案例源代码
多媒体视频

精通

黄佩虹　张冰晶　编著

Hibernate

—Java数据库持久层开发核心编程

- Hibernate基础及原理解析
- 对象关系映射、数据操作
- 事务与并发处理、数据检索
- 二级缓存和查询缓存
- 拦截器和事件框架
- Annotations应用
- Hibernate验证器
- Hibernate Tools介绍
- Hibernate Search
- Spring和Hibernate整合

清华大学出版社

精通

黄佩虹　张冰晶　编著

Hibernate

—Java数据库持久层开发核心编程

清华大学出版社
北京

内 容 简 介

Hibernate 持久层框架是优秀的解决对象关系不匹配问题的解决方案，它使得持久层开发人员可以方便地解决对象关系映射问题。另外，Hibernate 在性能上也提供了许多优化策略，接口简单，操作灵活，具有可扩展性，对 JDBC 仅做了轻量级封装，而且，多种框架已经显式地对 Hibernate 提供了支持的接口。Hibernate 已成为当前 Java 世界最流行的持久层框架之一。

本书共分 16 章，内容包括：使用 Hibernate 的对象关系映射，使用 Hibernate 进行对象保存、更新、删除、读取操作，对象检索，事务处理，二级缓存，查询缓存，事件框架和拦截器的使用；还介绍了 Hibernate 官方网站的推荐项目 Hibernate Annotations、Hibernate Validators、Hibernate Tools 和 Hibernate Search 以及与 Spring 结合使用的方法。本书循序渐进地指导初学者从入门到较全面地掌握 Hibernate 的高级特性，以及结合其他技术使用 Hibernate 进行项目开发，内容全面，案例清晰，实践性强。

本书适合 Hibernate 的入门者学习使用，同时也适合对 Hibernate 有一定了解的读者在项目开发时进行查阅。

图书在版编目(CIP)数据

精通 Hibernate——Java 数据库持久层开发核心编程/黄佩虹，张冰晶 编著. —北京：清华大学出版社，2009.3

ISBN 978-7-302-18886-5

Ⅰ.精…　Ⅱ. ①黄… ②张…　Ⅲ. JAVA 语言—程序设计　Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2008)第 175661 号

责任编辑：王　定　刘金喜
封面设计：久久度文化
版式设计：康　博
责任校对：胡雁翎
责任印制：孟凡玉
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京鑫丰华彩印有限公司
装 订 者：三河市新茂装订有限公司
经　　销：全国新华书店
开　　本：185×260　**印　张**：35　**字　数**：808 千字
附 DVD 光盘 1 张
版　　次：2009 年 3 月第 1 版　　**印　　次**：2009 年 3 月第1 次印刷
印　　数：1～4000
定　　价：65.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103　　产品编号：027541-01

很多时候，应用程序都需要对数据进行永久化的存储，特别是企业级应用中，更是需要存储大量的数据。关系数据库是目前主要的数据存储库。在应用程序的设计和实现中，面向对象的开发方法已经成为主流。在持久化数据的问题上，程序员不得不面对这种面向对象设计和关系数据库存储的不匹配所带来的问题，即 ORM(Object Relation Mapping)问题。Java 世界先后涌现出大量的持久层框架，目的就是让程序员以面向对象的思想来进行数据持久化工作，而把对象关系映射问题交给框架处理。其中，Hibernate 是这些持久层框架中最著名的项目之一。

Hibernate 提供了优秀的对象关系映射解决方案，在性能上也提供了许多优化策略，源代码开放，文档详细全面，接口简单，操作灵活，具有可扩展性，对 JDBC 仅做了轻量级封装，所以必要时还可以直接使用 JDBC 的 API，而且，多种框架已经显式地对 Hibernate 提供了支持的接口。所有这些特点，都使得 Hibernate 成为当今最流行的持久层框架之一。目前 Hibernate 的最新版本为 3.2 版。

本书从 Hibernate 的入门程序开始讲解，然后一步步深入地介绍 Hibernate 的一些高级应用，最后介绍 Hibernate 官方网站介绍的一些结合 Hibernate 和其他流行框架技术的项目以及 Hibernate 的运用实例。本书可以分为 3 大部分。

第 1 部分为入门篇，包括第 1 章~第 5 章。其中，第 1 章说明了什么是 Hibernate 并阐述了 Hibernate 的优点；第 2 章介绍了 Hibernate 的入门程序；第 3 章介绍了 Hibernate 的体系结构和原理；第 4 章讲解了如何使用 Hibernate 进行对象关系映射；第 5 章讲解了如何进行对象的保存、更新、删除、检索等操作。通过这些章节的学习，读者对 Hibernate 整体有一个了解，并且将学会使用 Hibernate 进行简单的对象关系映射和数据操作等持久化对象的相关工作。

