Broadview®
www.broadview.com.cn

以MyBatis 3.4源码为基础，详细分析MyBatis的架构设计和实现细节
深入剖析MyBatis的基础层、核心层中各个模块的功能和实现细节
介绍以插件方式扩展MyBatis的原理以及与Spring集成的原理

MyBatis技术内幕

徐郡明 编著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

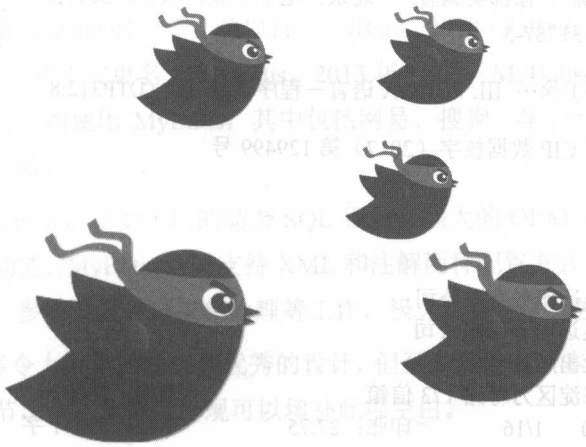
作者简介

徐郡明，武汉大学硕士，目前就职于航天科技集团旗下某研究所，主要负责政企云平台基础架构的设计和研发工作，关注多项Java开源技术的发展。

Broadview®
www.broadview.com.cn

MyBatis技术内幕

徐郡明 编著



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

内 容 简 介

本书以 MyBatis 3.4 为基础，针对 MyBatis 的架构设计和实现细节进行了详细分析，其中穿插介绍了 MyBatis 源码中涉及的基础知识、设计模式以及笔者自己在实践中的思考。本书共 4 章，从 MyBatis 快速入门开始，逐步分析了 MyBatis 的整体架构以及核心概念，对 MyBatis 的基础支持层、核心处理层中各个模块的功能和实现细节进行了深入的剖析。除此之外，还分析了 MyBatis 插件的应用场景和实现原理，介绍了 MyBatis 与 Spring 集成开发的示例和原理，以及一些实践中的小技巧和小工具的使用方法。

本书旨在为读者理解 MyBatis 的设计原理、阅读 MyBatis 源码、扩展 MyBatis 功能提供帮助和指导，让读者更加深入地了解 MyBatis 的运行原理、设计理念。希望本书能够帮助读者全面提升自身的技术能力，让读者在设计业务系统时，可以参考 MyBatis 的优秀设计，更好地应用 MyBatis。

图书在版编目（CIP）数据

MyBatis 技术内幕 / 徐郡明编著. —北京：电子工业出版社，2017.7
ISBN 978-7-121-31787-3

Ⅰ. ①M… Ⅱ. ①徐… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2017）第 129499 号

责任编辑：陈晓猛
印　　刷：三河市良远印务有限公司
装　　订：三河市良远印务有限公司
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：27.75　　字数：532 千字
版　　次：2017 年 7 月第 1 版
印　　次：2017 年 7 月第 1 次印刷
定　　价：79.00 元

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）88254888，88258888。

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。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

前　言

面向对象程序设计是企业级开发常用的设计方式，在实践中常用的编程语言大多都是面向对象的编程语言。而在实际生产环境中常用的数据库产品，如 MySQL、Oracle 等，都是关系型数据库。虽然 NoSQL 数据库在最近一段时间有飞速的发展，但是关系型数据库凭借多年的发展和技术积累，依然占据着市场的主导地位。

MyBatis 作为一个优秀的 Java 持久化框架，可以帮助程序员完成 ORM 映射、查询缓存等常用功能。MyBatis 以其高性能、易优化、易维护、可扩展等优点，受到越来越多的开发人员的青睐，也有越来越多的设计人员开始将 MyBatis 作为其首选的 Java 持久化框架。

MyBatis 的前身是 Apache 的一个开源项目——iBatis，2010 年 iBatis 项目由 Apache 基金会迁移到了 Google Code，并正式更名为 MyBatis。2013 年 11 月，MyBatis 迁移到 Github。目前，越来越多的互联网公司开始使用 MyBatis，其中包括网易、搜狗、华为等，依赖 MyBatis 搭建的创业项目更是数不胜数。

MyBatis 的亮点有很多，比如灵活的动态 SQL 语句、强大的 ORM 映射功能等，同时还提供了二级缓存等常用功能。MyBatis 同时支持 XML 和注解两种配置方式，帮助程序员屏蔽了近乎所有的 JDBC 代码、参数设置、结果集处理等工作，极大地提升了开发效率。

