信息科学与技术丛书 程序设计系列

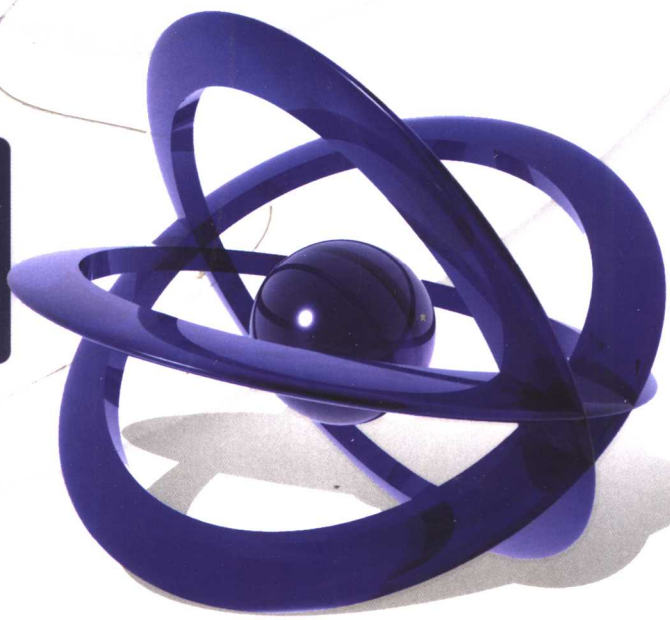
Hibernate

完全手册

侯志松 余周 郑焕 等编著

随书配套源程序
下载网址 www.cmpbook.com

- Hibernate 的基本配置和使用方法
- 使用 Hibernate 实现对象关系的映射
- 使用面向对象方式查询数据
- 用 Hibernate 构建企业级应用程序开发框架



机械工业出版社
CHINA MACHINE PRESS

信息科学与技术丛书·程序设计系列

Hibernate 完全手册

侯志松　余 周　郑 焕　等编著

机 械 工 业 出 版 社

本书从应用程序开发的角度出发，在介绍关系数据库系统知识和面向对象方法的基础上，全面介绍对象关系映射领域的工业标准——Hibernate的基本功能、体系架构、使用方法及高级特性。全书以对象关系映射为主线，分成3个模块：第1～6章介绍对象持久化的基础以及Hibernate的基本配置和使用方法；第7～11章详细介绍如何使用Hibernate实现对象关系的映射，并讨论了使用面向对象方式查询数据的方法和技巧；第12～16章分析了Hibernate中事务、并发、缓存等高级特性，并讨论了如何整合Hibernate和成熟技术，构建企业级应用程序开发框架的技术实践。书中源代码可免费下载。

本书结构严谨、条理清晰、实例丰富、理论详尽，适合软件开发专业人士以及计算机专业、软件工程专业的高校师生阅读，也可作为数据库课程和面向对象设计课程的参考教材。

图书在版编目（CIP）数据

Hibernate完全手册/侯志松等编著．—北京：机械工业出版社，2008.4
（信息科学与技术丛书·程序设计系列）
ISBN 978-7-111-23764-8

Ⅰ．H…　Ⅱ．侯…　Ⅲ．JAVA语言－程序设计　Ⅳ．TP312

中国版本图书馆CIP数据核字（2008）第037172号

机械工业出版社（北京市百万庄大街22号　邮政编码100037）
责任编辑：车　忱
责任印制：邓　博

北京双青印刷厂印刷
2008年4月第1版·第1次印刷
184mm×260mm·28.75印张·708千字
0001—5000册
标准书号：ISBN 978-7-111-23764-8
定价：46.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
销售服务热线电话（010）68326294
购书热线电话：（010）88379639　88379641　88379643
编辑热线电话：（010）88379753　88379739
封面无防伪标均为盗版

