计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2021,30(6):107−111 [doi: 10.15888/j.cnki.csa.007918]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

改进 JDBC 框架的研究与应用①

葛　萌, 欧阳宏基, 陈　伟

(咸阳师范学院 计算机学院, 咸阳 712099)
通讯作者: 葛　萌, E-mail: oyhj_nicholas@163.com

摘　要: 分析了传统 JDBC 框架存在的代码复用性低、耦合度高、不易移植、易出现网络故障导致 Connection 对象失效等缺陷, 结合若干设计模式和数据库重连机制, 提出了一种改进的 JDBC 框架. 该框架通过 DAO 模式向业务逻辑层提供调用持久化逻辑的接口、解耦合业务逻辑与持久化逻辑; 利用模板、策略和工厂模式封装 DAO 的具体实现, 实现具体持久化代码的统一性并减少代码的冗余度. 将改进的 JDBC 框架应用于某高校绩效考核管理系统, 实践结果表明, 通过使用改进的 JDBC 框架, 有效增强了系统的鲁棒性, 解耦合了数据持久层与业务逻辑层, 提高了数据持久层代码的复用率和开发效率.

关键词: JDBC; 设计模式; 数据库重连机制; 数据持久层

引用格式: 葛萌,欧阳宏基,陈伟.改进 JDBC 框架的研究与应用.计算机系统应用,2021,30(6):107–111. http://www.c-s-a.org.cn/1003-3254/7918.html

Research and Application of Improved JDBC Framework

GE Meng, OUYANG Hong-Ji, CHEN Wei

(School of Computer Science, Xianyang Normal University, Xianyang 712099, China)

Abstract: The traditional JDBC framework is featured by low code reusability, high coupling, and difficult transplantation, with connection object failures caused by frequent network faults. In this study, we propose a novel JDBC framework combined with several design patterns and the database reconnection mechanism. Through the DAO pattern, the proposed method provides the persistence logic interface, the decouple business logic and the persistence logic to the business logic layer. The concrete implementation of DAO is encapsulated through templates, strategies and factory patterns, improving uniformity of persistence codes while reducing code redundancy. This novel framework is applied to the performance evaluation system in a university. The results demonstrate that the improved JDBC framework decouples the data persistence layer from the business logic layer and improves the reuse rate and development efficiency of the codes in the data persistence layer, enhancing the robustness of the system.

Key words: JDBC; design pattern; database reconnection mechanism; data persistence layer

数据持久化能够提供获取和长久保存数据的能力, 采用合适的持久化技术构建性能良好、易于扩展和维护的持久化层是各类应用系统设计中必须考虑的重要问题之一. 在以 Java EE 技术为平台的应用中存在着两种主要的数据持久化解决方案: 一种以 SQL 技术为基础, 代表是 JDBC 和 MyBatis 框架[1,2]; 另一种是以对象/关系映射 (ORM) 技术为基础, 代表是 Hibernate 框架[3]. Hibernate 是对 JDBC 的轻量级封装, 以面向对象的域模型为基础, 通过配置文件或注解自动实现 POJO 对象与数据库表的映射, 避免传统方式的建库、建表、

① 基金项目: 咸阳师范学院“青年骨干教师”培养项目 (XSYGG201615); 咸阳师范学院专项科研计划 (XSYK19021, XSYK19054)
Foundation item: “Young Expert Teacher” Cultivating Program of Xianyang Normal University (XSYGG201615); Special Science and Technology Research Program of Xianyang Normal University (XSYK19021, XSYK19054)
收稿时间: 2020-09-23; 修改时间: 2020-10-21; 采用时间: 2020-10-28; csa 在线出版时间: 2021-06-01

设置主外键等关系操作[4]. 但是 MyBatis 和 Hibernate 都是在设计模式+Java 反射机制基础上形成的第三方框架, 其学习成本较高, 对开发人员具有较高的素质要求, 尤其 Hibernate 框架如果使用不当, 在频繁访问数据库、并发量大的场景下, 会出现严重的性能问题. JDBC 通过 SQL 语句直接操作数据库, 相对于 ORM 缺少了“翻译”过程, 经过精心设计与优化后, 具有较高的执行效率. 但是, 传统的 JDBC 也存在缺陷: 需要考虑数据库底层细节、业务逻辑与持久化逻辑高度耦合、相似持久化逻辑的代码冗余度较高、在出现网络故障的情况下, 需要人工重启 Web 容器来解决数据库连接对象失效等问题.

