# 轻量级持久层框架设计与实现

戚爱斌[1] 成秀秀[1] 于春雷[1] 彭 雪[2*]
1. 哈尔滨广厦学院信息学院 黑龙江 哈尔滨 150024
2. 哈尔滨医科大学基础医学院 黑龙江 哈尔滨 150081

**摘 要：**本研究针对小型应用中数据库操作频率和复杂性较低的特点，提出开发轻量级持久层框架（EasyDao）的策略。研究方法包括简化配置、提供简洁API、实现自动结果映射以及支持事务处理，旨在提升数据库操作效率。通过在小型教务管理系统和个人博客系统的应用案例中进行验证，结果表明该框架能显著提高开发效率，降低开发难度。为小型应用提供了一种简洁、高效的数据库操作解决方案，具有良好的应用前景。

**关键词：**轻量级；持久层框架；简化配置；事务处理

## 引言

在现代软件开发领域，持久层框架已成为企业级应用的关键组件[1]。诸如MyBatis、Hibernate等框架，凭借其强大的功能与成熟架构，在处理复杂数据库操作及业务逻辑映射方面表现出色。然而，对于小型应用，其数据库操作相对简单，使用这些重量级框架易增加系统的复杂性与开发成本。因此，开发轻量级持久层框架，对提升小型应用开发效率、降低开发难度意义重大。

## 1 轻量级持久层框架的需求背景

### 1.1 小型应用的特点

小型应用通常业务逻辑简单，数据库操作频率低，数据模型单一，对系统性能和资源消耗要求宽松。开发过程更倾向于采用简单、高效、易于上手的技术和工具。

### 1.2 传统持久层框架的局限性

尽管MyBatis、Hibernate等传统持久层框架功能强大，但它们在小型应用中的应用存在一些局限性。首先，这些框架的配置过程较为复杂，需要编写大量的配置文件，如数据库连接配置、映射文件等，这无疑增加了开发的初始成本。其次，框架本身的体积较大，加载和运行时会占用较多的系统资源，对于资源有限的小型应用来说，可能会造成一定的性能瓶颈。此外，传统框架的学习曲线相对陡峭，对于一些小型开发团队或个人开发者来说，掌握和使用这些框架需要投入较多的时间和精力。

## 2 轻量级持久层框架的设计与实现

### 2.1 核心技术介绍

#### 2.1.1 Java反射

Java反射是Java语言中一个强大的特性，它可动态创建对象实例、获取及设置属性值、动态调用方法。通过Java反射机制，轻量级持久层框架可以实现对象与数据库记录的自动映射[2]。

#### 2.1.2 动态代理

动态代理是Java提供的一种代理机制，它允许在运行时动态地创建代理类和代理对象。轻量级持久层框架利用动态代理技术实现了对数据访问对象（DAO）的代理。具体来说，开发者只需定义一个DAO类，轻量级持久层框架会在运行时为该类生成一个代理实现类。代理对象会拦截对DAO接口方法的调用，并将其转换为相应的数据库操作。这种方式不仅提高了代码的复用性，还使得数据库操作与业务逻辑的分离更加清晰。

#### 2.1.3 注解

注解是JDK5.0引入的一种标注机制，允许开发者在代码中添加额外信息，这些信息在编译、类加载或运行时被读取处理[3]。轻量级持久层框架通过注解实现事务声明式处理，自定义@Transactional注解来声明事务边界与行为。