第 2 部分为提高篇，包括第 6 章~第 10 章。其中，第 6 章介绍了 Hibernate 的事务；第 7 章介绍了使用 Hibernate 的组件、动态类等高级的对象关系映射方式；第 8 章讲解了 Criteria 查询和 Hibernate 的关联检索方式；第 9 章介绍了 Hibernate 的二级缓存和查询缓存；

第 10 章介绍了 Hibernate 的事件机制和拦截机制。通过这些章节的学习，读者将学会运用 Hibernate 的一些对象关系映射和数据检索的高级特性进行更灵活的持久化操作，并且学会选择和使用与 Hibernate 性能相关的缓存设置，以及使用 Hibernate 的扩展机制。

第 3 部分为实战篇，包括第 11 章~第 16 章。具体讲解了 Hibernate 官方网站推荐的 Hibernate 的多个相关技术项目。其中，第 11 章介绍了 Hibernate Annotations；第 12 章介绍了 Hibernate Validators；第 13 章介绍了 Hibernate Tools；第 14 章介绍了 Hibernate Search；第 15 章介绍了 Hibernate 和流行的应用程序框架 Spring 的结合使用方法；第 16 章介绍了一个结合 Hibernate、Spring、Struts 的项目实例。

本书讲解循序渐进，内容全面，并且在讲解每个知识点时都结合了案例代码进行解释，读者既能很容易地根据本书的章节安排入门，也可以较全面地对 Hibernate 进行学习。本书使用了 Hibernate 3.2 的最新版本进行介绍，内容包括了 Hibernate 3.2 的新特点。另外，实战篇中介绍的多个 Hibernate 和其他框架技术结合的项目，使得读者可以方便地学会如何结合 Hibernate 和其他项目进行开发。本书还配有完整案例源代码和安装软件环境配置指导，使得读者可以方便地进行学习。本书适合 Hibernate 的入门者，也适合对 Hibernate 有一定了解的读者在项目开发时进行查阅。

本书由黄佩虹和张冰晶编著。此外，张暘、曹川、杨丽娜、赵璇、鲍斌、季军、李伟、朱庆友、涂德志、王占富、周坤、沈永亮、张杨、杨明、司文峰、李长健、董军、黄锐等同志在项目案例编写上给予了编者很大的帮助，在此，对他们表示衷心的感谢。

由于时间仓促，加之水平有限，书中不足之处在所难免，敬请读者批评指正。

服务邮箱：wkservice@vip.163.com.

编　　者

2008 年 10 月

CONTENTS 目录

CHAPTER 1

为什么使用 Hibernate

Hibernate 是一个持久层框架，提供了一个具体的 ORM 解决方案。Hibernate 实现对象和数据库表，对象属性和数据库表列的映射，并实现它们之间的自动转换工作。Hibernate 内部封装了 JDBC 操作实现底层的对数据库的读写，对上层提供了对实体对象保存、更新、删除、检索的面向对象的 API。使用 Hibernate 实现持久层，使得开发人员可以以面向对象的思想进行持久化工作。

本章首先介绍什么是持久层，什么是 ORM，再介绍传统的使用 JDBC 编程的方法。然后通过一个例子对比使用 JDBC 编程和使用 Hibernate 进行持久化操作的不同，包括 Hibernate 如何提供更方便的接口让用户进行持久化工作，以及 Hibernate 如何解决对象关系映射中关联、继承、数据库表更改等问题。最后总结了使用 Hibernate 的优点。

本章的主要内容包括：

- 应用程序分层结构和持久层的概念
- 对象关系映射，即 ORM 问题
- 使用 JDBC 进行编程的基本方法
- 使用 JDBC 编程和 Hibernate 编程的不同，以及 Hibernate 如何解决 ORM 中的对象关系不匹配问题
- 使用 Hibernate 的优点