MyBatis 中有很多令人称赞的功能和优秀的设计，但至今还没有一本书籍深入剖析 MyBatis 的内部设计和实现细节，希望本书的出现可以填补此项空白。

本书以 MyBatis 3.4 为基础，针对 MyBatis 的架构设计和实现细节进行了详细分析，其中穿插介绍了 MyBatis 源码中涉及的基础知识、设计模式以及笔者自己在实践中的思考。除此之外，还分析了 MyBatis 插件的应用场景和实现原理，介绍了 MyBatis 与 Spring 集成开发的示例和原理，以及一些实践中的小技巧和小工具的使用。

如何阅读本书

由于篇幅限制，本书并没有详细介绍 Java 的基础知识，但为了便于读者理解 MyBatis 的设计思想和实现细节，笔者介绍了一些必需且重要的基础内容，例如涉及的多种设计模式。

本书共 4 章，它们互相之间的联系并不是很强，读者可以从头开始阅读，也可以选择自己感兴趣的章节进行学习。

第 1 章是 MyBatis 的快速入门，其中介绍了 MyBatis 出现的背景、与其他 Java 持久化框架的比较以及 MyBatis 的入门示例。之后介绍了 MyBatis 的整体架构，并简述了 MyBatis 中各个模块的基本功能。

第 2 章介绍 MyBatis 基础支持层中各个模块的功能，其中包括数据源模块、事务管理模块、缓存模块、binding 模块、反射模块、类型转换模块、日志模块、资源加载模块和解析器模块。这些模块相对独立，读者在实践中如果遇到类似的需求，可以直接参考 MyBatis 的实现。

第 3 章介绍 MyBatis 核心处理层的主要功能，其中包括 MyBatis 初始化过程、动态 SQL 的解析过程、结果集的映射原理、SQL 语句的参数绑定、KeyGenerator、StatementHandler 以及 Executor 等组件的实现原理。同时，还介绍了 MyBatis 接口层的设计原理。

第 4 章介绍 MyBatis 插件的编写和配置方式、运行原理以及常见的应用场景，并分析了笔者在实践中使用的分页插件和分表插件的具体实现。之后，介绍了 MyBatis 与 Spring 集成开发的相关内容，搭建了 Spring 4.3、MyBatis 3.4、Spring MVC 的集成开发环境，剖析了 MyBatis-Spring 中核心组件的实现原理。最后介绍了一些在使用 MyBatis 时用到的小技巧和一些小工具的使用方法。

在本书中，除了介绍 MyBatis 的实现细节，还介绍了其中涉及的设计模式，可以帮助读者了解 MyBatis 源码背后的设计思想。

如果读者在阅读本书的过程中，发现任何不妥之处，请将您宝贵的意见和建议发送到邮箱 xxxlxy2008@163.com，也欢迎读者朋友通过此邮箱与笔者进行交流。

致谢

感谢电子工业出版社博文视点的陈晓猛老师，是您的辛勤工作让本书的出版成为可能。同时还要感谢许多我不知道名字的幕后工作人员为本书付出的努力。

感谢朱碧颖、逄志强、杨俊灵、李全才、曾君实等朋友在百忙之中抽出时间对本书进行审阅和推荐。感谢米秀明、曾天宁、葛彬、杨杉、文静宇、刘浩、杨鹏林、路恒、藤少广等同事，

帮助我解决工作中的困难。

这里特别感谢王鲁老师，在软件架构、设计模式等方面对我的指导。

感谢冯玉玉、李成伟，是你们让写作的过程变得妙趣横生，是你们让我更加积极、自信，也是你们的鼓励让我完成了本书的写作。

最后，特别感谢我的母亲大人，谢谢您默默为我做出的牺牲和付出，您是我永远的女神。

徐郡明

读者服务

轻松注册成为博文视点社区用户（www.broadview.com.cn），扫码直达本书页面。

- **下载资源**：本书如提供示例代码及资源文件，均可在下载资源处下载。
- **提交勘误**：您对书中内容的修改意见可在提交勘误处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方读者评论处留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：http://www.broadview.com.cn/31787