近年来已有不少学者针对上述问题进行了相关研究, 主要集中在 JDBC 基本理论、复用性、访问效率等方面. 文献 [5,6] 主要研究了 JDBC 访问数据库的一般算法和事务控制, 分析了 JDBC 与 ODBC 的区别. 文献 [7] 从策略模式和模板模式入手对 JDBC 进行了改进, 并以改进后的 JDBC 设计了某电信客户关系管理系统的数据持久层. 文献 [8] 针对 JDBC 中的 Connection 对象在网络环境异常情况下无法自动重连的问题, 利用循环检测方法提出一种 Connection 自动连接策略. 文献 [9,10] 从多方面研究了 JDBC 的缓存方案, 提高了数据库访问效率.

本文针对传统 JDBC 存在的不足, 在前人研究基础上, 综合应用单例模式、模板模式、策略模式、DAO 模式、工厂模式和配置文件, 并在数据库驱动加载过程中引入重连机制, 在 DAO 工厂中增加缓存机制, 对 JDBC 进行了改进. 并以改进的 JDBC 框架为基础, 完成了某高校绩效考核管理系统持久层的设计与开发. 并为以 JDBC 作为数据持久化技术的 Java EE 应用系统的开发提供了一定的参考.

1 JDBC 体系结构与核心 API

JDBC 是 Java EE 应用中用来访问关系型数据库的一组 API. 主要包括: JDBC API、驱动管理器、JDBC 驱动程序以及驱动程序的注册机制[11]. 其中, 开发人员使用 JDBC API 访问数据库; 数据库厂商提供 JDBC 驱动程序; 驱动管理器负责加载驱动程序, 在 JDBC API 和驱动程序之间形成一个中转, 允许 JDBC API 通过驱动程序来操作关系型数据库, 如图 1 所示.

JDBC API 通过提供的几个核心接口供开发人员使用, 主要是 Connection、Statement、PreparedStatemnt、CallableStatement 和 ResultSet. Connection 代表应用程序与数据库之间的连接, 是创建其它 JDBC 接口的基础并处理数据库事务. Statement 用于执行普通的 SQL 语句. PreparedStatement 是 Statement 的子接口, 能够对 SQL 语句进行预编译并检查语法错误, 对于相同 SQL 语句的多次执行能够提高效率. CallableStatement 是 PreparedStatement 的子接口, 用于执行数据库端的存储过程. ResultSet 用来封装查询语句所返回的结果集, 并提供了对结果集进行遍历、访问和定位的相关方法. 上述各接口的使用步骤是: ① 通过反射机制加载驱动程序; ② 由 DriverManager 根据所访问数据库的 URL、用户名和密码创建 Connection 对象; ③ 根据 SQL 语句所访问的对象, 由 Connection 创建不同的 Statement 对象; ④ 相关 Statement 对象负责执行具体的 SQL 语句, 并处理执行的结果; ⑤ 释放相关资源.

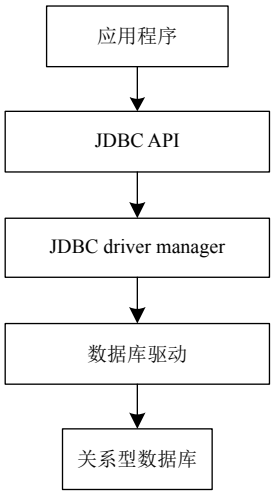

图 1 JDBC 体系结构

JDBC 通过提供相关接口的方式, 有效地在执行过程中明确了责任, 屏蔽了各具体关系型数据库的差异, 降低了应用程序与数据库的耦合度. 由于持久化逻辑具有相似但又不完全相同的特点, 相似性表现在任何一个持久化操作都要经过上述的 5 个步骤; 不同性表现在具体执行的 SQL 语句和处理结果不同. 所以传统 JDBC 存在着一些缺陷: 不同实体对象的持久化操作具有很大的相似性, 如果不进行封装, 会导致代码冗余度高, 复用率低; 业务逻辑与持久化逻辑耦合度高, 存在一些不必要的重复性操作 (例如驱动程序的多次加载等); 数据库的改变需要修改源代码, 包括数据库驱

动获取、连接配置和 SQL 语句等部分, 违背了“开-闭”原则.

