碁峰
www.gotop.com.tw

妙解

树头孤鸟 著

Hibernate 3.x

——叩响面向对象思想之门

张若飞 改编
飞思科技产品研发中心 监制

附书光盘包含本书工程项目源文件、依赖包及部署文件

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

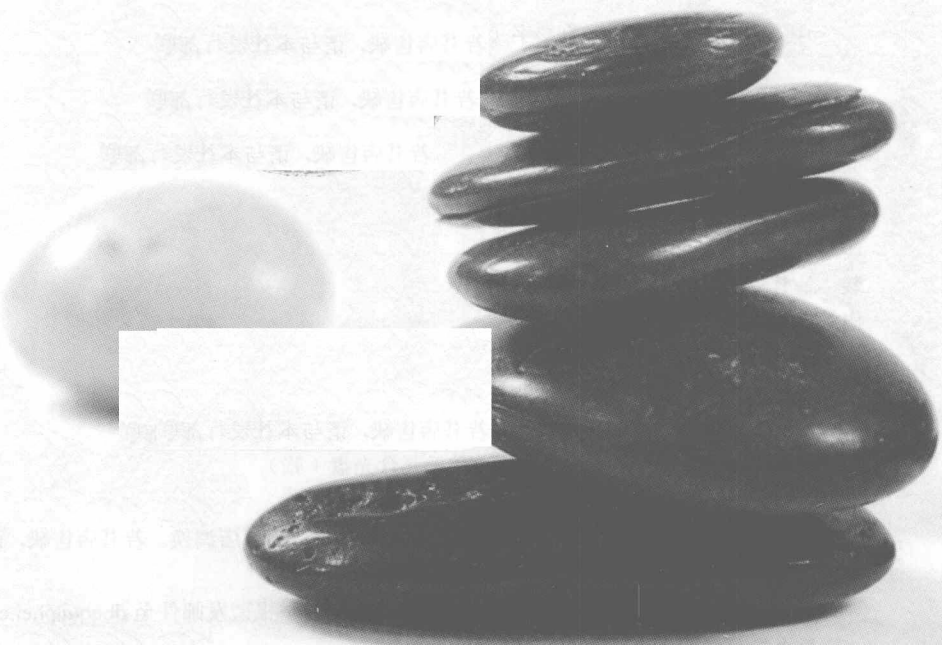
妙解

树头孤鸟 著

Hibernate 3.x

——叩响面向对象思想之门

张若飞 改编
飞思科技产品研发中心 监制



電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

内容简介

本书以通俗化的文笔，以数据库、面向对象思维、架构、Hibernate API、Hibernate 运作原理等作为切入点，让您心法、功法兼修，快快乐乐、扎扎实实学会运用 Hibernate 的技巧。本书还分别从静态结构面和动态行为面两个方面出发，为您解答如何实现细粒度（Fine-Grained）的面向对象设计！

随书所附光盘包含实例源文件。

本书从软件设计模式的角度解读 Hibernate，思路新颖，语言轻松，风格鲜明，是不可多得的 Hibernate 参考书。本书适合 Java 程序员参考学习，也可作为相关培训机构的参考教材，还可以作为高等院校相关专业师生的参考书。

本书为经台湾碁峰信息股份有限公司独家授权发行的中文简体版。本书中文简体字版在中国大陆之专有出版权属电子工业出版社所有。在没有得到本书原版出版者和本书出版者书面许可时，任何单位和个人不得擅自摘抄、复制本书的部分或全部以任何方式（包括数据和出版物）进行传播。本书原版版权属碁峰信息股份有限公司。版权所有，侵权必究。

版权贸易合同登记号 图字：01-2009-7746

图书在版编目（CIP）数据

妙解 Hibernate 3.x：叩响面向对象思想之门 / 树头孤鸟著；张若飞改编.—北京：电子工业出版社，2010.2
ISBN 978-7-121-10052-9