专家推荐

（排名不分先后）

《MyBatis 技术内幕》深入浅出地讲解了 MyBatis 的底层原理，清晰的写作思路、翔实的内容让我受益匪浅，这是一本优秀的进阶书籍。

——中量财富（北京）策略研发中心总经理　朱碧颖

MyBatis 现在已经是 Java 企业级开发中的主流框架之一。《MyBatis 技术内幕》全面地剖析了 MyBatis 的架构设计，同时作者也分享了很多实践经验，值得一读。

——小米科技高级研发工程师　逄志强

《MyBatis 技术内幕》深入分析了 MyBatis 的设计思想，帮助读者了解 MyBatis 的运行原理，作者分析源码时思路清晰、讲解到位，是一本非常难得的好书。

——华为高级研发工程师　杨俊灵

《MyBatis 技术内幕》展示了 MyBatis 框架的全景，其中特别喜欢作者将设计模式的讲解与 MyBatis 源码剖析相结合的写作方式，让我们不仅了解了设计模式的概念，还学习到了这些模式的最佳实践。

——搜狗高级开发工程师　李全才

我特别喜欢著名作家侯捷说过的一句话：“源码面前，了无秘密”。《MyBatis 技术内幕》可以让读者深入透彻地理解 MyBatis 内部结构。对于 Java 程序员来说，是一本不可错过的佳作。

——微医集团 Java 高级研发工程师　曾君实

目　录

第 1 章 MyBatis 快速入门

1.1 ORM 简介

面向对象程序设计是企业级开发常用的设计方式，我们在实践中常用的编程语言，如 Java、.Net、C++等，都是面向对象的编程语言。在实际生产环境中常用的数据库产品，如 MySQL、Oracle 等，则都是关系型数据库。虽然 NoSQL 数据库，如 HBase、MongoDB、Couchbase 等，在最近一段时间有了飞速的发展，也有一部分互联网应用开始尝试使用 NoSQL 数据库管理其部分数据，但是关系型数据库凭借多年的发展和技术积累，以及众多成功的案例等优势，依然占据着市场的主要地位。

在系统开发过程中，开发人员需要使用面向对象的思维实现业务逻辑，但设计数据库表或是操作数据库记录时，则需要通过关系型的思维方式考虑问题。应用程序与关系型数据库之间进行交互时，数据在对象和关系结构中的表、列、字段等之间进行转换。

JDBC 是 Java 与数据库交互的统一 API，实际上它分为两组 API，一组是面向 Java 应用程序开发人员的 API，另一组是面向数据库驱动程序开发人员的 API。前者是一个标准的 Java API 且独立于各个厂家的数据库实现，后者则是数据库驱动程序开发人员用于编写数据库驱动，是前者的底层支持，一般与具体的数据库产品相关。

在实际开发 Java 系统时，我们可以通过 JDBC 完成多种数据库操作。这里以传统 JDBC 编程中的查询操作为例进行说明，其主要步骤如下：

（1）注册数据库驱动类，明确指定数据库 URL 地址、数据库用户名、密码等连接信息。

（2）通过 DriverManager 打开数据库连接。

（3）通过数据库连接创建 Statement 对象。

（4）通过 Statement 对象执行 SQL 语句，得到 ResultSet 对象。

（5）通过 ResultSet 读取数据，并将数据转换成 JavaBean 对象。

（6）关闭 ResultSet、Statement 对象以及数据库连接，释放相关资源。

上述步骤 1～步骤 4 以及步骤 6 在每次查询操作中都会出现，在保存、更新、删除等其他数据库操作中也有类似的重复性代码。在实践中，为了提高代码的可维护性，可以将上述重复性代码封装到一个类似 DBUtils 的工具类中。步骤 5 中完成了关系模型到对象模型的转换，要使用比较通用的方式封装这种复杂的转换是比较困难的。

为了解决该问题，ORM（Object Relational Mapping，对象-关系映射）框架应运而生。如图 1-1 所示，ORM 框架的主要功能就是根据映射配置文件，完成数据在对象模型与关系模型之间的映射，同时也屏蔽了上述重复的代码，只暴露简单的 API 供开发人员使用。

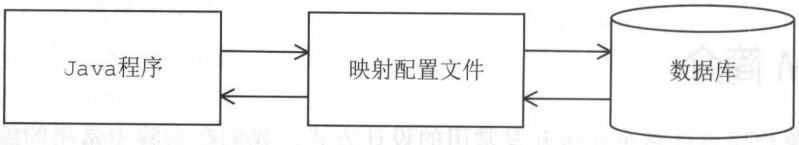


图 1-1

另外，实际生产环境中对系统的性能是有一定要求的，数据库作为系统中比较珍贵的资源，极易成为整个系统的性能瓶颈，所以我们不能像上述 JDBC 操作那样简单粗暴地直接访问数据库、直接关闭数据库连接。应用程序一般需要通过集成缓存、数据源、数据库连接池等组件进行优化，如果没有 ORM 框架的存在，就要求开发人员熟悉相关组件的 API 并手动编写集成相关的代码，这就提高了开发难度并延长了开发周期。