2 改进的 JDBC 框架

2.1 改进的 JDBC 架构设计

本文针对传统 JDBC 框架存在的复用性低、耦合度高、不易移植等缺陷, 设计了改进的 JDBC 框架, 通过 DAO 模式向业务逻辑层提供调用持久化逻辑的接口, 解耦合业务逻辑与持久化逻辑; 利用模板、策略和工厂模式封装 DAO 的具体实现, 实现具体持久化代码的统一性并减少代码的冗余度; 利用 XML+Property 配置文件来描述数据库配置、DAO 配置、SQL 语句等信息, 实现代码的调优和解耦. 改进的 JDBC 框架如图 2 所示.

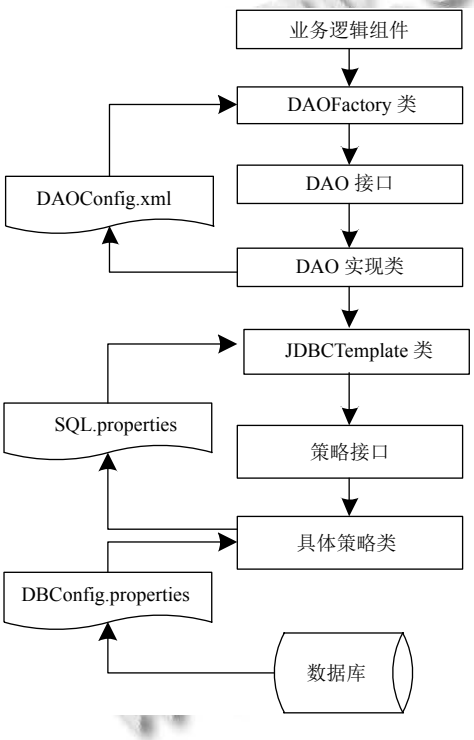

图 2 改进的 JDBC 框架

设计模式是在特定背景下, 解决某个问题的最佳实践, 具体表现为一组精心设计的接口与类的组合[12,13]. 在编码中应用设计模式能够确保可重用性、可靠性, 并易于被别人理解. 根据模式的目的, 一共分为: 创建型、结构型和行为型 3 类. 其中, 工厂模式属于创建型模式, 用于将对象的创建和使用相分离, 在该 JDBC 框架中用于生成 DAO 和具体策略对象. DAO 模式作为业务逻组建访问数据持久层的入口, 屏蔽数据持久化的细节. 模板方法模式属于行为型, 用于定义一个操作步骤的骨架, 将其中可变的操作延迟到子类中完成. 在改进的 JDBC 框架中, 将执行 SQL 操作这个步骤延迟到子类完成. 策略模式属于行为型, 通常定义一个算法族, 由子类实现每个具体的算法并相互之间能够转换. 在改进的 JDBC 框架中, 将 Connection 对象的创建和资源释放这两个操作由策略模式实现, 其中一个具体策略是通过传统的 DriverManager 获取, 资源释放方式是销毁; 另一个具体策略是通过数据库连接池获取 Connection, 资源释放是交回给连接池. 综合应用上述设计模式对 JDBC 框架进行改进, 可以把持久化操作中相似的功能进行封装, 进一步分离业务逻辑和持久化逻辑, 使业务逻辑层和持久化层的耦合度进一步降低, 提高代码复用率.

2.2 改进的 JDBC 架构的执行流程

改进的 JDBC 框架的调用时序图如图 3 所示. 业务层组件通过 DAOFactory 和配置文件获取对应实体的 DAO 对象; DAO 对象中创建 JDBCTemplate 对象, JDBCTemplate 对象通过策略工厂对象得到 JDBCStrategy 对象, 在创建过程中需要策略接口和配置文件的帮助; 并且 JDBCTemplate 封装了 JDBC 操作的公共逻辑, 在 JDBCStrategy 对象的配合下完成对数据库的访问, 并将操作结果返回给业务层组件.