I. 妙… II.①树…②张… III.Java 语言—程序设计 IV.TP312

中国版本图书馆 CIP 数据核字（2009）第 225530 号

责任编辑：杨　鸽　赵树刚
印　　刷：北京智力达印刷有限公司
装　　订：三河市鹏成印业有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：787×980　1/16　印张：24　字数：614.4 千字
印　　次：2010 年 2 月第 1 次印刷
印　　数：4 000 册　　定价：49.00 元（含光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

代序

很高兴能看到一本从软件设计的角度，用深入浅出的方式带领读者一窥 Hibernate 奥秘的书诞生。回想 JBoss 在中国台湾已扎根了两年多，虽然使用 JBoss Hibernate 的人很多，但是一般开发者大约只用到 Hibernate 三分之一的功能而已，并且有很多细节只知其然，而不知其所以然。本书对于新手来说是最好的入门书籍，它可以让你从应用与设计的角度来了解如何使用 Hibernate。即使对于高水平的人来说，书中仍有很多细节值得去深入探索，再加上作者以不同形式的笔触来表现自己的思维，相信你也可以通过本书被他的热情所感染，并一同见证 Open Source 在中国台湾的发展。

Red Hat JBoss 中国台湾区资深技术顾问——Simon Chen 写于中国台湾

本书阅读时的注意事项

书中有些术语直接采用原文，尤其是因直译可能会产生误会的术语，基本上保持原文不译（例如 Hibernate Component 译为组件并不恰当）。

程序在 Sample 光盘中的位置（path）如下图所示。

AboutOracle（文件名称）	Item.hbm.xml（类或者配置文件）
_7._7222（包名）	（method 的名称，如果有的话）

```
<id  name="id"
column="ITEM_ID"
type="java.lang.Long"
>

<generator class="sequence">
       <param name="sequence">SEQ_ITEM_ID</param>
   </generator>

</id>
```

此为试读，需要完整PDF请访问：www.ertongbook.com

目录

3 把软件做软——浅谈面向对象思想、分析模式/设计模式、域模型

4 开启对象和关系型数据库的一扇门——静态结构的解决之道

5 开启对象和关系型数据库的一扇门——动态行为的解决之道

6 事务！事务！！事务

7 更有效地使用 Hibernate

8 Hibernate 的两把查询利器——HQL & Criteria API

9 查漏补缺——更多的 Hibernate 应用

1

初探 Hibernate

本章涵盖以下内容：

- 1.1 Hibernate？据说……
- 1.2 从简化 JDBC 实际操作的角度来看 Hibernate
- 1.3 从体现“面向对象思想”的角度来看 Hibernate
- 1.4 主线任务与支线任务

1.1 Hibernate？据说……

Hibernate！究竟是何方神圣？

据说，连续赢得 2003、2004 Jolt 大奖的 Hibernate 被 Java 社区公认为持久层（Persistence Layer）框架（Framework）的第一把交易。是的，在逐鹿中原、群雄四起（JDBC、EJB2.X、JDO、iBatis、TopLink……）的 Java 国度里，快速崛起的 Hibernate 以其优越的战斗力，确实打出了一片江山；事实上，Hibernate 已经俨然成为事实上的标准，并进而深远地影响了 Sun 的官方技术——EJB3/JPA（Java Persistence API）的发展及制定。

据说，在克服一些学习曲线后，使用 Hibernate API 可以极大简化使用“纯”JDBC 编写的烦琐代码。相信曾经历过 Hibernate 洗礼的程序设计人员，大多都能够认同：Hibernate 确实能在一定程被上提高生产力！（我们将在下一节，略微比较二者的写法。）

据说，在当年 OO（面向对象）登上舞台之时，伴随而来了 UML（Unified Modeling Language）、方法论如 UP（Unified Process）、Analysis Patterns（分析模式）、Design Pattern（设计模式）等技术。但是，当人们试图通过这些传说中的银弹，来打造更美好的软件时，无奈的是，一旦对象遇到了数据库，原来脑中 OO 设计中的很大一部分，都难以在现实中实现。于是，课本上的 OO 设计思想几乎成了海市蜃楼、梦幻泡影。以业务系统为例，业务逻辑层（Business Layer）中最为核心的域对象（这里指的域对象不是只有 getXXX 和 setXXX 的“数据”对象[1]），一旦遇到数据库就无所对应了。毕竟，**域对象（Domain Object）和数据库中的表（Table）不全是一一对应的**。逻辑上的 O/R Mapping（即域对象与数据库表之间的对应）还有规律可循，但是若考虑到框架的完整性，实际上的转换（将表中的数据取出，转换为内存中的对象，或者将内存中的对象存储到数据库中）并非易事。而 Hibernate 正是 ORM 机制（O/R Mapping Mechanism）中的翘楚！

