宝典丛书

100万

Hibernate 项目开发宝典

没有枯燥乏味的理论，透过大大小小的案例全面覆盖使用Hibernate进行开发的知识点。

绝非简单的代码罗列，由资深Java开发专家为您细致讲解每个项目的开发细节，各个难点，逐一突破。

使用JSP，Struts，Spring以及JSF等多种组件进行集成开发，令您在开源组件中灵活组合、游刃有余。

陈天河　等编著

電子工業出版社
Publishing House of Electronics Industry
http://www.phei.com.cn

内 容 简 介

本书以Hibernate为核心，详细讲解了基于组件的J2EE应用软件的开发方法。在讲解方式上，使用基础知识与具体实例相结合的方式对Hibernate进行了全面、深入、细致的讲解，使读者在学习的过程中可以通过具体的练习来加深对讲解内容的理解和把握。

本书在讲解Hibernate的同时，还介绍了经常与Hibernate配合使用的Struts，JSF以及Spring等优秀的框架组件。另外，还涉及到了在开发中常会使用到的开源组件，包括Ant，Digester，XDoclet，Log4j，Eclipse和JFreeChart等。

本书的配套光盘中包含了书中所有实例的完整源代码以及在开发中需要用到的开源组件的安装程序。另外，作者提供了交流空间为广大读者进行后续的支持。

本书适用于使用Hibernate进行Web应用程序开发的技术人员，同时也非常适合对Hibernate相关技术感兴趣的读者进行学习和提高。

图书在版编目(CIP)数据

Hibernate项目开发宝典 / 陈天河等编著. —北京：电子工业出版社，2006.6
（宝典丛书）
ISBN 7-121-02634-1

Ⅰ.H... Ⅱ.陈... Ⅲ.JAVA语言－程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字（2006）第049335号

责任编辑：刘 舫
排版制作：华信卓越公司制作部
印 刷：北京天竺颖华印刷厂
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
经 销：各地新华书店
开 本：787 × 1092 1/16 印张：46 字数：1310千字
印 次：2006年6月第1次印刷
定 价：79.00元（含光盘一张）

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：（010）68279077。质量投诉请发邮件到zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

序

数据持久化已经成为软件开发中最重要的基础操作之一，是现在软件开发过程中的一个必要环节。在早期的Java领域中，是通过JDBC来完成数据的持久化操作的，但这种方式存在着很大的缺陷，它不能很好地解决面向对象的开发方式与结构化的数据之间的矛盾。为了解决这个矛盾，越来越多的开发人员转向了ORM(Object-Relational Mapping，对象－关系映射)组件的研究和使用。现在，Hibernate已经成为最受欢迎的ORM组件。

本书以Hibernate为核心详细讲解了基于组件的J2EE的应用软件的开发方法。在内容安排上采取了以具体实例为主线，讲解与实际练习相结合的方式，使读者在学习过程中可以通过具体的练习来加深对讲解内容的理解和把握。

本书的第1部分介绍了关于持久层的基础知识以及Hibernate的基本知识，然后介绍了软件环境的配置方法。接着，通过一个具体的实例讲解了如何使用Hibernate进行留言板系统的开发。最后，在前面实例的基础上对Hibernate的框架结构以及Hibernate的主要类进行了全面的讲解。在这一部分中还进行了Hibernate辅助工具的配置和使用方法的介绍。

本书的第2部分包括了两个实例。第一个实例介绍了如何使用JSP和Hibernate完成在线投票系统的开发。在这个项目中，通过JFreeChart组件实现了统计结果的图形化显示。第二个实例详细介绍了如何使用Struts和Hibernate来完成在线考试系统的开发。在这个实例中还对Struts的一些必要的基础知识以及集成方法进行了介绍。

第3部分的Hibernate的集成开发部分也包括两个实例。分别是用JSF，Spring和Hibernate完成的商品管理系统以及使用Struts，Spring和Hibernate完成的在线订货系统。这一部分重点讲解的是现在软件开发的方法以及如何实现Hibernate与不同的框架结构之间的集成与配合。

本书的作者希望读者不但能够通过本书了解到Hibernate的使用方法，更重要的是了解现在Web应用系统的发展情况，掌握现代软件的设计、集成和开发的方法，从而实现在整体上对Web应用系统的开发工作有深入的认识和提高的目的。