出版说明

随着信息科学与技术的迅速发展，人类每时每刻都会面对层出不穷的新技术、新概念。毫无疑问，在节奏越来越快的工作和生活中，人们需要通过阅读和学习大量信息丰富、具备实践指导意义的图书，来获取新知识和新技能，从而不断提高自身素质，紧跟信息化时代发展的步伐。

众所周知，在计算机硬件方面，高性价比的解决方案和新型技术的应用一直备受青睐；在软件技术方面，随着计算机软件的规模和复杂性与日俱增，软件技术受到不断挑战，人们一直在为寻求更先进的软件技术而奋斗不止。目前，计算机在社会生活中日益普及，随着因特网延伸到人类世界的层层面面，掌握计算机网络技术和理论已成为大众的文化需求。由于信息科学与技术在电工、电子、通信、工业控制、智能建筑、工业产品设计与制造等专业领域中已经得到充分、广泛的应用，所以这些专业领域中的研究人员和工程技术人员越来越迫切需要汲取自身领域信息化所带来的新理念和新方法。

针对人们对了解和掌握新知识、新技能的热切期待，以及由此促成的人们对语言简洁、内容充实、融合实践经验的图书迫切需要的现状，机械工业出版社适时推出了“信息科学与技术丛书”。这套丛书涉及计算机软件、硬件、网络、工程应用等内容，注重理论与实践相结合，内容实用，层次分明，语言流畅，是信息科学与技术领域专业人员不可或缺的图书。

现今，信息科学与技术的发展可谓一日千里，机械工业出版社欢迎从事信息技术方面工作的科研人员、工程技术人员积极参与我们的工作，为推进我国的信息化建设作出贡献。

机械工业出版社

前　言

随着社会的发展和进步，信息资源逐步成为企业中最重要的财富和资源。目前，大部分企业都使用关系型数据库来建立自己的信息系统。而在企业级应用软件开发领域，人们往往借助面向对象的技术来简化软件设计人员和开发人员的工作，提升软件开发的效率。当企业级应用软件处理信息后，必须将有用的信息存储到企业的信息系统中，也就是关系数据库中，以便下次使用。由于关系数据库的关系模型和对象模型的实现机制不同，两者之间必然存在匹配问题。为了解决这个问题，人们常采用对象关系映射机制实现对象模型和关系模型的自动转化。

在 Java 社区，人们就对象关系映射机制展开了空前激烈的讨论，进行了大量的探索和实践，也出现了很多解决方案。其中 Hibernate 逐渐崭露头角，成为对象关系映射领域中事实上的工业标准。Hibernate 提供了一个完整的面向对象数据存取解决方案。使用 Hibernate，可以解决企业级应用开发中的大部分问题。Hibernate 采用简单的传统 Java 对象（Plain Old Java Object，POJO）编程模型，使用透明的对象关系映射机制，屏蔽了底层关系数据库复杂的技术细节，降低了编写应用程序的复杂度，提高了编写应用程序的效率。本书主要关注如何使用 Hibernate 实现面向对象的关系数据库存取工作。

本书在讲解关系数据库理论和面向对象设计理论的基础上，以对象关系映射为主线，分 3 个模块，16 章，讲述了 Hibernate 最新版本的应用特性。第 1 章介绍了当前软件开发领域的对象持久化需求和现有的对象持久化技术；第 2 章讲述了一个简单完整的 Hibernate 应用，是 Hibernate 的入门介绍；第 3 章深入分析了 Hibernate 在不同应用环境下采用的体系架构；第 4 章讨论了 Hibenrate 如何使用对象标识符将对象和数据库记录一一对应起来；第 5 章列举了 Hibernate 应用的多种运行时配置和配置参数；第 6 章详细讲解了如何使用 Hibernate 映射类型将 Java 数据类型和 SQL 数据类型对应起来；第 7 章介绍对象关联关系的映射和对象层次结构的映射；第 8 章深入分析了 Hibernate 中对象的状态，并讲解如何使用面向对象的方式操作关系数据库；第 9 章详细讨论了 Hibernate 查询语言；第 10 章讲解条件查询的应用程序接口；第 11 章介绍如何在 Hibernate 中使用原生的 SQL 语句；第 12 章深入分析 Hibernate 中实现事务和并发的机制和策略；第 13 章详细讨论了如何使用 Hibernate 缓存提高应用程序的性能；第 14 章讲解 Hibenate 提供的批量处理、延迟加载等高级特性；第 15 章讨论使用 JPA 的对象持久化实现；第 16 章讨论了如何结合现今 Java 社区流行的开源技术，构建企业级应用开发框架的技术实践。本书每一章都自成一体，是一个独立的学习单元，读者可以根据自身情况选择学习。同时，本书采用统一的应用实例，内容前后呼应，难度由浅入深，是一个不可分割的整体。另外，本书在讲解实例的同时，穿插基础理论的讲解，能够让读者在知其然的同时知其所以然，加深读者的理解，提升读者的应用技能。

