Broadview
www.broadview.com.cn

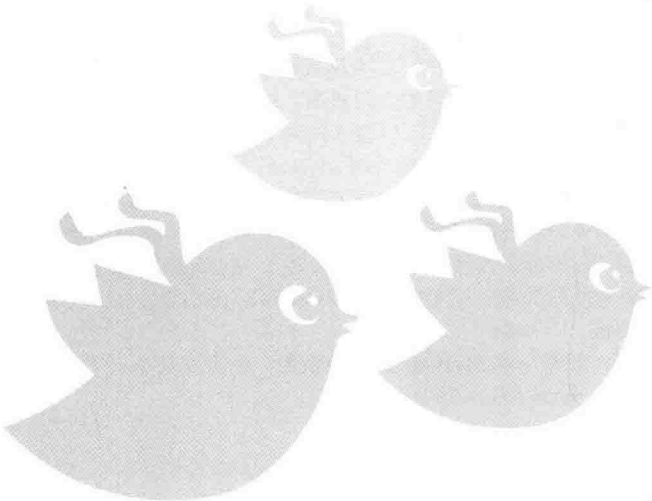
手把手教你整合开发MyBatis-Spring项目

深入浅出 MyBatis 技术原理与实战

杨开振／著

基于官方API的完全解读，开MyBatis应用之先河
详细阐述MyBatis内部运行原理和插件开发

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



深入浅出 MyBatis 技术原理与实战

杨开振 / 著

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

内 容 提 要

随着大数据时代的到来，Java 持久层框架 MyBatis 已经成为越来越多企业的选择。遗憾的是，时至今日国内依然没有一本讨论 MyBatis 的书，这增加了初学者的学习难度，初学者往往只能基于零星的案例来学习 MyBatis，无法系统地掌握 MyBatis，更不用说精通了。《深入浅出 MyBatis 技术原理与实战》是笔者通过大量实践和研究源码后创作而成的，是国内第一本系统介绍 MyBatis 的著作。

本书分为 3 个部分，依次介绍了 MyBatis 的基础应用、原理及插件开发、实践应用，使读者能够由浅入深、循序渐进地掌握 MyBatis 技术。首先，本书在官方 API 的基础上完善了许多重要的论述和实例，并且给出了实操建议，帮助读者正确掌握 MyBatis。其次，本书详细讲述了 MyBatis 的内部运行原理，并全面讨论了插件的开发。最后，本着学以致用的原则，笔者阐述了 MyBatis-Spring 项目和一些 MyBatis 开发常见的实例，使读者能够学得会，用得好。

本书不是一本味同嚼蜡的理论专著，而是一本 MyBatis 的实践指南，无论你是 Java 程序员、MyBatis 开发者，还是 Java 持久层框架的研究者，你都能从本书中收获知识。

图书在版编目（CIP）数据

深入浅出 MyBatis 技术原理与实战 / 杨开振著. —北京：电子工业出版社，2016.9
ISBN 978-7-121-29594-2

Ⅰ. ①深… Ⅱ. ①杨… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2016）第 183295 号

责任编辑：徐津平
印　　刷：中国电影出版社印刷厂
装　　订：三河市华成印务有限公司
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：16.75　字数：310 千字
版　　次：2016 年 9 月第 1 版
印　　次：2016 年 10 月第 2 次印刷
定　　价：69.00 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

随着手机、平板电脑等移动终端的广泛应用，移动互联网时代已经到来。在这个时代里，构建一个高效的平台并提供服务是移动互联网的基础，在众多的网站服务中，使用 Java 构建网站的不在少数。移动互联网的特点是大数据、高并发，对服务器往往要求分布式、高性能、高灵活等，而传统模式的 Java 数据库编程框架已经不再适用了。在这样的背景下，一个 Java 的持久框架 MyBatis 走入了我们的世界，它以封装少、高性能、可优化、维护简易等优点成为了目前 Java 移动互联网网站服务的首选持久框架，它特别适合分布式和大数据网络数据库的编程。

本书主要讲解了 MyBatis 的应用。从目前的情况来看，国内图书市场上没有介绍 MyBatis 的书籍，有的只是官方的 API 和少数的几篇博客文章，国外图书市场上的这类书籍也是凤毛麟角，这使得系统学习 MyBatis 困难重重。官方的 API 只是简单介绍了 MyBatis 有些什么功能和一些基本的使用方法，没有告诉我们如何用好，其中原理是什么，需要注意哪些问题，这显然是不够的。有些博客虽然讲解得比较深入，但是内容支离破碎，没有形成一个完整的知识体系，不易于初学者对 MyBatis 进行系统学习。随着移动互联网应用的兴起，系统掌握 MyBatis 编程技巧已经成了用 Java 构建移动互联网网站的必要条件。为了顺应时代的要求，笔者写下了这本书，以期为广大需要掌握 MyBatis 的开发者提供学习和参考的资料。