1.2 从简化 JDBC 实际操作的角度来看 Hibernate

JDBC 已经出现多年，是 Java 中数据库统一进行存取的规范 API。尽管 JDBC 可谓劳苦功高，但是仍有改进的空间：

- JDBC 定义了数据库的存取标准，但是每种数据库的 SQL 语句仍有所不同，JDBC 也并未统一 SQL 语句的写法；如果要实现一套可适用于不同数据库的软件，使用 JDBC 恐怕难以逃脱编写多套 SQL 语句的厄运。
- 我们在上一节简单提到了业务逻辑层体现面向对象思想的难处（我们将在下节进一步进行探讨），的确，JDBC 从数据库中读取的是结果集（ResultSet），而非域对象。但是我

1 指我们通常看到的，只有存取属性方法（Method）的值对象（Value Object）。

此为试读，需要完整PDF请访问：www.ertongbook.com

们不应该责怪 JDBC，因为它本来就是针对于数据库方面的处理，其本质上就不属于 ORM（O/R Mapping Mechanism）框架!

- 使用纯 JDBC 编写的代码十分烦琐。接下来，我们将会看到同为添加数据的功能，分别用 JDBC、Hibernate，以及 Hibernate 搭配 Spring 来实现各有什么不同（请先关注于代码的行数，暂时不要太注意程序的内容）。

比较 JDBC/Hibernate/Hibernate+Spring 的实际操作:

使用 JDBC

```
Public void saveMember(Member member) throws SQLException{

    …
    try {
         conn= dataSource.getConnection();
         stmt = conn.prepareStatement("insert into Member "
              + " (FirstName, LastName, … … … … … … CreateTime )"
              + " values (?, ?, ?, … … … … … … … … … … … … ?) ");

         stmt.setInt(0, member.getId().intValue());
         stmt.setString(1, member.getFirstName());
         stmt.setString(2,member.getLastName());
         …
         // 一个萝卜一个坑
         …
         // 红萝卜蹲完黄萝卜蹲
         …
         // 黄萝卜蹲完绿萝卜蹲
         …
         // 实在太累人了
         …
         stmt.setString(99,member.getCreateTime());
         stmt.executeUpdate();
    }
    catch(SQLException e) {
        …
    }
    finally {
        …
    }

// ORZ …
```

使用 Hibernate

```
public void saveMember(Member member) …
     Session session =
         HibernateHelper.getSessionFactory().openSession();
     Transaction tx = session.beginTransaction();
     session.save(member);
     tx.commit();
     session.close();
```

```
//清爽多了
使用 Hibernate 搭配"Spring Framework"[2]
public void saveMember(Member member) …
    getHibernateTemplate().save(member);
…
}
// 哈哈，这下变得更加简洁了
// 我们将在第 10 章介绍 Hibernate 和 Spring 的集成
```

仅仅是代码行数的显著差别，就已经让人迫切想要了解 Hibernate 了！在稍微了解各自的写法后，我们可以发现，从资源管理（Resource Management）、异常处理（Exception Handling）、事务（Transaction）下的 SQL 语句来看，Hibernate 的写法都简单了许多。而 Spring 则让 Hibernate 如虎添翼！如果你以前没有接触过 Hibernate，相信一定已经开始心动了！

针对刚才提到的 JDBC 的不足之处，Hibernate 分别做出了以下改进：

- **Hibernate 将 JDBC 进一步封装，其在 SQL 的可移植性方面做出的努力是有目共睹的。**
- **对于各种 ORM 的疑难杂症（不管是静态的结构层面，还是动态的行为层面），Hibernate 基本上都可以迎刃而解。**
- **Hibernate 在程序代码的简化上绝对是不遗余力的。**