本书主要由侯志松、余周、郑焕编著，参与编写的人还有：曲培新、王鹏飞、杨中科、王伟娜、华莹、杨献峰、毛新华。在编写过程中，我们参考了很多优秀的教材、论文和开源技术框架，在此对所有引用文献、代码的作者或组织表示衷心的感谢；同时要感谢屈辰晨先生等在本书编写过程中的大力帮助。

书中所有实例的源代码可以在 http://www.cmpbook.com 下载。

本书适合软件开发专业人士以及计算机专业、软件工程专业的高校师生阅读，也可作为数据库课程和面向对象设计课程的参考教材。

本书是我们几年来软件开发经验的总结，但由于作者水平有限，加之时间仓促，书中的错误和欠妥之处在所难免。特别是有些术语，翻译不一定准确，恳请各位读者和同行批评指正，以便我们今后改进。

作　者

2007 年 10 月

目　录

第1章 对象持久化基础

当今，信息资源已成为企业重要的知识资本和资源，传统的手工操作和文件式管理已经不能满足企业快速发展的需求。因此，作为企业信息系统核心和基础的数据库技术得到了越来越广泛的应用。使用数据库技术，企业可以保存既有的信息资源，并能够快速获取信息，提供基于数据的定量或定性分析，清晰地展现企业业务运行状况。因而，在很大程度上，企业使用数据库技术的水平也代表了一个企业的信息化水平。

在企业级软件开发中，应用程序不可避免地需要和数据库交互，存取业务系统的信息资源。当前，在企业数据运算环境中，大多数应用都是使用关系型数据库来保存数据。而在软件开发领域，面向对象的分析和设计占据了主导地位。因此，在开发企业级应用系统时，需要综合考虑关系型数据库和面向对象模型之间的匹配问题。

在面向对象领域，尤其是在 Java 领域，为了解决关系模型和对象模型之间的差异，人们进行了大量的探索和实践。从 Java 提供的 JDBC (Java Database Connectivity)技术到现今流行的对象关系映射(Object Relational Mapping,ORM)技术，都在不同程度上解决了对象的持久化问题。其中以 Hibernate 为代表的对象关系映射框架逐渐成为 Java 领域数据持久化技术的主流。

本章主要内容：

- 数据持久化在企业级应用中的地位
- 层次软件体系结构的特性
- 正在发展中的对象持久化技术
- 对象关系映射概念

1.1 企业级应用中的数据持久化需求

在社会高速发展的今天，信息资源已经成为所有企业的重要财富和资源。建立一个能够满足企业内部或企业之间信息需求的、具备高可用性的信息系统，成了一个企业赖以生存和发展的条件。在企业级运算环境中，人们往往借助计算机技术来完成数据的处理和管理工作。数据的处理是指对各种数据进行收集、存储、加工和传播的一系列活动的总和。而数据管理则是指对数据进行分类、组织、编码、存储、检索和维护的过程。数据管理是数据处理的中心问题。