1.1 持 久 层

1.1.1 应用程序的分层结构

一般的企业级的应用程序中都会有较明显的分层结构，分为表现层、业务逻辑层、持久层，再接入数据库，如图 1-1 所示。

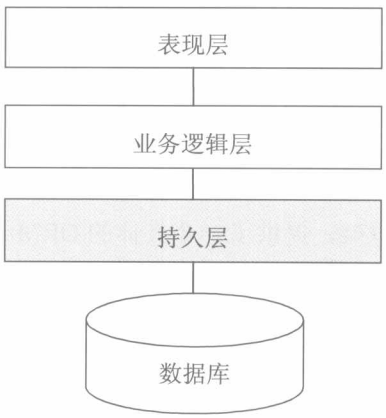

图 1-1 应用程序的分层结构

下面介绍各层的定义。

- 表现层

用户界面的逻辑位于最顶层。表现层负责把用户要求的业务逻辑处理结果以可视化的友好的方式返回给用户，并提供接受用户命令的接口和表现层页面控制逻辑的代码。

- 业务逻辑层

业务逻辑层负责处理问题领域的业务规则和根据用户需求进行的业务处理以满足用户的功能需求。通常情况下，业务逻辑层处理使用的实体对象由持久层提供。

- 持久层

数据通过持久层进行持久化。所谓持久化，即把数据(如内存中的对象)保存到可永久保存的存储设备中(如磁盘)。一般情况下，是将内存中的数据存储在关系型的数据库中。而持久层，即专注于实现数据持久化应用领域的某个特定系统的一个逻辑层面，将数据使用者和实体对象相关联。对于业务逻辑层，返回服务业务层需要的实体模型，对于数据库，把实体对象转换为关系数据库存储的形式与数据库交互。

持久层的设计，使得业务逻辑层只需要负责业务逻辑的实现，而把对数据的操作交给了持久层。持久层对数据及对数据操作的封装有以下几个优点：

(1) 屏蔽数据库平台的变化对业务逻辑层的影响。当数据库变化时，只需修改持久层操

作数据库的代码，而持久层提供给业务逻辑的对象模型没有变化，从而避免了业务逻辑的修改。

(2) 通过持久层的封装处理，可以在持久层实现支持多种数据库平台，而对业务逻辑层提供统一的接口。

(3) 代码可重用性高，能够完成所有的数据库访问操作。

通过持久层的设计，将复杂的业务逻辑和数据逻辑分离，降低系统的耦合程度，从而在开发时更明确地进行分工，维护工作也更容易进行，系统的体系结构也变得更加清晰。

1.1.2 持久化技术

Java 世界中，有多种持久层可以采用的用于持久化实体对象的技术，使用 SQL 和 JDBC 进行硬编码是最直接的方式。即在持久层使用 SQL 语句进行对象到数据库表，对象属性到数据库表列的转换来实现保存、更新、删除和载入。这种方式设计的 JDBC 实现将在 1.3 节做介绍。其他的常用的持久化技术还包括：

- JDO

Java Data Objects(JDO)是存储 Java 对象的规范，而且 Sun 公司还为它制定了用于描述对象持久化语义的标准 API。JDO 规范 1.0 的提出可以使用户将精力集中在设计 Java 对象模型上，然后在应用程序中存储和使用这些 Java 对象，采用 JDOQL 语言进行数据库读写操作。

- EJB

J2EE 的规范为 EJB 定义了两种持久化的解决方案：一种是 BMP，另一种是 CMP。CMP(Container-Managed Persistence)表示由 EJB 容器来管理实体 EJB 的持久化，EJB 容器封装了对象关系的映射及数据访问细节。CMP 提供对象关系映射服务，把对象持久化的任务从业务逻辑中分离出来。CMP 负责持久化实体 EJB 组件。CMP 的使用比较复杂，而且想要非常清楚地了解 EJB 规范也是非常费时的。

BMP(Bean-Managed Persistence)，即由 EJB 本身管理持久化。也就是说，由实体 EJB 自身管理数据访问细节。这种方式也称为主动域对象模式。主动域对象本身位于业务逻辑层，因此采用主动域对象模式时，整个应用仍然是三层应用结构，但是实际上并没有从业务逻辑层分离出独立的持久层。

- 持久层框架

Java 世界还涌现出许多轻量级的持久层框架，用于在设计时帮助持久层实现对象关系映射。最著名的即 Hibernate 框架，还有 iBatis、SMYKE、mirage、Cayenne 等许多开源的持久层框架。它们都意在内部实现对象关系的映射，使得开发人员可以使用它们面向对象的 API 进行对象的持久化操作。

1.2 ORM 简 介

ORM 全称是 Object Relational Mapping，即对象关系映射。

当使用面向对象的思想来设计应用程序时，通常都会通过领域建模抽出其中的对象实体作为实体类对象。其中，封装、继承、多态、依赖以达到高内聚低耦合是设计类对象时常涉及的面向对象思想和技术。