我们在后续的章节中，会逐步揭开 Hibernate 强大的功能。

1.3 从体现“面向对象思想”的角度来看 Hibernate

记得当 UML/UP 开始流行的时候，多少人将它们视为救命仙丹，但是实践出真知，我们发现要想在业务逻辑层按“图”施工[3]，确实相当有难度。毕竟，从结构与行为两方面来看对象和数据库，两者是截然不同的。

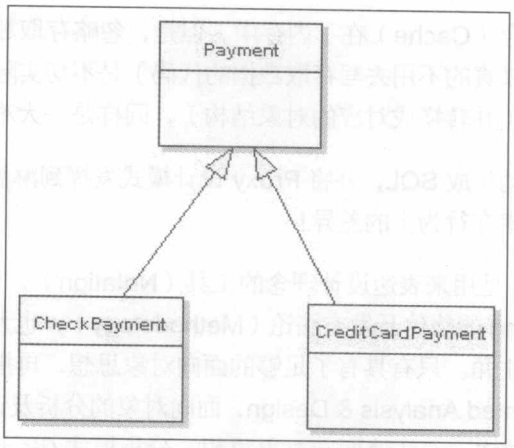
我们来看看两种常见的施工蓝图，下面是一张典型的类图（Class Diagram）。

---

2 改成这样会稍微好些（至少省去了在字段有所增减时改序号的麻烦）：

```
Int  i  =  -1;
stmt.setInt(i++, member.getId());
stmt.setString(i++, member.getFirstName());
stmt.setString(i++, member.getLastName());
```

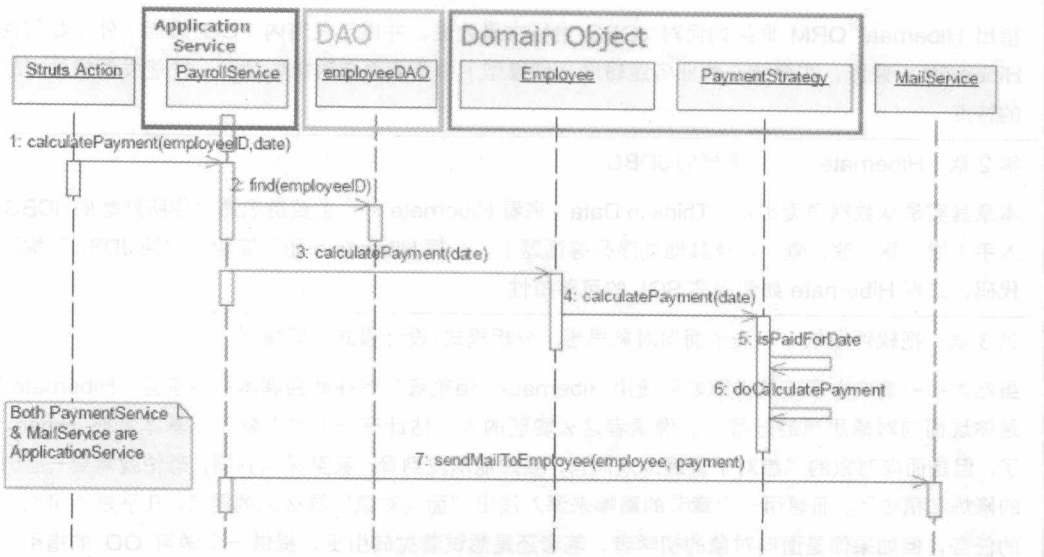
3 不要误会，笔者并非 UML 的忠实拥护者。是否真的具有 OO 的思想，远比会不会画 UML 图重要得多。



以上图中的继承来说，应该如何将对象对应到数据库的表呢？显然，将对象和表直接进行一一对应（有 3 种方法，请参考第 4 章），并不一定是最好的办法。而当两个对象存在上图所示的关系，在数据库的设计上，基于性能的考虑，往往将两个对象汇合到同一个表中。还有，多对多关系的 Class Diagram 和 E-R Diagram 的结构不会完全相同。这些种种结构上的不同，都是需要克服的问题。