对于一个企业或组织而言，完整、有效的数据信息是企业和组织良好运行的重要保障。企业和组织中的数据千差万别，其中有些数据使用过后就可以丢弃，而有些数据必须持久地保存，以便在需要的时候重新获取。比如：客户关系管理(Customer Relationship Management，CRM)系统中的用户信息、企业资源计划(Enterprise Resource Planning，ERP)中的产品、物料信息等等。这些持久化信息往往是企业信息系统运行的基础数据或业务数

据，用来反映企业的业务范围和运行状况，为企业的决策者提供可靠的数据依据。

企业中的数据经过收集、分类、组织等阶段后，必须以某种形式完整地保存下来，以便信息的传播和检索。此类信息是企业的数据资产，它不仅仅是企业业务运行的基层保证，还是企业保持长盛不衰的知识资本。持久数据的积累过程也是企业知识积累的过程，对持久数据进行客观的统计，可以使企业决策者在决策、选择战略和执行战术时，更客观、更审慎、更准确地评估得与失，清晰地认识自身的业务边界和差距，准确地把握现有市场，并不失时机地开拓新市场。

现今的大部分企业都采用了计算机管理的数据平台。虽然计算机研制的初衷是为了实现复杂的科学计算，但是自从计算机诞生的那天起，人们就开始利用它进行数据处理和管理工作。

在数据处理和数据管理需求的驱动下，在计算机硬件和软件的发展基础上，数据的管理技术经历了人工管理、文件管理和数据库管理三个阶段。其中每一个阶段都是前一个阶段的质的飞跃。在前两个阶段，数据的处理和管理难度较大，还不能适应企业级应用的需求。随着数据管理规模的扩大，计算机技术在各行各业的应用也越来越广泛，随之而来的数据量也开始呈指数级增长。在数据处理方面，联机实时处理和分布式应用也是现今社会必不可少的应用需求。数据库技术正是为了解决这一问题而出现的。在现今的企业和组织中，数据库技术已经是企业和组织中实现数据处理和管理的关键技术。

数据库技术的发展适应了企业对于数据的需求，因此，数据库技术作为企业信息系统核心和基础的应用越来越广泛。从小型单项的事务处理系统到大型的信息系统，从联机事务处理到联机事务分析处理，从一般的企业管理到计算机辅助设计和制造再到计算机集成制造系统，从办公信息系统到地理信息系统，越来越多的领域都采用了数据库技术来存储信息资源。这也标志着企业的数据信息变得越来越重要，管理也越来越复杂，从侧面反映了企业中数据持久化的重要性。

由此可以看出：数据库技术已经成为企业信息平台的核心，用来完成企业数据的存储工作。应用程序必须将业务相关的数据通过一定的方式持久化到数据库，作为企业的数据资产保存下来。另一方面，应用程序还应该能够从数据库获取已有的数据，以适当的形式提供给客户或支撑系统的运行。如何高效地存取数据、简化编程模型、降低应用的复杂度，正是本书所要讨论的课题。

1.2 软件层次体系结构

软件体系结构的一个核心问题是能否重复使用体系结构模式，即能否达到体系结构级的软件重用。也就是说：能否在不同的软件系统中，采用同一体系结构。软件体系结构风格反映了同一领域中众多系统所共有的结构和语义特性，用来指导如何将各个模块和子系统有效地组织成一个完整的系统。采用软件体系结构来描述软件系统，可以促进设计的重用，使用原有的、经过实践的、可靠的解决方案来解决新的问题，在不同的应用中复用有效的实现组件。另外，这也方便在不同的设计者之间传递知识。

软件体系结构关注系统级别的架构体系，为大粒度的软件重用提供了可能。然而，对于应用体系架构来说，由于视点和角度的不同，系统架构师有很大的选择余地。要为系统选择

或设计体系架构，必须根据特定项目的具体特点，进行分析、比较后再行确定。经典的体系结构有管道和过滤器、数据抽象、基于事件的调用等，它们都在一定的程度上解决了软件重用问题，便于实现高内聚、低耦合的系统。