阅读本书要求开发人员拥有 Java 语言基础和 JDBC 基础知识，对数据库也要掌握入门知识，最好能够掌握常用的设计模式，因为在介绍 MyBatis 构造时，常常涉及设计模式，尤其是第 6 章和第 7 章的内容。

本书以讲解 MyBatis 基础运用和原理为主，所以适合初级到中高级开发人员阅读。

本书分为三大部分。

第一部分是 MyBatis 基础应用，主要介绍如何高效地使用 MyBatis。

第 1 章：MyBatis 的内容简介，告诉读者 MyBatis 是什么，在何种场景下使用它。

第 2 章：主要介绍 MyBatis 的基础模块及其生命周期，并给出实例。

第 3 章：主要介绍 MyBatis 配置的主要含义和内容。

第 4 章：介绍 MyBatis 映射器的主要元素及其使用方法。

第 5 章：介绍动态 SQL，助你轻松应对大部分的 SQL 场景。

第二部分是 MyBatis 原理，我们将深入源码去理解 MyBatis 的内部运行原理以及插件的开发方法和技巧。

第 6 章：介绍 MyBatis 的解析和运行原理，我们将了解到 SqlSession 的构建方法，以及其四大对象是如何工作的。

第 7 章：在第 6 章的基础上着重介绍 MyBatis 的插件，这里我们将学习插件的设计原理，以及开发方法和注意的要点。

第三部分是 MyBatis 的实战应用，主要讲解 MyBatis 的一些实用的场景。

第 8 章：介绍 MyBatis-Spring，主要讲解如何在 Spring 项目中集成 MyBatis 应用，帮助读者在 Spring 的环境中顺利使用 MyBatis。

第 9 章：介绍 MyBatis 的实用场景，精选一批典型且又常用的场景。详细解析每一个场景下，开发人员需要注意避免的一些错误和性能上的损失。

MyBatis 源于 2002 年的 iBatis 项目，至今 MyBatis 中依然有许多 iBatis 的痕迹。本书默认使用 MyBatis 的版本是 3.3.0，使用 MyBatis-Spring 的版本是 1.2.3。而历史上的 iBatis 的书籍已经跟不上技术发展的步伐，于是笔者通过自己的努力和实践，在研究 MyBatis 源码的基础上，写作本书。从本书中既能学习如何使用 MyBatis，也可以学习 MyBatis 的原理和应用，为国内的 MyBatis 开发者提供一条系统掌握 MyBatis 编程技巧的捷径，当然读者也可以把本书作为工具书参考。在实际操作中，MyBatis 往往是结合 Spring 使用的，于是本书花费了一些篇幅讲解 MyBatis-Spring 技术，笔者也会略略提到 Spring 项目的内容，以便更好地论述它们。最后笔者还将讲解一些使用频率高、参考价值大的场景，使读者能熟练掌握 MyBatis 的开发。

本书坚持实用原则，对于一些使用频率低的技术并没有提及太多，比如注解 SQL、SQL 构造器等内容，使用这些内容，会造成代码的可读性下降。

感谢我的公司为我提供真实的使用 MyBatis 的环境，所有的程序代码都经过了调试。感谢我的姐姐杨坚，她参与编写并通篇审校了本书，润色了那些晦涩的句子。同时也感谢电子工业出版社的编辑们，尤其是汪达文的全程跟进。没有他们的辛苦付出，就没有本书的成功出版。在出版本书的欣喜之余，也伴着战战兢兢，因为笔者才疏学浅，很多东西都是从对源码的理解和实际操作中获得的，因此书中难免有疏漏之处，或有不能让读者满意的地方。如果有困惑，读者可以发邮件到我的邮箱：ykzhen2013@163.com，也可以在我的博客（http://blog.csdn.net/ykzhen2015）中和我讨论，还望各位同行不吝赐教。

杨开振

2016 年 7 月

目　录

第 1 章

MyBatis 简介

本章主要介绍了 Java ORM 的来源和历史，同时分别介绍了 JDBC、Hibernate 和 MyBatis 三种访问数据库的方法，在分析它们优缺点的基础上，比较它们之间的区别和适用的场景。

1.1 传统的 JDBC 编程

Java 程序都是通过 JDBC（Java Data Base Connectivity）连接数据库的，这样我们就可以通过 SQL 对数据库编程。JDBC 是由 SUN 公司（SUN 公司后被 Oracle 公司收购）提出的一系列规范，但是它只定义了接口规范，而具体的实现是交由各个数据库厂商去实现的，因为每个数据库都有其特殊性，这些是 Java 规范没有办法确定的，所以 JDBC 就是一种典型的桥接模式。

