Broadview
WWW.BROADVIEW.COM.CN

Java技术大系

精通 Hibernate

刘 洋 编著

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

Java 技术大系

精通Hibernate

刘 洋 编著

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

内 容 简 介

本书深入地介绍了 Hibernate 实现 ORM 的关键技术，包括 Hibernate 的底层技术、Hibernate 的实现架构、Hibernate 的配置和使用方法、EJB3.0 技术和基于 Hibernate 的开发实例。通过本书，可以全面了解 Hibernate 的技术和开发方法，深入理解 Hibernate 的数据持久化设计，掌握数据层应用开发的方法。

本书适合所有想了解、利用 Hibernate 的技术开发人员阅读和参考，也可作为企业和院校的相关培训教材。

图书在版编目（CIP）数据

精通 Hibernate / 刘洋编著. —北京：电子工业出版社，2005.5
（Java 技术大系）
ISBN 7-121-01047-X

Ⅰ. 精… Ⅱ. 刘… Ⅲ. JAVA 语言－程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2005）第 022957 号

责任编辑：孙学瑛 高洪霞
印　　刷：北京东光印刷厂
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：29　字数：644 千字
印　　次：2005 年 5 月第 1 次印刷
印　　数：6000 册　定价：45.00 元

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

前　言

开放源代码运动得到了世界范围的支持，操作系统、办公软件、数据库、浏览器、中间件等越来越多的技术领域已经拥有了最为成功的开放源代码项目，不仅是 Linux 世界，基于 Java 的所有技术领域几乎都有开放源代码项目的异军突起，同时多位开放源代码项目的核心人物在领导着 Java 技术的发展。本书和作者的另一本书《精通 JBoss——EJB 与 Web Services 开发精解》都旨在向国内的软件开发人员率先介绍国外最先进的开放源代码项目，在详细介绍其项目内容的同时解释其核心技术和成功经验，希望能够有利于国内的自主研发项目。

数据持久化是一个如此基础的技术环节，以至于在技术发展的各个时期都要提供对应的解决方案。在 Java 的世界里一直在找寻着最佳的处理方法，包括 JDBC，CMP，JDO 和 Hibernate 等 ORM 工具项目的探索，目前 Hibernate 可能是应用最为成功和广泛的工具，但是技术世界对持久化方式的追求并没有结束，通过创新的技术手段可以不断提高数据持久化的开发效率，本书的第 1 章甚至介绍了一种几乎不需要写 Java 代码的实现方式，而且 Hibernate 自身也在发展。可见，技术的发展不会停止。

所有接触过 Hibernate 的开发者几乎都会为其感到惊讶，这个工具让他们几乎脱离了从前烦琐的 SQL 语言的工作任务，更不用说那些 EJB 2.0 中的复杂配置了，似乎整个的开发工作都因此而变得更加轻松，包括数据库表的设计和建立、对原有数据的整合、修改复杂的数据关系、查询性能的优化等，重复的工作得以自动化，困难的部分变成了简单的设定，复杂的任务有时只需要一行代码来完成，一切都变得如此方便，整体项目的代码也因此而变得简洁易懂。

目前的 Hibernate 3.0 比之前的 2.1.x 版有重大的改进，除了全面支持 J2SE 5.0 以外，Hibernate 3.0 实现了 EJB 3.0 的 ORM 部分，对于 Java 的开发者来说不用再在 Hibernate 和 EJB 之间做出选择了，它们已经统一为 EJB 3.0 标准。作为 Java 技术的开发者和爱好者请尽快升级你的技术储备到当前的标准版本上来吧！

本书详细介绍了 Hibernate 的底层技术、使用方法、扩展特性和以 Hibernate 3.0 为基础的 EJB 3.0 标准实现工具。本书的代码示例可以从 http://www.broadview.com.cn 网站下载得到，使用这些代码需要使用 J2SE 5.0、Hibernate 3.0、Eclipse 3.1M4、JBoss 4.0.1 以上版本的开发工具和软件环境。

由于时间仓促，不足之处在所难免，欢迎读者批评指正。

刘　洋

目　录

第 1 章　数据的持久化之道

数据是软件系统中不可缺少的环节，数据持久化的解决之道也一直是企业计算标准中最有影响力的部分。如今关系型数据库已经稳定地占据着主要的数据库市场，XML 技术和 XML 数据库技术也日渐成熟。本章试图用一个简单的小例子来讨论一下数据持久化的大问题，并以这个角度来介绍主要的数据持久化方法。

1.1　持久化数据

数据是我们软件开发人员必须考虑的对象。无论你使用哪种技术、借助于哪种工具，以及购买了哪种软件平台产品，当接手一个实际的问题时，你都需要面对实际的数据，考虑实际数据的各种问题。在这些问题中往往都包含数据持久化的问题，这不仅是因为我们的计算机内存有限，更主要的原因是业务逻辑的需要，作为一个接触软件技术、尝试写程序的人，我们要接触各种类型的数据。下面列举了 3 种常见类型的数据，让我们来尝试一下找到持久化这些数据的方法。