本书具有以下几个特点：

✦ 适用范围广

本书不但适用于具有一定Java开发经验的人员使用，同样适用于刚刚步入软件领域的初级程序员。本书从最基本的入门知识讲起，结合具体的开发实例使读者可以逐步掌握Hibernate的基础知识和使用方法。本书在内容的安排上特别注意知识的层次，除了基础的入门知识外还通过各种实例讲解了Hibernate与不同组件的集成开发方法，使读者可以更全面地了解Hibernate。

✦ 知识全面

现在的软件开发工作不再是只依赖于单一组件就可以完成的。本书以Hibernate为核心，同时还介绍了Struts，JSF以及Spring等常用的优秀框架组件。另外，还包括了在开发中经常会使用到的开源组件，包括Ant，Digester，XDoclet，Log4j，Eclipse以及JFreeChart等。

✦ 通俗易懂

本书语言平实，讲解透彻。对每一个专业术语都进行了详细的讲解，并辅助以多个完整实例加强读者对知识的理解。

✦ 实用性强

本书所介绍的开发方法是被目前大多数软件开发小组所采用的，讲解的具体开发实例也是软件开发人员经常会遇到的。

✦ 经验总结

本书作者具有多年的大型Java项目的开发经验，曾担任过多个项目的结构设计师、系统分析师。本书的讲解内容中包含了很多提示和注意事项，这都是作者的经验流露，使读者可以在软件开发过程中少走弯路，迅速丰富开发经验并提高开发水平。

本书的作者还提供了一个与读者进行交流的空间，网址是http://spaces.msn.com/programbible/。读者可将对本书或是编程语言方面的意见、建议、感想以及感兴趣的话题发布到这里，作者愿意与每一位热心的读者进行交流和互动。读者也可以通过电子邮件的方式与作者进行联系，电子邮件地址是programbible@hotmail.com。

参加本书编写工作的人员主要有陈天河、刘秀文、马连杰、刘书琴、陈秀菊、刘博、张小会、李铁、盖江南、王勇、胡淑霞、袁建洲、田翠丽、马琳、汪涛、杨晓霞、张华、李军、唐伟、邱燕明、董琳、王红利、薛荣华、牛力、马云、梁心东、关爱华、于红、李晓明、郭林、张苏伟、贾辉、郝杰、雷三兵、余旭东、林红、靳雪峰、沈大林、杨旭、马广月、顾瑞瑾、夏京、王坤、张磊、李强、张铮、张明玉、周荣先、林义雄、张昊、郑松、李稚平、崔元如、巩樱、蒋静、张秀梅、沈栋梁、江峰、张晓蕾等。

由于编者水平有限，书中疏漏和不足之处在所难免，敬请广大读者批评指正。

编　者

2006年6月

目　录

Part

第1部分 Hibernate基础知识

1

第1章 Hibernate与持久层技术

本章包括

- ✦ 持久化技术
- ✦ 持久层技术的实现
- ✦ Hibernate 简介
- ✦ 持久层技术
- ✦ ORM 概述

本章主要介绍持久层技术的基础知识、常用的持久化技术以及 Hibernate 的基础知识。

持久化技术

数据处理已经成为当今计算机应用系统中的最主要功能，而数据的存储无疑是开发一个应用系统最重要的工作之一。将数据保存到可掉电的存储设备中的过程就是进行数据持久化的操作。

“持久”一词对应于英文的“Persistence”。在计算机领域中，持久的数据就表示保存在掉电后也不会丢失数据的存储设备中的数据。相应的持久化操作通常表示将内存中的数据保存到磁盘的操作。在完成了数据的保存操作后，就表示相应的数据被持久化了。

现在的持久化操作通常是将数据“固化”到磁盘中，而“固化”的手段也是多种多样的。既可以是格式化的文本文件，也可以是 XML 文件，还可以是存在着复杂关系的数据库系统。

为了方便处理，绝大多数系统都会采用数据库来进行数据的持久化操作。本书所进行的数据持久化操作也是通过数据库来进行的。

持久层技术

持久化是将内存中的数据保存到磁盘的操作，而持久层则是软件开发模型领域的一个基本概念。

随着计算机软件开发技术的快速发展，应用软件的体系结构也由单层向两层、三层甚至更多层次的方向发展。随着数据库技术的发展和完善，现在的应用系统已经都采用两层甚至更多层次的体系结构了。

对于两层的开发模型，主要实现了应用层与数据层的分离，如图 1.1 所示。应用层负责接收用户的输入，然后进行业务逻辑的处理以及反馈处理的结果等一系列的工作。而数据层则只负责进行数据的持久化操作，保存业务中的持久化数据。