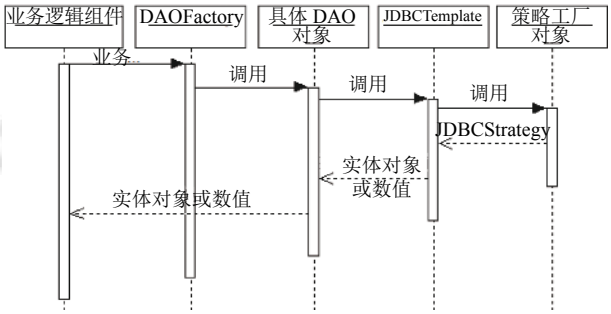

图 3 改进 JDBC 框架的执行过程

3 改进的 JDBC 框架的应用

以下给出改进的 JDBC 框架在高校绩效考核管理系统中的应用实例. 系统环境是 Window Server 2012 作为服务器, MyEclipse16.1 是集成开发工具, 数据库服务器是 MySQL5.7.

3.1 系统流程设计

采用绩效考核系统对高校人员进行量化评价是采用信息化手段进行人力资源管理的一种变革, 能够使

每位人员得到公正、公平的考评结果, 为提高教学质量、工作效率, 并为学校领导层提供正确的决策数据. 系统的考核对象主要分为: 教师系列、行政人员系列和工程实验技术人员系列. 以教师系列为例, 实际的考核流程如图 4 所示.

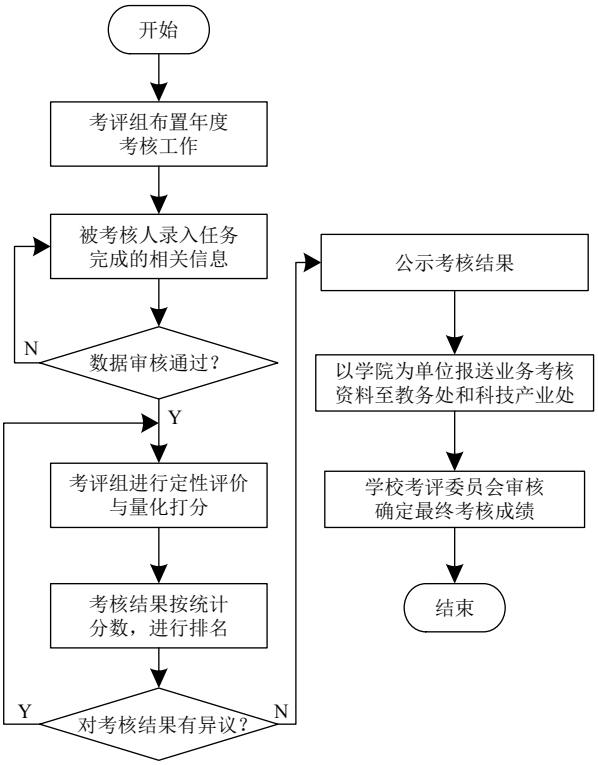

图 4 绩效考核工作流程

首先由各二级学院布置年度考核工作安排, 教师按照年度制定的目标责任书中的内容, 逐条将自己相关的内容录入系统. 然后各二级学院召开考核评议会, 由业务考评组根据考核内容进行量化打分, 按照统计分数进行排名并定性评价. 在此基础上划分优、良、合格和不合格比例. 考核小组在院系内公布考核结果, 如有异议, 教师在三日内到院系考核小组进行复审. 以院系为单位将教师考核相关汇总资料报送教务处和科技处, 最后由教务处将各院系考核材料报送学校考核工作委员会审核, 最终确定各教师本年度绩效考核成绩.

