上面的三个逻辑单元。



顶部的逻辑单元和前面一样包含类名。这个逻辑单元已经扩展，用一行存放该图示类所扩展的类名(如果不是对象的话)，并且用另一行存放这个类所实现的接口。本书中的 JDO 类和接口对其中的属性不感兴趣。因此，顶部的逻辑单元还是现有的那样，只是看起来更像是两行。

第三个逻辑单元和前面一样也包含类操作。在本书中，只有公共的操作是经过检验的操作。当操作重载时就用一个注解来进行简要说明。当操作后面跟加号("+")时，将会有比上面所示的参数符号更多的参数符号。使用这个注释是为了尽可能地保持 UML 类图的简洁。

在 UML 图的底部，引入第四个逻辑单元。在这个逻辑单元中列出了和 JavaBean 相一致的属性。JavaBean 的属性是方法，这些方法通常以 get/set 对的形式进来并由实现的私有属性回送。getter 方法返回一个特定的基本类型或者它的引用，setter 方法使用一个该返回类型的参数。于是，例如方法对

```
public int getCount()
public void setCount(int num)
```

为类定义了 JavaBean 属性 Count。Boolean 型的属性会有一点不同，因为“get”方法可能是个“is”方法。例如，方法对

```
public boolean isFast()
public void setFast(boolean flag)
```

定义了 Boolean 型的属性 Fast。

由于人们已经很好地理解了 JavaBean 的属性，因此在 UML 属性逻辑单元中拥有一个数据项(而不是操作逻辑单元中的两个数据项)就更简单和更清晰。例如，在操作逻辑单元中说明 Count 和 Fast 属性需要四次操作：

```
getCount: int
setCount(num: int): void
isFast: boolean
setFast(flag: boolean): void
```

而另一方面，在属性逻辑单元中表示同样的信息只需要两个数据项：

```
Count: int
Fast: boolean
```

如果属性名和该属性返回的类型相同，那么忽略所返回的类型。例如，属性逻辑单元中的数据项“Synchronization”表明在该类中有两个方法：

```
public Synchronization getSynchronization();
public void setSynchronization(Synchronization sync);
```

并非所有的 JavaBean 属性都是读/写的，有些是只写的，还有些是只读的。这时，相应的 getter 方法或 setter 方法就不在类中给出。在本书中，为了说明一个只读属性， UML 属性实体后面跟一个减号，就像下面的属性项：“Initialized:boolean –”。只写属性在属性项的最后用加号说明。

本书结构

本书假定您将从Java标准化组织网站(http://www.jcp.org/jsr/detail/12.jsp)下载发布的JDO产品。在这个JDO产品中，您会找到JDO参考实现、JDO的Java文档以及JDO规范。JDO规范主要适用于编写JDO实现的人。本书正好相反，它打算给应用程序开发人员提供在使用JDO时所必需的信息。

本书分为两部分。第一部分是本书的主体部分，这部分试图通过从程序员的观点讨论JDO的概念和差别来补充Java文档。这部分是概念上的，而不是操作上的。它将引导您理解JDO。

本书的第二部分主要是动手实现，并分析了被称为JDO学习工具(JDO Learning Tools)的开放源程序。JDO学习工具是用JDO编写的程序。JDO学习工具中有两组程序。第一组是非典型的程序，这些程序探索JDO的行为，使用这些程序能使您更深刻理解JDO实现是如何工作的。

JDO学习工具中的第二组程序是简单但典型的应用程序，这些程序被确定为应用程序结构的典范。JDO学习工具详细考虑了三种结构：

- Swing客户/服务器应用程序
- 使用了JavaServer Pages和Java servlet的Web应用程序
- 在EnterPrise JavaBeans内使用了JDO的企业应用程序

本书的第二部分讨论每种体系结构的应用程序设计问题和编码模式问题。这个讨论利用了书中第一部分所涵盖的概念。

JDO学习工具的源代码可以从发行者的网站(http://www.apress.com)得到，也可以从SourceForge.net(http://sourceforge.net/projects/jdo-tools)上得到。本书所使用的JDO学习工具是1.0版的。

为了运行这些工具和示例，您需要下载JDO参考实现或者JDO商业实现的测试版本。第8章到第11章提供了安装测试环境的说明。JDO学习工具提供了Kodo、Lido、IntelliBO以及参考实现的构建文件。JDO学习工具将随时间而发展。为了能在将来发布更好的版本，希望您也为JDO学习工具的改进作贡献。