很多 ORM 框架都提供了集成第三方缓存、第三方数据源等组件的接口，而且这些接口都是业界统一的，开发和运维人员可以通过简单的配置完成第三方组件的集成。当系统需要更换第三方组件时，只要选择支持该接口的组件并更新配置即可，这不仅提高了开发效率，而且提高了系统的可维护性。

最后，建议读者在开发大中型项目时，优先考虑使用 ORM 框架，并根据下文的介绍选择合适的 ORM 框架。之所以这么说，是因为笔者在设计某些项目时，未使用 ORM 框架，到项目的中后期时为了便于项目的扩展和维护，使用各种设计模式等相关知识对程序的 DAO 层（Data Access Object，数据访问对象）进行了重构，最后得到一个不完善的、类似于 ORM 框架的设计。这也算是笔者实践中的一个教训，希望对读者有一定参考价值。

1.2 常见持久化框架

Hibernate

Hibernate 是一款 Java 世界中最著名的 ORM 框架之一，笔者撰稿时 Hibernate 的最新版本是 5.2 版本。作为一个老牌的 ORM 框架，Hibernate 经受住了 Java EE 企业级应用的考验，替代了复杂的 Java EE 中 EJB 解决方案，一度成为 Java ORM 领域的首选框架。

Hibernate 通过 hbm.xml 映射文件维护 Java 类与数据库表的映射关系。通过 Hibernate 的映射，Java 开发人员可以用看待 Java 对象的角度去看待数据库表中的数据行。数据库中所有的表通过 hbm.xml 配置文件映射之后，都对应一个 Java 类，表中的每一行数据在运行过程中会被映射成相应的 Java 对象。在 Java 对象之间存在一对多、一对一、多对多等复杂的层次关系，Hibernate 的 hbm.xml 映射文件也可以维护这种层次关系，并将这种关系与数据库中的外键、关联表等进行映射，这也就是所谓的“关联映射”。

例如，一个用户（User）可以创建多个订单（Order），而一个订单（Order）只属于一个用户，两者之间存在一对多的关系。在 Java 代码中可以在 User 类中添加一个 List<Order>类型的字段来维护这种一对多关系，在数据库中可以在订单表（t_order）中添加一个 user_id 列作为外键，指向用户表（t_user）的主键 id，从而维护这种一对多的关系，如图 1-2 所示。

在 Hibernate 中，可以通过如下 User.hbm.xml 配置文件将这两种关系进行映射。

```
<hibernate-mapping>
    <!-- 表和类之间的映射 -->
    <class name="com.xxx.User" table="t_user">
        <!-- 主键映射 -->
        <id name="id" column="id"/>
        <!-- 属性映射 -->
        <property name="name" column="name"/>
        <!-- 表之间关系映射 -->
        <set name="orders" cascade="save-update,delete">
            <key column="user_id"/>
            <one-to-many class="com.xxx.Order"/>
        </set>
    </class>
</hibernate-mapping>
```

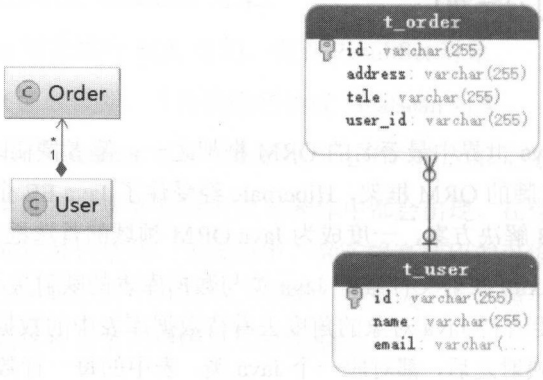


图 1-2

如果是双向关联，则在 Order 中添加 User 类型的字段指向关联的 User 对象，并使用相应的 Order.hbm.xml 配置文件进行配置，示例如下：

```
<hibernate-mapping>
    <!-- 表和类之间的映射 -->
    <class name="com.xxx.Order" table="t_order">
        <!-- 主键映射 -->
        <id name="id" column="id"/>
        <!-- 属性映射 -->
        <property name="address" column="address"/>
        <property name="tele" column="tele"/>
        <!-- 表之间关系映射 -->
        <many-to-one name="user" column="user_id"></many-to-one>
    </class>
</hibernate-mapping>
```