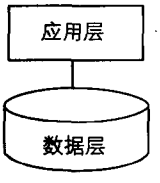

图 1.1 两层体系结构模型

在两层体系结构模型中，包括了以下两个开发层次。

- **应用层**：包括用户界面、业务逻辑处理和数据的持久化等功能。
- **数据层**：主要用于保存需要进行持久化的数据。

在两层体系结构中，实现了数据与应用的分离。

随着互联网技术的飞速发展，这种两层体系结构已经不能完全满足人们的需求了。这样就提出了三层的软件开发结构，将原来的应用层划分为表示层和业务逻辑层，结构如图 1.2 所示。这样就实现了业务逻辑和界面显示的分离，由此产生了 MVC（Model-View-Control，模型－视图－控制）模式的典型系统开发架构。这也是目前最受欢迎的软件开发架构之一。

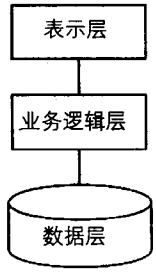

图 1.2 三层体系结构模型

在三层体系结构模型中，包括以下三个开发层次。

- **表示层**：提供用户界面的显示和与用户所进行的交互操作。
- **业务逻辑层**：用于进行业务逻辑的处理和数据的持久化操作。
- **数据层**：用于保存需要进行持久化的数据。

三层体系结构模型实现了业务逻辑和显示逻辑的分离，降低了程序开发的复杂度，并且特别适合于开发基于浏览器的应用系统。

现在，为了提高软件开发的效率以及降低软件开发的难度，更多的软件开发人员开始转向四层式的软件开发框架。它是在三层体系结构模型的基础上发展起来的，它将三层模型中的业务逻辑层分离出一个单独的持久层，进行数据的持久化操作，示意图如图 1.3 所示。

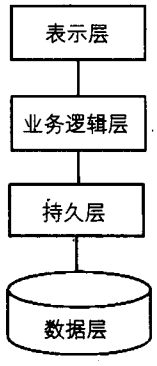

图 1.3 四层体系结构模型

在四层体系结构模型中，包括以下四个开发层次。

- **表示层**：提供用户界面的显示和与用户所进行的交互操作。
- **业务逻辑层**：进行软件核心业务逻辑的处理。
- **持久层**：用于进行对象的持久化操作。
- **数据层**：用于保存持久化数据。

这样做的好处是降低了业务逻辑层的复杂度，使其可以只关心业务处理，而相应数据的持久化操作则交给持久层来进行。这样实现了业务逻辑与数据的持久化操作的分离，减轻了业务逻辑开发的工作量和复杂程度。同时，数据的持久化可以通过其他的组件来完成，Hibernate就是用来完成数据的持久化操作的优秀解决方案之一。

在这里需要注意的是持久层并不是进行简单的持久化操作。它应该建立在整个系统开发的一个逻辑层面上，是专门用于实现数据持久化的一个相对独立的领域。它对于系统的其他部分而言，应该具有较为清晰和严格的边界，并能够提供完整的数据持久化的解决方案。

通过上面的分析可以简单地了解什么是持久化、什么是持久层以及它们之间的关系。简单地说，持久化是一个动词，意味着某种动作或者机制的执行。通过进行持久化操作，可以将内存中的数据保存到数据库中或者其他的媒介中以供以后使用。而持久层则是一个名词，它代表着软件体系结构中的一个逻辑层次。在这个层次中，可以将数据对象与数据实体联系起来，实现数据的持久化操作。

那么，什么样的系统结构才能说具有持久层呢？也就是说具有怎样特征的软件系统才能说具有独立的持久层呢？对于这个问题没有一个明确的答案，通常认为满足下面这些条件的软件系统就可以说具有一个独立的持久层：

- 当显示层的显示形式和实现机制发生改变时，持久化代码不需要修改和编译。
- 当业务逻辑发生改变的时候，持久化代码不需要进行修改和编译。
- 当数据层的持久化机制发生改变的时候，持久化的代码不需要进行修改和重新编译。

当然，这些条件只能作为确定一个系统是否具有独立的持久层的基本特征，在不同的情况下，也会存在着一定的变化。通过持久层的判定方法也可以看出现在所采用的多层逻辑结构进行软件开发的最大好处是可以降低系统的耦合度，使一个部分的修改不会影响到其他部分的正常使用。

使用Java语言实现一个独立持久层的方法是多种多样的，既可以使用最基础的JDBC来实现，也可以通过各种持久层的独立组件来完成。下面就来看一下目前主要使用的持久层实现方法。