随着企业级应用规模的扩大，软件的复杂度不断提高，为了合理地划分应用，提出了层次架构的解决方案。层次系统组织成一个层次架构，每一层只对上层负责，并作为下层的客户。层次中除了一些精心设计的输出方法外，内部系统只对本层开放，实现高内聚的组件架构。使用层次架构，可以在层次之间增加或抽取抽象层，将复杂的问题简化为一个简单的增量序列实现。同时一个层的修改最多只会影响到相邻的两层，而且在层的接口不变的情况下，可以任意修改本层的实现。层次架构合理地分割了应用的逻辑层次，简化了应用设计，最大限度地支持重用，便于系统的扩展。

经典的层次体系结构包括两层结构和三层结构。其中两层结构是基于集中式计算结构的资源不对称，为实现共享而提出的。两层结构体系分解应用，将数据展示的处理工作交给客户端，业务的处理和数据管理则由服务端负责。两层结构体系可以提供比较强大的数据处理能力，但其开发、部署较为困难，安全问题较为突出，维护较为困难。针对这些问题，提出了三层体系结构，即在两层体系结构基础上，引入应用服务器概念，将应用的逻辑全部放置在应用服务器上。

三层体系结构将应用的逻辑分为表现层、业务层和数据层三个部分，如图 1-1 所示。

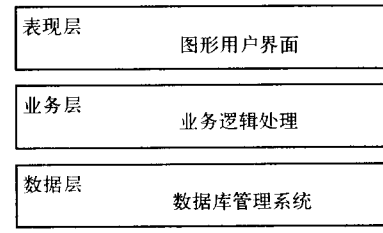


图 1-1　三层体系结构

三层体系结构的层次功能如下：

- 表现层：表现层是应用的用户接口部分，负责用户和应用之间的交互。表现层将检查用户的数据输入，并在处理后向用户展现处理结果。该层常采用图形用户界面，方便了用户的操作。实现表示层时可采用合适的技术进行开发，并可以随时更换实现而不影响其他两层。表现层承担了数据的展现逻辑，但不负责任何与业务有关的处理逻辑。
- 业务层：业务层也叫服务层，是应用业务逻辑处理的核心。该层负责从表现层接受用户的输入，按照规定的业务逻辑和业务规则进行处理，并从数据层获取数据或向数据层提交数据。业务处理结束后，向表现层提交处理结果。
- 数据层：该层负责数据的持久化，为应用提供一个统一、安全的数据持久机制。该层一般为数据库管理系统。

层次体系结构允许合理地划分应用的功能，使之保持逻辑上的相对独立性，从而使整个应用系统逻辑结构保持清晰、统一，提高系统的可扩展性和可维护性。其次，由于层次系统将功能划分为不同的层次，因此可以在各层并行开发，提高软件交付的能力。另外，层次结构可以灵活地选择实现平台，在接口不变的情况下，改变层次的内部实现或对某个层次进行再次划分。

使用层次结构体系也会给软件开发带来相应的问题。首先，划分层次时需要平衡各层的功能，同时保持层次之间高效的交互，如果层次功能划分失衡，通信不畅，将直接影响系统的性能，因为系统最薄弱的环节往往决定了系统的整体性能。其次，使用层次结构进行软件

开发时，项目管理的难度也会随之增加。项目管理人员必须在开发团队之间维护一个畅通的信息沟通渠道。另外，层次划分将增大系统集成和调试的难度。因此，选择使用层次结构体系时，需要综合考察应用的业务，构建一个适合业务的系统体系结构。

在现今的企业级软件开发中，业务层集中了企业的业务逻辑处理和业务规则处理。对于大型的企业级应用，如果在业务层集成数据的处理逻辑，那么业务层将变得臃肿不堪。同时，业务逻辑或业务规则中包含数据处理逻辑，代码逻辑混乱，不仅折损系统的性能，还会给应用的开发和维护工作带来很大的困难。因此，在企业级架构中，常把数据访问逻辑从业务层分离出来，作为单独的层次进行开发、管理，即将业务层分解成为独立的两个层次：业务逻辑层和持久层。如图 1-2所示。