传统的 JDBC 编程的使用给我们带来了连接数据库的功能，但是也引发了巨大的问题。代码清单 1-1 是用 JDBC 编程的一个例子。我们将从 MySQL 数据库中查询一个角色的名称，假设我们已经知道角色编号为 1。

代码清单 1-1：JdbcExample.java

```
public class JdbcExample {
    private Connection getConnection() {
        Connection connection = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            String url = "jdbc:mysql://localhost:3306/mybatis?zeroDateTime
Behavior=convertToNull";
            String user = "root";
```

```
            String password = "learn";
            connection = DriverManager.getConnection(url, user, password);
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(JdbcExample.class.getName()).log(Level.SEVERE,
null, ex);
            return null;
        }
        return connection;
    }
    public Role getRole(Long id) {
        Connection connection  = getConnection();
        PreparedStatement ps = null;
        ResultSet rs = null;
        try {
            ps = connection.prepareStatement("select id, role_name, note from
t_role where id = ?");
            ps.setLong(1, id);
            rs = ps.executeQuery();
            while(rs.next()) {
                Long roleId = rs.getLong("id");
                String userName = rs.getString("role_name");
                String note = rs.getString("note");
                Role role = new Role();
                role.setId(id);
                role.setRoleName(userName);
                role.setNote(note);
                return role;
            }
        } catch (SQLException ex) {
            Logger.getLogger(JdbcExample.class.getName()).log(Level.SEVERE,
null, ex);
        } finally {
            this.close(rs, ps, connection);
        }
        return null;
    }
    private void close(ResultSet rs, Statement stmt, Connection connection)
{
        try {
            if (rs != null && !rs.isClosed()) {
                rs.close();
```

```
            }
        } catch (SQLException ex) {
            Logger.getLogger(JdbcExample.class.getName()).log(Level.SEVERE,
null, ex);
        }
        try {
            if (stmt != null && !stmt.isClosed()) {
                stmt.close();
            }
        } catch (SQLException ex) {
            Logger.getLogger(JdbcExample.class.getName()).log(Level.SEVERE,
null, ex);
        }
        try {
            if (connection != null && !connection.isClosed()) {
                connection.close();
            }
        } catch (SQLException ex) {
            Logger.getLogger(JdbcExample.class.getName()).log(Level.SEVERE,
null, ex);
        }
    }
    public static void main(String[] args) {
        JdbcExample example = new JdbcExample();
        Role role = example.getRole(1L);
        System.err.println("role_name => " + role.getRoleName());
    }
}
```

从代码中我们可以看出整个过程大致分为以下几步：

- 使用 JDBC 编程需要连接数据库，注册驱动和数据库信息。
- 操作 Connection，打开 Statement 对象。
- 通过 Statement 执行 SQL，返回结果到 ResultSet 对象。
- 使用 ResultSet 读取数据，然后通过代码转化为具体的 POJO 对象。
- 关闭数据库相关资源。

使用传统的 JDBC 方式存在一些弊端。其一，工作量相对较大。我们需要先连接，然后处理 JDBC 底层事务，处理数据类型。我们还需要操作 Connection 对象、Statement 对象

和 ResultSet 对象去拿到数据，并准确关闭它们。其二，我们要对 JDBC 编程可能产生的异常进行捕捉处理并正确关闭资源。对于一个简单的 SQL 在 JDBC 中尚且如此复杂，何况是更为复杂的应用呢？很快这种模式就被一些新的方法取代，于是 ORM 模型就出现了。不过所有的 ORM 模型都是基于 JDBC 进行封装的，不同的 ORM 模型对 JDBC 封装的强度是不一样的。

1.2 ORM 模型

由于 JDBC 存在的缺陷，在实际工作中我们很少使用 JDBC 进行编程，于是提出了对象关系映射（Object Relational Mapping，简称 ORM，或者 O/RM，或者 O/R mapping）。那什么是 ORM 模型呢？

简单地说，ORM 模型就是数据库的表和简单 Java 对象（Plain Ordinary Java Object，简称 POJO）的映射关系模型，它主要解决数据库数据和 POJO 对象的相互映射。我们通过这层映射关系就可以简单迅速地把数据库表的数据转化为 POJO，以便程序员更加容易理解和应用 Java 程序，如图 1-1 所示。