目　录

第 1 章　JDO 的基本概念 …… 1
1.1　JDO 持久性服务 …… 2
1.1.1　事务 …… 3
1.1.2　创建持久对象 …… 3
1.1.3　检索持久对象 …… 3
1.1.4　更新持久对象 …… 4
1.1.5　删除持久对象 …… 5
1.2　托管对象和非托管对象 …… 5
1.2.1　首要类对象和应用程序数据对象 …… 7
1.2.2　次要类对象和嵌入式对象 …… 7
1.2.3　非托管的数据对象 …… 10
1.3　数据对象的十种管理状态 …… 12
1.4　标识和惟一性要求 …… 14
1.4.1　三种类型的 JDO 标识 …… 16
1.4.2　惟一性要求 …… 22
1.4.3　把瞬态对象链接到持久状态 …… 24
1.5　小结 …… 25
第 2 章　查询 …… 26
2.1　Extent 接口 …… 26
2.1.1　Extent 和持久性管理器的 IgnoreCache 属性 …… 26
2.1.2　用于产生 Extent 对象的工厂方法 …… 27
2.1.3　只读的 CandidateClass 属性 …… 27
2.1.4　只读的 PersistenceManager 属性 …… 28
2.1.5　判断 Extent 是否包含子类 …… 28
2.1.6　在 Extent 之外获得迭代器 …… 28
2.1.7　关闭 Extent 迭代器 …… 28
2.2　JDO 查询服务的设计 …… 29
2.3　Query 接口 …… 29
2.3.1　设置查询的候选项 …… 30
2.3.2　设置查询的 Candidate 类 …… 30
2.3.3　设置查询的过滤器 …… 31

2.3.4 声明查询的参数……31
2.3.5 声明查询的变量……31
2.3.6 声明查询的输入……32
2.3.7 将查询结果排序……32
2.3.8 运行查询……32
2.3.9 关闭查询结果……33
2.3.10 编译查询……33
2.3.11 IgnoreCache 属性……33
2.3.12 只读的 PersistenceManager 属性……34
2.4 用于产生 Query 对象的工厂方法……34
2.5 JDO 查询过滤器的句法……35
2.5.1 JDOQL 的术语……35
2.5.2 JDOQL 的操作符……37
2.5.3 JDOQL 的查询方法……38
2.5.4 接口类型的一个普遍问题……39
2.6 查询变量……39
2.6.1 contains 方法特殊的语法语义……41
2.6.2 受约束变量的语义……42
2.6.3 DeMorgan 规则没有用于受约束的变量……42
2.7 排序查询结果……43
2.8 在查询中是使用缓存还是忽略缓存……44
2.9 可映射到 JDOQL 的 SQL 查询……46
2.9.1 在一个表中选择……48
2.9.2 用单对单关系在连接中选择……48
2.9.3 用单对多关系在连接中选择……48
2.9.4 用多对多关系在连接中选择……49
2.9.5 在自连接中选择……49
2.9.6 在外连接中选择……50
2.9.7 使用子查询选择……50
2.9.8 简单性是 JDOQL 的力量所在……50
2.10 在 JDOQL 不能满足需要时使用 SQL……51
2.11 JDO 和 JDOQL 是如何有助于开发过程的……51
2.12 小结……52

第 3 章 持久性管理器……53
3.1 方法参数中的 null 值处理……54
3.2 获得和关闭持久性管理器……54
3.2.1 关闭持久性管理器……54

3.2.2 只读的 Closed 属性 ····· 55
3.2.3 只读的 PersistenceManagerFactory 属性 ····· 55
3.3 数据对象的 JDO 管理的控制 ····· 55
3.3.1 xxxAll 操作的行为 ····· 55
3.3.2 设置和删除持久对象 ····· 56
3.3.3 向事务添加对象和从事务中移除对象 ····· 58
3.3.4 从 JDO 的管理中移除持久对象 ····· 61
3.3.5 能替代调用 makeTransient 方法的方法 ····· 62
3.4 控制缓存 ····· 63
3.4.1 检索持久状态 ····· 63
3.4.2 驱逐持久状态 ····· 65
3.4.3 刷新持久状态 ····· 66
3.5 获得和构造标识对象 ····· 67
3.6 根据标识提取应用程序数据对象 ····· 68
3.7 持久性管理器中的工厂方法 ····· 69
3.8 获得持久性管理器的事务 ····· 69
3.9 持久性管理器的属性 ····· 69
3.9.1 IgnoreCache 属性 ····· 69
3.9.2 Multithreaded 属性 ····· 70
3.9.3 UserObject 属性 ····· 70
3.10 小结 ····· 70
第 4 章 事务和缓存 ····· 71
4.1 事务 ····· 71
4.1.1 JDO 中的乐观事务和数据存储事务 ····· 72
4.1.2 隔离等级和事务间的交互 ····· 73
4.1.3 JDO 实现定义其事务的隔离等级 ····· 74
4.2 JDO 的 Transaction 接口 ····· 74
4.2.1 控制事务边界 ····· 75
4.2.2 只读的 PersistenceManager 属性 ····· 76
4.2.3 5 个事务属性 ····· 76
4.2.4 与事务的完成同步 ····· 82
4.3 事务属性是如何控制状态转换的 ····· 84
4.3.1 事务之外的 JDO 状态转换 ····· 84
4.3.2 事务内的 JDO 状态转换 ····· 85
4.3.3 数据存储事务独特的 JDO 状态转换 ····· 86
4.3.4 乐观事务独特的 JDO 状态转换 ····· 87
4.3.5 当 RetainValues 为 false 时 JDO 的状态转换 ····· 88