一对一、多对多等关联映射与上述配置类似，这里就不再赘述，感兴趣的读者可以参考 Hibernate 的相关资料进行学习。

Hibernate 除了能够实现对象模型与关系模型的映射，还可以帮助开发人员屏蔽不同数据库产品中 SQL 语句的细微差异。现在不同的关系型数据库产品对 SQL 标准的支持不尽相同，这就会出现同一条 SQL 语句在 MySQL 上可以正常执行，而在 Oracle 数据库上执行报错的情况。Hibernate 封装了数据库层面的全部操作，开发人员不再需要直接编写 SQL 语句，只需要使用 Hibernate 提供的简单易懂的 API 即可完成数据库操作。例如，Hibernate 提供的 Criteria 是一个完全面向对象、可扩展的条件查询 API，使用它查询数据库时完全不需要考虑数据库底层如何

实现、SQL 语句如何编写。除了 Criteria，Hibernate 还提供了一种称为 HQL（Hibernate Query Language）的语言，从语句的结构上来看，HQL 语句与 SQL 语句十分类似，但它是一种面向对象的查询语言。对于复杂的数据库查询，开发人员可以按照面向对象的思维方式编写 HQL 实现，Hibernate 会根据实际配置的数据库方言，将 HQL 语句生成对应的 SQL 语句。Hibernate 通过其简洁的 API 以及统一的 HQL 语句，帮助上层程序屏蔽掉底层数据库的差异，增强了程序的可移植性。

另外，Hibernate 的 API 没有侵入性，业务逻辑不需要继承 Hibernate 的任何接口。Hibernate 默认提供了一级缓存和二级缓存，这有利于提高系统的性能，降低数据库压力。Hibernate 还有其他的特性和优点，例如，支持透明的持久化、延迟加载、由对象模型自动生成数据库表等，感兴趣的读者可以查阅 Hibernate 的相关资料进行学习。

但是，Hibernate 并不是一颗万能药。数据库本身有自己的组织方式，并不是数据库中所有的概念都能在面向对象的世界中找到合适的映射，例如，索引、存储过程、函数等，尤其是索引，它对数据库查询的性能帮助很大，适当优化 SQL 语句，选择使用合适的索引会提高整个查询的速度。但是，我们很难修改 Hibernate 生成的 SQL 语句，当数据量比较大、数据库结构比较复杂时，Hibernate 生成 SQL 语句会非常复杂，而且要让生成的 SQL 语句使用正确的索引也比较困难，这就会导致出现大量慢查询的情况。在有些大数据量、高并发、低延迟的场景下，Hibernate 并不是特别适合。最后，Hibernate 对批处理的支持并不是很友好，这也会影响部分性能。后来出现了 iBatis（Mybatis 的前身）这种半自动化的映射方式来解决性能问题。

JPA

JPA（Java Persistence API）是 EJB 3.0 中持久化部分的规范，但它可以脱离 EJB 的体系单独作为一个持久化规范进行使用。Gavin King 作为 Hibernate 创始人，同时也参与了 JPA 规范的编写，所以在 JPA 规范中可以看到很多与 Hibernate 类似的概念和设计。这里需要读者了解的是，JPA 仅仅是一个持久化的规范，它并没有提供具体的实现。其他持久化厂商会提供 JPA 规范的具体实现，例如，Hibernate、EclipseLink 等都提供了 JPA 规范的具体实现。JPA 规范的愿景很美好，但是并没有得到很好的发展，现在在实践中的出场率也不是很高。如果读者对 JPA 感兴趣，可以查阅相关资料进行学习，这里就不再做过多介绍了。

Spring JDBC

严格来说，Spring JDBC 并不能算是一个 ORM 框架，它仅仅是使用模板方式对原生 JDBC 进行了一层非常薄的封装。使用 Spring JDBC 可以帮助开发人员屏蔽创建数据库连接对、Statement 对象、异常处理以及事务管理的重复性代码，提高开发效率。

Spring JDBC 中没有映射文件、对象查询语言、缓存等概念，而是直接执行原生 SQL 语句。Spring JDBC 中提供了多种 Template 类，可以将对象中的属性映射成 SQL 语句中绑定的参数，