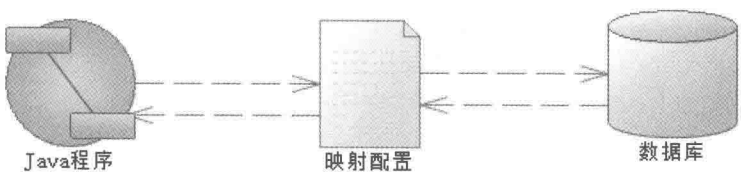


图 1-1　ORM 映射模型

有了 ORM 模型，在大部分情况下，程序员只需要了解 Java 应用而无需对数据库相关知识深入了解，便可以写出通俗易懂的程序。此外，ORM 模型提供了统一的规则使得数据库的数据通过配置便可轻易映射到 POJO 上。

1.3 Hibernate

最初 SUN 公司推出了 Java EE 服务器端组件模型（EJB），但是由于 EJB 配置复杂，且适用范围较小，于是很快就被淘汰了。与 EJB 的失败伴随而来的是另外一个框架的应运而

生。它就是从诞生至今都十分流行的 Hibernate。

Hibernate 一问世就成了 Java 世界首选的 ORM 模型，它是建立在 POJO 和数据库表模型的直接映射关系上的。

Hibernate 是建立在若干 POJO 通过 XML 映射文件（或注解）提供的规则映射到数据库表上的。换句话说，我们可以通过 POJO 直接操作数据库的数据。它提供的是一种全表映射的模型。如图 1-2 所示是 Hibernate 模型的开发过程。相对而言，Hibernate 对 JDBC 的封装程度还是比较高的，我们已经不需要编写 SQL 语言（Structured Query Language），只要使用 HQL 语言（Hibernate Query Langurage）就可以了。

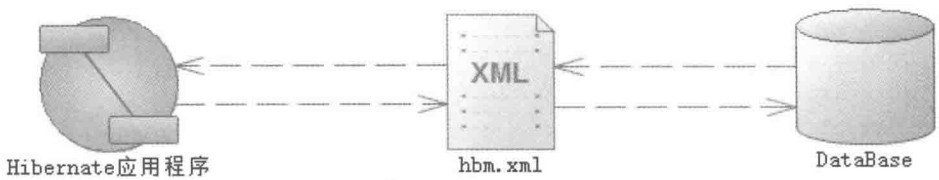


图 1-2　Hibernate 模型的开发过程

首先我们需要提供 hbm.xml 文件，制定映射规则。下面以开发角色类为例进行讲解，如代码清单 1-2 所示。

代码清单 1-2：TRole.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate Mapping DTD
3.0//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<!-- Generated 2015-12-12 22:50:58 by Hibernate Tools 4.3.1 -->
<hibernate-mapping>
   <class name="com.learn.mybatis.chapter1.po.TRole" table="t_role"
catalog="mybatis" optimistic-lock="version">
      <id name="id" type="long">
         <column name="id" />
         <generator class="assigned" />
      </id>
      <property name="roleName" type="string">
         <column name="role_name" length="60" />
      </property>
      <property name="note" type="string">
         <column name="note" length="512" />
```

```
        </property>
    </class>
</hibernate-mapping>
```

这是一个简单的 XML 文件，它描述的是 POJO 和数据库表的映射关系。Hibernate 通过配置文件(或注解)就可以把数据库的数据直接映射到 POJO 上，我们可以通过操作 POJO 去操作数据库记录。对于不擅长 SQL 的程序员来说，这是莫大的惊喜，因为通过 Hibernate 你几乎不需要编写 SQL 就能操作数据库的记录。代码清单 1-3 是 Hibernate 的配置信息。

代码清单 1-3：hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE    hibernate-configuration    PUBLIC    "-//Hibernate/Hibernate
Configuration DTD 3.0//EN" "http://hibernate.sourceforge.net/hibernate-
configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <property
name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
    <property
name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property
>
    <property
name="hibernate.connection.url">jdbc:mysql://localhost:3306/mybatis?zero
DateTimeBehavior=convertToNull</property>
    <property name="hibernate.connection.username">root</property>
    <property name="hibernate.connection.password">learn</property>
    <mapping resource="com/learn/mybatis/chapter1/po/TUser.hbm.xml"/>
    <mapping resource="com/learn/mybatis/chapter1/po/TRole.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

别看代码很多，但是全局就是这样的一个 XML 文件，作为数据库连接信息，配置信息也相对简易。然后建立 Hibernate 的工厂对象（SessionFactory），用它来做全局对象，产生 Session 接口，就可以操作数据库了，如代码清单 1-4 所示。

代码清单 1-4：HibernateUtil

```
public class HibernateUtil {
    private static final SessionFactory sessionFactory;
    static {
```