4.3.6 当 RetainValues 为 true 时 JDO 的状态转换 ····· 88
4.3.7 当 RestoreValues 为 false 时 JDO 的状态转换 ····· 89
4.3.8 当强 RestoreValues 为 true 时 JDO 的状态转换 ····· 90
4.3.9 当弱 RestoreValues 为 true 时的 JDO 状态转换 ····· 90
4.4 可选的瞬态事务特征 ····· 91
4.5 JCA、JTA 和 JDO 事务 ····· 92
4.6 持久对象及其状态的 JDO 缓存 ····· 93
4.6.1 从缓存中移除持久对象 ····· 94
4.6.2 持久性管理器的缓存何时可以提高性能 ····· 94
4.6.3 控制持久性管理器的缓存 ····· 94
4.6.4 JDO 实现的二级缓存 ····· 95
4.7 小结 ····· 96

第 5 章 增强类和托管字段 ····· 97
5.1 应当增强哪些应用程序类 ····· 97
5.2 托管字段和非托管字段 ····· 98
5.2.1 哪些类型的字段可以是非托管字段 ····· 98
5.2.2 哪些类型的字段可以是事务字段 ····· 98
5.2.3 哪些类型的字段可以是持久字段 ····· 99
5.3 增强 ····· 100
5.3.1 预期的增强效果 ····· 101
5.3.2 增强的副作用和局限性 ····· 102
5.4 JDO 元数据 ····· 113
5.4.1 JDO 元数据文件的名字和位置 ····· 113
5.4.2 JDO 元数据的结构 ····· 114
5.5 小结 ····· 124

第 6 章 产生持久性管理器的工厂 ····· 125
6.1 PersistenceManagerFactory 接口 ····· 125
6.2 获得持久性管理器 ····· 126
6.2.1 从持久性管理器工厂获得持久性管理器 ····· 126
6.2.2 从连接工厂获得持久性管理器 ····· 126
6.3 获得持久性管理器工厂 ····· 127
6.3.1 JDOHelper 中的 getPersistenceManagerFactory 方法 ····· 127
6.3.2 从 JNDI 获得持久性管理器工厂 ····· 130
6.3.3 构造持久性管理器工厂 ····· 131
6.4 关闭持久性管理器工厂 ····· 131
6.5 获得连接工厂 ····· 132
6.6 分析 JDO 实现 ····· 132

6.6.1 确定实现对可选特征的支持……132
6.6.2 从实现获得特定供应商的信息……134
6.7 配置持久性管理器工厂……134
6.8 配置到数据存储的连接……134
6.9 JDO 对容器管理事务的支持……135
6.10 JDO 对 bean 管理事务的支持……136
6.11 在 CMT 会话 bean 中使用 JDO……136
6.12 从业务方法返回已串行化的数据对象……142
6.13 在带 BMP 的实体 bean 中使用 JDO……142
6.13.1 EJB 的实体环境回调方法……146
6.13.2 QuoteServer bean 的 setup 方法和 cleanup 方法……146
6.13.3 ejbStore 回调方法……147
6.13.4 调用 ejbStore 时，谁负责把修改刷新到数据存储……147
6.13.5 实体 EJB 的业务方法……148
6.13.6 ejbLoad 回调方法……148
6.13.7 ejbPassivate 回调方法……149
6.13.8 ejbActivate 回调方法……149
6.13.9 ejbCreate 方法……149
6.13.10 ejbRemove 方法……149
6.13.11 ejbFindByPrimaryKey 方法……149
6.13.12 EJB 查找器方法……150
6.13.13 在业务方法的响应中串行化数据对象……150
6.13.14 转换到一个连接工厂……150
6.14 在 BMT 会话 bean 中使用 JDO……150
6.14.1 无状态 BMT 会话 bean……151
6.14.2 有状态 BMT 会话 bean 必须作出的改变……153
6.15 在消息驱动 bean 中使用 JDO……154
6.16 小结……155
第 7 章 JDOHelper、回调和异常……156
7.1 JDOHelper 实用类……156
7.1.1 JDOHelper 中的各种实用方法……156
7.1.2 确定数据对象的管理状态……159
7.2 InstanceCallbacks 接口……161
7.2.1 jdoPostLoad 回调……161
7.2.2 jdoPreStore 回调……162
7.2.3 jdoPreClear 回调……162
7.2.4 jdoPreDelete 回调……163