1.1.1　随机数据

称它为“随机数据”是因为实在找不出其他的形容词来描述，这种数据指的是不确定结构，不了解数量，目前也搞不清关系的数据。它听起来很奇怪，但在实际的程序设计中十分常见。例如，从一个振动源传递到计算机采集卡上的数据，再比如一个自然物生长过程中的数据。可能有些数据实际上是很有规律的，但是在设计程序时由于对它的了解很少，使它对于我们来说也成了随机的数据了。

我曾经在遇到这种搞不太清的数据时，使用很大的数组或者多个链表来表述，不管下一个数据的值是什么，先存储起来再说。有时候当遇到的数据总也摸不到规律，或者越来越多，内存也容纳不下时，就找个文件存放起来，后来发现其实存入文件的方式很不错，起码可以一直继续下去。也许这就是最佳的持久化策略了。

1.1.2　属性数据

在接触具体的项目开发任务时，更多的时候要接触到属性类型的数据，这些数据来源于对象的属性，当然这些属性之间还存在关系，例如本书的最后一章介绍的衣网的项目，其中的 Cloth，User，UserOrder，OrderItem 和 Image 对象都可以被抽象出来，而真

正需要保存的数据就是这些对象的属性。对于此类数据，似乎最好的持久化方式就是利用关系型数据库的数据表了，在存入数据表的过程中还可以借助于本书所介绍的 Hibernate 来提高开发的效率，或者直接使用 JDBC 来传递 SQL 语句。这两种方式比较相似，但是仍然存在着细微的差别。

1.1.3 XML 数据

现在很多网络上的应用程序都使用 XML 类型的数据，或者说是一种树状的数据，如何将这种数据持久化保存下来呢？一种方法是将 XML 数据拆分为属性数据，然后存储到关系型数据库中。另一种方式比较新，是将 XML 数据存储到 XML 数据库中，在 XML 数据库中存储这种数据时，数据的结构不会被改变，XML 数据在 XML 数据库中以文档对象形式保存，使用 XPath 和 XQuery 查询语言来查询 XML 数据中的节点数据，或者多个 XML 文档中的多个节点的数据。

相对于属性数据而言，XML 数据是很宽松的。首先，XML 数据的结构没有被确定下来，起码没有被定死为某种僵化的结构。其次，对于 XML 数据的各个节点来说，数据的类型并没有被确定下来，在 XML 中如果不加特殊的约束（例如使用 DTD 或者 Schema），XML 文档中的数据和结构可以有十分大的变化空间，在 XML 中以字符串形式存在的数据，很难确定它的类型是类似 int 的整数型还是类似 String 的文本类型。最后，XML 是可以被转换处理的，利用 XSLT 可以将 XML 转换成希望的结果。因此可以说对 XML 数据的约束是比较宽松的，XML 数据类型甚至可以用来模拟实现一些随机性质的数据类型。

1.2 持久化技术

对于完成数据的持久化工作而言，可以采用不同的技术方案，而目前除了文件保存的方式以外，比较成型的方式就要数关系型数据库和 XML 了。下面我们举一个简单的 BOOKS 示例来对比各种持久化技术。这个示例的内容很简单，就是实现对如下两本书的数据的持久化。两本书的信息用 XML 表示如下：

```
<BOOKS>
    <ITEM CATALOG="ORM">
    <TITLE>精通 Hibernate</TITLE>
        <PUBLISHER>电子工业出版社</PUBLISHER>
    <AUTHOR>刘洋</AUTHOR>
    <PRICE>8.0</PRICE>
    <QUANTITY>3</QUANTITY>
    </ITEM>
    <ITEM CATALOG="J2EE">
        <TITLE>精通 JBoss</TITLE>
        <PUBLISHER>电子工业出版社</PUBLISHER>
        <AUTHOR>刘洋,魏飞</AUTHOR>
        <PRICE>7.0</PRICE>
        <QUANTITY>5</QUANTITY>
```

```
    </ITEM>
</BOOKS>
```

1.2.1 JDBC

JDBC 配合 SQL 语句的技术方案，虽然只是针对关系型数据库而言的技术，但是这种搭配和 CMP 与 Hibernate 相比还是宽松的，因为那些技术都是基于 JDBC 和 SQL 语句的，它们不可能突破 JDBC 自身的限制，当然也就无法超越 JDBC 了，而且作为高一级别的技术存在，它们又产生了新的限制。