### 2.2 EasyDao框架结构

框架的架构设计遵循了分层和模块化的原则，主要包括以下几个核心模块，其结构如图1所示

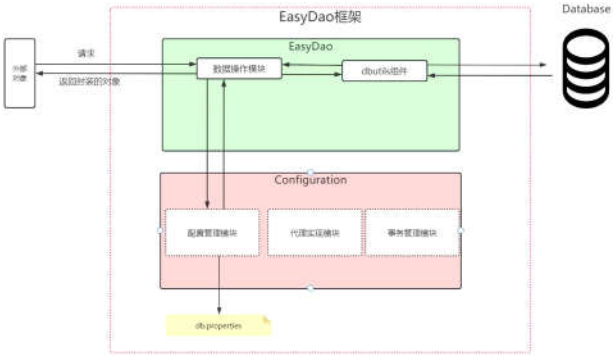


**图1 轻量级持久层框架结构图**

各个模块的作用如下：

配置管理模块：负责读取和解析框架的配置信息，如数据库连接信息、驼峰命名转换、事务管理配置等。配置信息可以通过简单的配置文件进行声明。

数据库操作模块：封装了dbutils组件操作，提供了统一的数据库查询、更新、删除等功能。该模块会根据SQL语句来动态执行相应的数据库操作，并将查询结果封装成对应的实体类。

代理实现模块：利用动态代理技术，为用户定义的DAO接口生成代理实现类。该模块会拦截对DAO接口方法的调用，并将其委托给数据库操作模块执行。

事务管理模块：负责事务的创建、提交、回滚等操作。该模块会根据@Transactional注解的声明，自动管理事务的边界和行为，确保数据库操作的原子性、一致性、隔离性和持久性

## 3 框架的特点

### 3.1 简洁易懂

该框架设计简洁，无需复杂配置和冗长代码。开发者定义实体类、DAO类并用注解声明事务，即可实现数据库访问操作。这种设计使其适合教学及低代码量开发，便于开发者快速理解原理并应用于实际项目。

### 3.2 灵活易用

该框架提供丰富的注解，支持自定义，满足复杂映射与操作逻辑需求，兼容多种数据库操作，支持批量插入、条件查询等，适用多场景开发，声明式事务管理简化了事务处理，避免繁琐的事务代码编写。

### 3.3 性能高效

虽然EasyDao框架是一个轻量级框架，但其性能表现却非常出色。通过合理的SQL解析和执行策略，EasyDao能够有效地减少数据库操作的开销，提高程序的运行效率。同时，事务管理模块也经过优化，确保了事务操作的高效性和可靠性。

## 4 框架应用

### 4.1 应用示例

以下是一个简单的EasyDao框架应用示例，此示例完成的是在事务中完成图书入库的操作。入库涉及到对两个表分别进行插入和修改的操作。即向库存操作表中插入入库记录，同时修改图书表的库存记录。其结构如图2所示

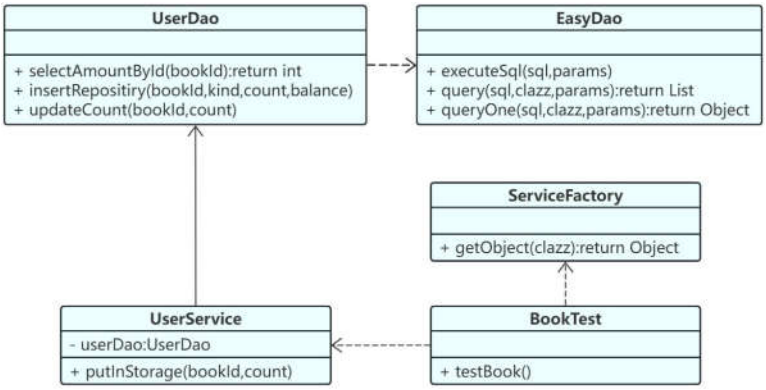


图2 图书入库业务类图

上图中各个类的作用如下：

EasyDao：框架中定义的类，用来操作数据库。

BookDao：数据访问类，定义了三个访问数据库的方法。

BookService：业务类，在此类中控制事务，用来调用Dao的方法。

ServiceFactory：框架中定义的工厂类，用来获取代理对象。

BookTest：测试类，在此类中调用BookService.