持久层技术的实现

数据持久层的设计目标是为整个项目提供一个高层、统一、安全和并发的数据持久机制，完成对各种数据进行持久化的编程工作，并为系统业务逻辑层提供服务。数据持久层提供了数据访问方法，能够使其他程序员避免手工编写程序访问数据持久层（Persistence layer），使其专注于业务逻辑的开发，并且能够在不同项目中重用映射框架，大大简化了对数据进行增加、删除、修改、查询等功能的开发过程，同时又不丧失多层结构的天然优势，继承延续J2EE特有的可伸缩性和可扩展性。

由于现在绝大多数的应用系统都采用数据库来保存持久化的数据，因此持久层的实现也就和

数据库紧密地联系在一起了。我们知道，在Java语言中，访问数据库的普遍做法是通过JDBC(Java Database Connectivity，Java 数据库连接) 技术来进行的，这样就自然而然地联想到通过 JDBC 来实现独立的应用系统的持久层。

当然，并不是所有的系统都将数据持久化到数据库中，也可以直接保存到文件中。例如，Java 已经提供了串行化的机制，可以通过串行化来将 Java 对象转化为二进制的字节流，然后就可以直接保存到文件中了。然后通过反串行化又可以得到原来的对象，但这种持久层实现的方法不在本书的讨论范围之内。本书只讨论如何使用 Java 语言实现将数据保存到数据库中的方法。

对于数据库访问技术来说，现在主要存在两种主流的数据库访问技术，它们分别是：

- ✦ JDBC (Java Database Connectivity)
- ✦ 微软公司开发的相关的数据访问技术

数据库访问技术只是访问数据库的手段，而数据库访问的模式才是开发中需要重点关注的内容。架构师应该能够根据不同的应用环境来选择不同的数据库访问模式，以获得易用性、开发成本、执行效率等多方面的最优效果。

下面就来了解一下目前比较常见的几种数据持久化的实现方案。

- ✦ 通过会话 Bean 和实体 Bean 来实现数据的持久化

对于这种实现数据的持久化的方法是通过会话Bean和实体Bean来共同完成的。对于采用这种方法实现的软件系统的最大缺点是持久层对资源的消耗极大 (主要是内存)，通常会导致系统的运行速度缓慢。

这种数据持久化的实现方法的示意图如图 1.4 所示。

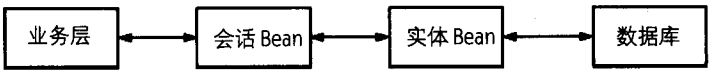

图 1.4　使用会话 Bean 和实体 Bean 实现数据的持久化

- ✦ 通过 DAO 和 JDBC 来实现数据的持久化

在这种数据持久化实现方案中通过 DAO (Data Access Object，数据访问对象) 来实现数据的持久化操作。在这种实现方法中，DAO 通过 JDBC 来完成对数据库的存取操作，实现了 Java 对象的持久化操作。但这种方法要求开发人员对 JDBC 的底层信息要非常熟悉，并可以依据不同的需求来完成不同的功能。

这种数据持久化的实现方法的示意图如图 1.5 所示。

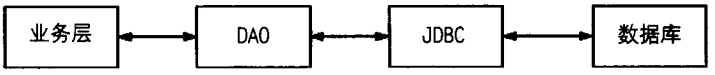

图 1.5　通过 DAO 和 JDBC 来实现数据的持久化

- ✦ 通过 DAO 和 ORM 组件来实现数据的持久化操作

在这种数据持久化的实现方案中，通过DAO来实现Java对象的持久化操作，而ORM(Object-Relational Mapping) 组件则用来实现Java对象与数据库中数据之间的相互转换，同时它还包括了数据库的连接管理、事务管理以及对象的缓冲管理等功能。

这种数据持久化的实现方法的示意图如图 1.6 所示。

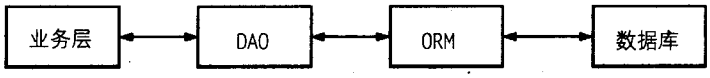

图 1.6　通过 DAO 和 ORM 组件来实现数据的持久化

对比以上三种数据的持久化的实现方法，它们之间的区别主要表现在以下几个方面。

✦ 内存消耗

对于这三种实现方案而言，Entity Bean（实体 Bean）结构的实现方法是最消耗内存的，所需要的内存数量非常巨大。而使用 JDBC 的方案则是最节约内存的，ORM 组件对内存的消耗量介于前两者之间。