使用 JDBC 和 SQL 可以将 BOOKS 的数据存入 MySQL 数据库中，具体的代码如下：

```
//加载 MySQL 数据库驱动
Class.forName("org.gjt.mm.mysql.Driver");
//设置访问属性
Properties prop = new Properties();
prop.setProperty("user","root");
prop.setProperty("password","123456");
//打开数据库连接并连接到指定的 URL
Connection con = DriverManager.getConnection(
"jdbc:mysql://localhost:3306/hibernate",prop);
//编辑 SQL 数据库语句
Statement sm = con.createStatement();
String query = "insert into
book(id,title,publisher,author,price,quantity,category) values(NULL,'精通
Hibernate','电子工业出版社','刘洋','8.0','3','ORM');";
//执行 SQL 查询语句
int rs = sm.executeUpdate(query);
//关闭数据库连接
con.close();
```

使用这种 JDBC 程序需要在数据库中建立如图 1-1 所示的 book 数据表。

Columns | Indexes | Primary Key | Foreign Key | Preview | Row Count

id	title	publisher	author	price	quantity	category
1	精通 Hibernate	电子工业出版社	刘洋	8.0	3	ORM

图 1-1　book 数据表

1.2.2 Hibernate

如果利用 Hibernate 来实现 BOOKS 的持久化，则需要设计一个持久化的 Book 类，Book 的代码如下：

```
package com.weportal.books;
public class Book {
    private Long id;
    private String title;
```

```
    private String publisher;
    private String author;
    private double price;
    private int quantity;
private String category;
//对应的 get 和 set 方法
}
```

同时，为 Book 设计一个 Hibernate 映射文件，即 Book.hbm.xml，其内容如下：

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="com.weportal.books">
    <class name="Book">
        <id name="id">
            <generator class="native"/>
        </id>
        <property name="title"/>
        <property name="publisher"/>
        <property name="author"/>
        <property name="price"/>
        <property name="quantity"/>
        <property name="category"/>
    </class>
</hibernate-mapping>
```

1.2.3 XML 数据库

如果使用 XML 数据库来完成 BOOKS 示例，则可以直接将 BOOKS 导入 Apache 的 Xindice 数据库中。本书以 Xindece 1.0 为例，首先启动 Xindece，然后在 Xindece 添加一个名为 books 的 Collection，再使用下面的代码将 BOOKS 直接导入这个 Collection。

```
String fname = "xml/books.xml";
String driver = "org.apache.xindice.client.xmldb.DatabaseImpl";
Class c = Class.forName(driver);
Database database = (Database) c.newInstance();
DatabaseManager.registerDatabase(database);
Collection col = DatabaseManager.getCollection("xmldb:xindice:///db/books");
String data = readFileFromDisk(fname);
XMLResource document = (XMLResource)
col.createResource("book2","XMLResource");
document.setContent(data);
col.storeResource(document);
```

其中的“xml/books.xml”文件就是包含 BOOKS 的 XML 文件，导入 Xindice 之后以 book2 作为资源名称，利用这个资源名称还可以从 Xindice 中取出 BOOKS。

1.3 数据层的设计模式

基于上一节介绍的 3 种持久化技术，可以实现 BOOKS 的持久化任务。将这种数据

持久化策略融入整体系统的设计时，可以采用不同的设计模式来完成。对于 Hibernate 来说可以采用 DAO 的设计模式，对于 XML 数据来说可以使用过滤器的设计模式。本节下面的部分将分别举例说明这两种设计模式。

1.3.1 DAO

在数据层中可以使用 DAO 模式，即数据访问模式来实现数据持久化的任务，通过一个持久化对象来存储数据，将业务数据设置到持久化类的属性中，然后使用另一个 DAO 对象操纵这个持久化类，在 DAO 对象的方法中实现业务方法，例如下面的 BookDAO。

```
package com.weportal.books;

import org.apache.log4j.PropertyConfigurator;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.cfg.Configuration;

public class BookDAO {

    public void add(Book book){
        PropertyConfigurator.configure("log4j.Properties");
        Configuration cfg = new Configuration();  cfg.configure();
        SessionFactory sf = cfg.buildSessionFactory();
        Session s = sf.openSession();
        Transaction t = s.beginTransaction();
        s.save(book);
        s.flush();
        t.commit();
    }
}
```

DAO 的设计模式很适合于将数据表示为 Java 对象的方式，通过操纵 Java 持久化对象来改变属性数据。利用 DAO 来实现业务方法，是符合面向对象设计原则的合理模式，适合与实现 MVC 架构的 Struts 配合，形成一个稳健的 OO 系统。

1.3.2 过滤器模式

当数据以 XML 形式存在时（例如这里使用的 BOOKS 数据），如果仍然坚持使用 DAO 模式来完成业务功能的话，就需要从 XML 数据产生 Java 对象。这个过程可以通过 Java 的 XML 编程接口来完成，也可以使用 Java 数据绑定的方式从 XML 直接产生 Java 对象，但是这种做法不适合于 XML 数据。处理 XML 数据可以使用过滤器模式，通过转换和过滤来实现，例如下面的 XMLTransform 所实现的 XSLT 过程。

```
package com.weportal.xslt;

import javax.xml.parsers.SAXParser;
```