#### 4.1.1 BookDAO

在BookDAO类中创建三个方法，selectAmountById（Integer bookId），insertRepositiry（Integer bookId,String kind，int count，int ballance）和updateCount（int bookid，int amount），三个方法的作用分别是查询图书数量，向库存操作表插入记录和更新图书数量。以下代码为BookDAO类的核心代码。

```
public int selectAmountById(Integer bookId){
Object[] objs=EasyDao.queryOne("select amount from tb_book where book_id=? for update",new Object[]{bookId});
return (int)objs[0];
}
public void insertRepositiry(Integer bookId,String kind,int count,int ballance){
EasyDao.executeSql("insert into tb_repository values(null,?,?,?,?,now(),null)",bookId,kind,count,ballance);
}
```

```
public void updateCount(int bookid,int amount){
EasyDao.executeSql("update tb_book set amount=? where book_id=?",amount,bookid);
}
```

4.1.2 BookService

以下putInStorage(Integer bookId，int count)方法是入库操作，此方法调用了DAO的三个基本方法，通过@Transactional注解将三个方法封装成一个原子操作。

```
@Transactional
public void putInStorage(Integer bookId,int count){
int amount=dao.selectAmountById(bookId);
dao.insertRepositiry(bookId,"in",count,amount+count);
dao.updateCount(bookId,amount+count);
}
```

4.1.3 BookTest

以下方法用来测试入库操作。

```
BookService service=ServiceFactory.getObject(BookService.class);
service.putInStorage(1,20);
```

4.1.4 小结

在上述示例中，定义了一个BookDao类，在此类中定义了selectAmountById，insertRepositiry，updateCount方法，同时定义了BookService类，并在putInStorage方法上使用了`@Transactional`注解声明事务。通过ServiceFactory的`getObject`方法获取了`BookService`类的代理实现对象。通过代理对象，可以方便地进行数据库的查询、插入、更新和删除操作，同时事务管理也得到了自动化的处理。

4.2 应用案例

4.2.1 教务管理系统

以小型教务管理系统为例，该系统涵盖学生、教师、就业管理等模块，数据库操作较为简单，主要涉及增删改查及用户注册登录功能。运用轻量级持久层框架开发，开发者仅需编写少量代码。如学生管理中，调用save（student）添加学生，deleteById（studentId）删除指定学生，findAll（）查询所有学生；就业管理中，调用find（params）多条件查询就业信息。在开发过程中，开发者无需关注复杂数据库连接与事务逻辑，只需专注业务实现，相比JDBC减少70%代码，较Mybatis节省50%代码，效率显著提升。

4.2.2 个人博客系统

个人博客系统是数据库操作较少的小型应用，含文章发布、浏览、评论管理等功能。使用轻量级持久层框架，可快速搭建数据访问层。如文章发布时，调用save（article）保存内容；浏览时，findById（articleId）查询指定文章，findAll（）查所有文章列表；评论管理时，调用save（comment）添加评论，deleteById（commentId）删除评论。该框架简洁高效，使开发者能更多投入前端设计与体验优化。

## 5 结论

轻量级持久层框架在小型应用开发中具有重要的应用价值。它通过简化数据库操作、支持事务处理、提供高效的性能表现以及易于集成与扩展的特点，满足了小型应用在开发过程中的实际需求。本文详细探讨了轻量级持久层框架的需求背景、关键技术实现以及应用案例，旨在为小型应用的开发提供一种更为高效、简洁的数据库操作解决方案。未来，随着软件开发技术的不断发展，轻量级持久层框架有望在更多的小型应用中得到广泛应用，为开发者带来更加便捷的开发体验。

## 参考文献

[1]汪荣波.MyBatis3源码深度解析[M].北京:清华大学出版社,2024.

[2]曾水新,黄日胜. Java注解机制的应用研究[J].电脑知识与技术.2022,18(34):35-38.

[3]李黄骏,兰全祥.代理设计模式底层浅析及其在Spring中的应用[J].大众科技.2024,26(2):10-14.