在系统中引入持久层，负责所有相关数据的持久化操作，可以为整个应用系统提供一个高层、统一、安全、并发的数据持久机制。将数据逻辑从业务层抽取出来，使得业务逻辑层可以专注于业务逻辑和业务规则的处理，降低了层次之间的耦合度，增加了业务的伸缩性和灵活性。

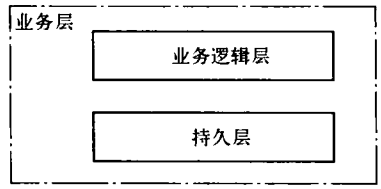


图 1-2 业务层分解

持久层屏蔽底层数据层的实现细节，为业务逻辑层提供透明的数据访问接口，消除了业务逻辑和数据逻辑之间的耦合，使得程序开发人员只需关注业务逻辑的实现，提高应用开发的速度，降低项目的风险。

在持久层，可以采用多种技术框架实现。本书主要关注对象关系映射框架中的开源代表 Hibernate。

1.3 对象持久化技术概览

在软件开发过程中，人们常借助一定的软件模型来描述现实世界，对现实世界进行抽象，完成问题域到解域的对应。域对象模型(Domain Object Model)是现实世界实体以及其业务上下文关系的抽象。借助域模型，架构师可以捕捉主题专家或域专家的系统观点，表达现实世界真实的业务运行状况。域模型不是针对系统实现的，其中不包含任何实现细节。在业务需求的基础上，域模型可以对应一个或多个对象模型，这些对象模型可以由程序设计语言来描述，获取问题域的解。使用程序设计语言实现系统时，域模型中的域对象需要持久化保存，以支撑系统业务的运行。

在面向对象系统中，持久化可以理解为：某个对象的生命周期不依赖于程序的执行与否，这个对象以某种形式持久地存活在系统中，可以随时被获取。持久化的主要宗旨是在稳定的存储介质中保存业务数据，以便在需要的时候重复利用这些数据。

在对象持久化领域，人们从来都没有停止探索，一些中立的软件供应商曾试图解决持久化问题并制定了相应的技术标准，比如 Project Forest 项目和对象数据管理组(Object Data Management Group,ODMG)。迄今为止，在 Java 领域中，涌现了很多的技术方案试图解决持久化问题。这些技术方案虽然都可以解决一定的问题，但也存在一定的局限性。

1.3.1　Java 对象序列化

序列化是 Java 语言中内置的轻量级数据持久化机制。序列化机制可以将任何实现了 java.io.Serializable 接口的对象转化为连续的字节流数据，保存在文件中，或者通过网络进行传输。这些数据日后可被还原为原先的对象状态。

当对象被序列化时，Java 将遍历对象图的闭包，将所有可访问的对象写入数据流中，如果类成员标记为 transient，则忽略该成员。当对象重新读取时，Java 将建立等价的对象图，并初始化 transient 成员为默认值。使用 Java 内置序列化机制将数据持久化的形式如例程 1.1 所示。

【例程 1.1】

```
public class SerializableTest extends TestCase {
    public void testSerializable() throws FileNotFoundException, IOException,
            ClassNotFoundException {

        //实现了 java.io.Serializable 接口的对象
        Product p = new Product(new Long(1), "Hibernate");

        //序列化对象
        ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(
                "product.out"));
        oos.writeObject(p);
        oos.close();

        //从文件中反序列化对象
        ObjectInputStream ois = new ObjectInputStream(new FileInputStream(
                "product.out"));
        Product pi = (Product) ois.readObject();
        assertEquals(1, pi.getId().intValue());
        assertEquals("Hibernate", pi.getName());
        ois.close();
    }
}

//持久类,实现 java.io.Serializable 接口
class Product implements java.io.Serializable {
    //类属性
    private Long id;
    private String name;
    //构造方法
    public Product(Long id, String name) {
```