在运行应用程序时，对象是保存在内存中的，但是当对象不在运行逻辑时经常需要被保存到存储设备中。如果是存储在关系数据库中，数据库只是一系列二维表的集合，而对象是一个多维结构，那么对象中的一些关系，如继承、集合关联在数据库的二维表中就无法直接表示。怎样建立这两者之间的映射关系而不会造成保存到数据库后对象模型中的信息丢失呢？这就是 ORM 所尝试解决的问题。

一个 ORM 解决方案应该包括以下几方面的内容。

(1) 提供对持久化对象进行保存、删除、更新、载入等操作的接口。

(2) 提供面向持久化对象及其属性的查询语言或接口。

(3) 提供一种方便的方式用于定义对象类和数据库表以及对象属性和数据库表属性映射。

(4) 提供对事务中的对象进行是否脏数据的检测以达到数据库数据和持久化对象的同步。

1.3 使用 JDBC 编程

Java 应用程序访问数据库最直接的方法就是使用 JDBC 的 API，Hibernate 实现持久化对象的操作底层也是最终转化为 JDBC 来完成的。JDBC 全称是 Java Database Connectivity。java.sql 包提供了 JDBC 的 API。先来看一个使用 JDBC 编程的例子。

JdbcTest代码如下，这个类使用传统的JDBC的API来连接数据库，并进行数据的查询、保存、更新、删除等操作。可以看到使用JDBC编程，这些操作都需要使用SQL语句来实现。

```
import java.sql.PreparedStatement;
import java.sql.Date;
import java.sql.ResultSet;
import java.sql.Connection;

import basicCar.bean.BasicCar;
import basicCar.bean.Lorry;
import basicCar.bean.Salesman;
import basicCar.bean.SaloonCar;
```

```
public class JdbcTest {
 //连接数据的 url
 private String dbUrl = "jdbc:mysql://localhost:3306/test";

 //连接数据的用户名字
 private String dbUser = "root";

 //连接数据的密码
 private String dbPwd = "111111";

 //代表数据库连接
 private Connection con = null;

 //负责执行 SQL 语句，具有预定义 SQL 语句的功能
 PreparedStatement stmt = null;

 public static void main(String[] args) throws Exception {
        JdbcTest jdbc = new JdbcTest();
        jdbc.doConfiguration();
        jdbc.saveEntity();
        jdbc.updateEntity();
        jdbc.deleteEntity();
        jdbc.queryEntity();
 }

 //加载数据库驱动程序，并获得数据库连接
 void doConfiguration() {
        try {
              //加载数据库驱动程序
              Class.forName("com.mysql.jdbc.Driver");
              //使用用户名、密码、数据库 url 获得数据库连接
              con = java.sql.DriverManager.getConnection(dbUrl, dbUser, dbPwd);
        } catch (Exception e) {
              System.out.println("exception");
        }
 }

 //向数据库中插入一条记录
 void saveEntity() throws Exception {
        stmt = con.prepareStatement("insert into salesman(sid,salesname)"
                    + "values(?,?)");
        stmt.setInt(1, 116);
```

```
        stmt.setString(2, "hello");
        stmt.execute();
    }

    //向数据库中更新一条记录
    void updateEntity() throws Exception {
        stmt = con.prepareStatement("update basiccar "
                + "set name=?, date=? where id=?");
        stmt.setString(1, "carname");
        stmt.setDate(2, (java.sql.Date) (Date.valueOf("1994-03-04")));
        stmt.setInt(3, 4);
        stmt.execute();
    }

    //向数据库中删除一条记录
    void deleteEntity() throws Exception {
        stmt = con.prepareStatement("delete from salesman where id=116");
        stmt.execute();
    }

    //查询数据库
    void queryEntity() throws Exception {
        ResultSet rs = null;
        stmt = con
                .prepareStatement("select s2.sid , s2.salesName    from basiccar11 lorry"
                        + " inner join carorder carOrder on lorry.id=carOrder.carId "
                        + "inner join salesman s2 on carOrder.salesId=s2.sid "通过
                        + "where lorry.carType='lorry'");
        rs = stmt.executeQuery();
        while (rs.next()) {
            System.out.println("The person sell lorry: " + rs.getString(2));
        }
    }
}
```

使用 JDBC 编程，主要涉及以下几个类。

- DriverManager：驱动程序的管理器，负责创建数据库连接。创建数据库连接时，以连接数据库的用户名、密码，url 等作为连接的参数。主要通过 DriverManager 的 getConnection()函数获得一个数据库连接。
- Connection：代表数据库连接的类。通过其 prepareStatement()方法获得一个 PreparedStatement 对象。
- PreparedStatement：SQL 语句的预编译接口。它继承 Statement 接口，表示预编译