域对象和数据库的表并非全然是一对一的对应！Hibernate 帮助我们解决了两者在结构上的不对称，并能够从一定程度上做到：当修改数据库的 Schema 时，不会影响现有的程序。

再来看看序列图（Sequence Diagram），如下图所示。



我们可以看到，几乎所有 OO 课本上的建模（Modeling）过程，其出发点似乎是假设所有的

表都已转换为对象，并缓存（Cache）在了内存中。但是，忽略存取数据的操作（或许我们可以换个角度思考：是不是可以真的不用去写存取数据的代码）是不切实际的。而且在实际中，一次该读取多少数据到内存中（并转换成对应的对象结构），同样是一大难题！

而 Hibernate 可以自动生成 SQL，并将 Proxy 设计模式发挥到淋漓尽致。在某种程序上，克服了这种面向对象与数据库在行为上的差异！

但话又说回来，UML 是用来表达设计理念的工具（Notation），而即使应用了重如 RUP、轻如 eXtreme Programming 的软件开发方法论（Methodology），也无法保证软件的成功，因为人才是把软件做“软”的主角。只有具有了足够的面向对象思想，再搭配优秀的 ORM 框架，才能让 OOAD（Object-Oriented Analysis & Design，面向对象的分析及设计）真正地发挥作用（我们将在第 3 章“把软件做软——浅谈面向对象思想、分析模式/设计模式、域模型”中来讨论 OOAD）。正所谓，剑（即 ORM 框架）随心（即 OO 思想）生——像 Hibernate 这样的 ORM 框架，不过是体现面向对象思想的手段罢了！

1.4 主线任务与支线任务

笔者分别用逐章导读和任务分配的方式，来组织本书的架构。各章简介如下表所示。

第 1 章　初探 Hibernate 指出 Hibernate ORM 框架如何对 JDBC 进行大量改进，并揭示只有内（OO 思想）外（如何使用 Hibernate）兼修，在最核心的业务逻辑层（域模型）中落实面向对象的精神，才能发挥软件“软”的特点
第 2 章　Hibernate，一个更好的 JDBC 本章其实是从数据角度出发（Think in Data）来看 Hibernate 的，主要是想通过你所熟悉的 JDBC 来入手（增、删、改、查，以及其他如序列等话题），一探 Hibernate 如何简化用“纯 JDBC”编写的代码，二探 Hibernate 如何提高 SQL 的可移植性
第 3 章　把软件做软——浅谈面向对象思想、分析模式/设计模式、域模型 虽然本书的重点主要还是讲解如何使用 Hibernate，但笔者只想在此强调本书的宗旨：Hibernate 3.x 是体现面向对象思想的好帮手。像读者这么聪明的人，估计有一个晚上就可以基本掌握 Hibernate 了，但是面向对象的思想对于普遍大众而言，要想能从皮到骨，甚至深入到髓，恐怕就需要一些时间的修炼和精进了。而想用一个章节的篇幅来深入浅出“面向对象”这么大的题目，几乎是不可能完成的任务，但如果你是面向对象的初学者，笔者还是想试着抛砖引玉，提供一个学习 OO 的指引。如果将全书比做一首诗，那么本章应该就是这首诗的“诗眼”了