✦ 性能对比

对于这三种实现方案，针对不同的系统而言，它们的性能表现是不同的。在理论上，JDBC 架构的代码量是最少的，运行效率也应该是最高的。但在实际中，这一点几乎做不到，这需要程序员精通 JDBC，运用 Batch 语句，调整 PreparedStatement 的 Batch Size 和 Fetch Size 等参数，以及在必要的情况下采用结果集 cache 等，但是一般情况下程序员是做不到这一点的。

ORM 组件这种实现方案取得了另外两种方案的一个平衡。它所占用的内存量位于前两种方案之间，在实际运行中的代码量也是介于两者之间的。同时它又专门具有对持久化操作所进行的各种优化，降低了对开发人员的要求。因而，采用 ORM 组件的数据持久化实现方法具有更好的适应性，可以应用于绝大多数的软件系统。

✦ 开发效率

对于现有的 Eclipse 和 JBuilder 等开发工具，都有专门支持 Entity Bean 和 ORM 的插件，因而这两种方法都具有很高的开发效率。而使用 JDBC 进行数据持久化的操作方法则由于自己开发的代码过多而导致开发过程比较烦琐。

以上介绍了三种常见的数据持久化的实现方法，下面来介绍一下未来主流的持久层实现框架——ORM 框架。

ORM 概述

ORM 的全称是 Object-Relational Mapping，翻译成中文就是“对象—关系映射”。ORM 组件的主要功能是实现实体域对象的持久化并封装数据访问的细节。

在对象—关系映射中涉及到的两个关键点是：Object（对象）和 Relation（关系）。它们分别代表了目前应用系统中所要处理的绝大多数工作——对对象的操作和对关系型数据库的访问。

当今的应用系统大多采用了多层次的体系结构，在业务逻辑层和表示层将系统中各参与实体进行了面向对象的封装，可实现对现实世界的高度抽象。但在持久层中，目前的数据库都是关系型数据库，所以就必须通过持久层来实现 Java 对象与数据库中数据之间的相互转换，很方便地可以完成对 Java 对象的持久化操作以及对数据库中数据的读取操作（还需要将数据转化为 Java 对象）。这样，实现这一功能的持久层实现框架——ORM 组件就诞生了。

在这里需要注意，ORM 本身并不是一个组件，它是具有某种功能的组件的总称，也可以说是一种框架结构。

实现 ORM 功能的组件有很多，其中 Apache 软件基金组织的 OJB（ObJect-Relational Bridge）组件和现隶属于 JBoss 组织的 Hibernate 是目前最常被使用的 ORM 组件。

OJB官方网站的网址为http://db.apache.org/ojb/。关于其具体信息请参考其官方网站以及相关的资料。

在本书中主要介绍JBoss组织的Hibernate组件。

1.5 Hibernate简介

Hibernate是采用ORM模式实现数据持久层的一个优秀的Java组件，它提供了强大、高效的将Java对象进行持久化操作的服务。利用Hibernate，开发人员可以方便地按照Java对象的结构进行持久层的开发，并可以使用Hibernate所提供的HQL（Hibernate Query Language，Hibernate查询语言）完成Java对象和关系型数据库之间的转换和操作。

Hibernate官方网站的网址为http://www.hibernate.org/。对于其更具体的介绍请参考其官方网站以及相关的资料。

2001年末，Hibernate的第一个正式版本发布了。它在发布之初就受到了开发人员的热烈欢迎，并得到了大家的支持和赞许。2002年6月，Hibernate 2的发布则为Hibernate的成功奠定了坚实的基础。随后，Hibernate获得了Jolt2004大奖并被著名的开源组织JBoss收纳，成为JBoss的子项目之一。2005年3月，Hibernate 3正式发布了。它使得Hibernate发展到了一个崭新的高度，在性能、灵活性和可扩展性方面得到了进一步的提升，在很大程度上已经超越了其他持久层技术。

现在，Hibernate已经成为最具影响力的ORM工具了，被广泛地应用于实现各种应用系统的持久层。

1.6 小结

在本章中主要对什么是持久化技术、什么是持久层技术进行了简单的介绍，使读者对这两个基础的概念有了一个明确的区分。

在接下来的内容中介绍了持久层技术的主要实现方法以及这些实现方法的对比特点，并在此基础上介绍了什么是ORM以及目前最好的实现ORM的组件——Hibernate的简要发展过程。

本章的重点在于使读者对持久化操作以及相关概念有一个初步的了解，为后面的讲解打下基础。

✦ ✦ ✦