第 4 章　开启对象和关系型数据库的一扇门——静态结构的解决之道 前文已经提到：域对象和关系型数据库的表，应该不会全部是一对一的对应，毕竟数据库要考虑到性能和数据量等因素。因此，E-R 的设计会有合并整理、正规范化/逆规范化（normalization / denormalization）的考虑（因此导致语意的失真）。而两者在结构上的差异，Hibernate 算是鞠躬尽瘁地为我们服务了
第 5 章　开启对象和关系型数据库的一扇门——动态行为的解决之道 本章不仅从动态行为方面探讨了 OO 和数据库的差异，以及 Hibernate 中相关的解决办法，还深入讲解了 Hibernate 的运行原理，以便读者更进一步地掌握 Hibernate。 特别是“5.2 持久化环境（Persistent Context）”一节，我们一定要好好地、用力地吸收消化，才有可能学好 Hibernate
第 6 章　事务！事务！！事务！！！ 像事务这样极其重要的话题，我们当然要另辟一章来好好地谈一谈。本章讨论了乐观锁（版本控制）/ 悲观锁等与事务有关的话题
第 7 章　更有效地使用 Hibernate 本章从效率、内存使用、代码的简化等角度，乘胜追击 Hibernate 的进阶使用
第 8 章　Hibernate 的两把查询利器——HQL & Criteria API 查询是重头戏，因此特辟一章对 Hibernate 中的查询娓娓道来
第 9 章　查漏补缺——更多的 Hibernate 应用 所有其他难以归类或比较另类，更为高级的话题和用法，均在本章集中讲解
第 10 章　实战演练 本章讨论了轻量级 Java EE 的架构，以及 Hibernate、Spring、Struts 的整合
第 11 章　向 Annotation 迈进 Annotation 已是大势所趋，本章介绍 Hibernate Annotation（包含 JPA 和部分 Hibernate 自带的 Annotation）的用法

如果将本书比做角色扮演游戏（RPG：显然笔者正迷失在龙与地下城[4]这个被遗忘的国度里），那么主线任务应该是，了解如何使用 Hibernate 实现面向对象的思想。

而其他大大小小的支线任务，略微说明如下表所示（建议你稍微有一个印象即可，日后根据需要再来回顾本节）。

4 龙与地下城（Dungeons & Dragons，D&D），是世界上第一个商业化的桌面角色扮演游戏。

关于“代码的简化”（相比 JDBC 而言） 基本上，建议你阅读完每一个段落后，都能思考书中的范例，如何用“纯”JDBC 来实现，两者再进行比较，这样对 Hibernate 的做法能有更深刻的印象
关于“减震点” 更改 Database Schema，能够尽可能地不需要修改程序代码，这是多么令人开心的事啊。Hibernate 的配置文件即扮演了这么一个“减震点”（或“防火墙”）的角色。你将在全书中看到 Hibernate 这种贴心的表现
关于“提升 SQL 的可移植性” 参考第 2 章、第 4 章、第 5 章、第 6 章、第 7 章、第 8 章、第 9 章、第 11 章
内存消耗 Hibernate 如果操作不慎，可能会导致内存使用量暴增（甚至爆掉），“5.5 进一步掌握持久层环境（Persistent Context）”一节特别谈到如何针对内存节衣缩食。此外，“7.2 大量数据的处理——Bulk & Batch”、8.3 节中的“游标滚动”和“org.hibernate.readOnly”等内容，都是你应该特别关注的
Hibernate 的头号大陷阱：LazyInitializationException Hibernate 的延迟加载的确带来了很大的便利，但是稍有不慎，很容易引爆 LazyInitializationException 这颗地雷。杜绝该问题的办法不只一个，请参考“7.3 延迟加载 v.s. 预加载”一节的内容，以及“10.2 实现 Hibernate 的基础架构（Infrastructure）”、“10.3 Spring Framework 诺曼底大空降”两节的 Open Session In View。 Hibernate 的延迟加载还影响到了 HQL 和 CriteriaAPI，请参考“8.1.5 HQL 的预加载”和“8.2.2 Join 与预加载”
“N + 1 Select”问题，对于初学 Hibernate 的人来说，可能也是一个不小的困扰。本书在“7.3 延迟加载 v.s. 预加载”、“8.1.5 HQL 的预加载”和“8.2.2 Join 与预加载”中都有讲解
关于“Hibernate 运行原理” 请参考第 5 章、第 7 章
关于“性能” 请参考第 5 章和第 7 章
关于“组能定义”
关于“开发辅助工具”（Hibernate Tool） 我们将在附录 C 中，以图解的方式来说明 Hibernate“开发辅助工具”的使用。Hibernate 大幅简化了存取数据库的代码，而“Hibernate Tool”可以帮助你更轻松地操作 Hibernate 框架
关于“测试”

此为试读，需要完整PDF请访问：www.ertongbook.com