3.2 关键技术

定义 JDBCStrategy 作为抽象策略接口, 包括获取连接和释放资源两个操作. DBPoolStrategy 和 DMStrategy 是两个具体策略类. 在单元测试阶段使用 DMStrategy 策略, 因为它不需要 Web 容器的支持, 可以提高测试效率. DBPoolStrategy 用在整体测试和项目运行阶段, 它以 JNDI 方式获取连接和回收资源[14]. 该策略需要在 Web 服务器中配置数据库连接池, 因为连接池方式能够对 Connection 进行复用, 避免反复创建和释放连接对象而造成的性能消耗. DMStrategy 通过驱动管理者类获取连接和释放资源. 这两个策略类在获取连接对象时都加入了数据库重连机制. 以 DMStrategy 策略为例, 在一个 while 循环中通过 DriverManager 获取 Connection. 如果发生异常就让当前线程对象休息一段时间后重新获取 Connection 对象, 直到获取成功就退出循环, 这样就可以避免重启 Web 服务器或应用程序所带来的麻烦.

具体策略类利用静态代码段加载 JDBC 驱动, 避免了传统方式中多次加载的弊端, 提高了执行效率. 数据库驱动名称、URL、用户名和密码等信息都写在了 DBConfig.properties 配置文件中, 数据库的变更不用修改具体策略类代码, 便于系统的移植. FileUtil 类采用单例模式来实现, 用来把配置文件中的<K,V>键值信息保存到静态的 Properties 对象中, 便于在具体策略类和工厂类中共享.

JDBCStrategyFactory 是策略工厂, 根据配置文件来创建不同的具体策略对象. 具体策略切换时, 只需要修改配置文件中的值,工厂类的代码不需任何改动; 当增加新的连接获取方式时, 只需要定义具体的策略类, 原有的代码不需要任何修改, 很好的满足了“开-闭”原则.

定义 JDBCTemplate 作为模板封装数据表通用的增、删、改、查方法, 利用关联的工厂对象获取具体策略, 利用 RowObjMapper 接口中的 rowObjMapping() 方法将查询结果集转换为实体对象. 以 find() 方法为例, 相关代码如下所示.

```
public class JDBCTemplate
{ private JDBCStrategy jdbcStrategy=JDBCStrategy-Factory.getJDBCStrategy();
public Object find(String sql, Object[] args, RowObj-Mapper mapping)
{ Connection conn = null;
PreparedStatement ps = null;
ResultSet rs = null;
try {conn = jdbcStrategy.getConnection();
ps = conn.prepareStatement(sql);
for (int i = 0; i < args.length; i++)
ps.setObject(i + 1, args[i]);
rs = ps.executeQuery();
```

```
Object obj = null;
if (rs.next()) {
obj = mapping.rowObjMapping(rs);
}return obj;
} catch (SQLException e)
{throw new DaoException(e.getMessage(), e);}
finally
{ jdbc.close(rs, ps, conn); }
}}
```

利用 DAO 模式将业务逻辑与持久化逻辑解耦. DAO 接口定义相关实体对象的持久化方法, DAOImp 实现 DAO 与 RowObjMapper 接口, 通过关联的 JDBCTemplate 完成持久化操作. 为了进一步解耦 SQL 语句与 Java 代码, 将 DAO 对应的 SQL 语句写到 Peroperty 类型的配置文件中, 如果出现持久化逻辑错误只需修改配置文件即可.

通过上述设计, 改进后的 JDBC 框架将传统 JDBC 操作中的加载驱动、创建连接、创建 Statement 对象和释放资源等操作交由 JDBCStratey、JDBCTemplate 和 RowObjMapper 完成, 开发人员只需专注 DAO 的开发. 从而能够降低持久化代码的编写量, 提高开发效率.

3.3 改进的 JDBC 框架应用评价

分别使用传统 JDBC 和改进的 JDBC 对绩效考评系统中的持久化层代码进行设计, 采用 JUnit 框架测试持久化层代码的执行时间, 并统计出各自代码的复用情况, 具体数据见表 1.

表 1 JDBC 持久层代码复用与执行时间对比图

名称	代码总行数	可复用代码总行数	执行1次时间(s)	执行100次时间(s)	可复用率(%)
传统	1276	58	0.37	1.94	4.5
改进	960	180	0.46	1.58	18.7

执行时间是指从业务逻辑层调用持久层代码的执行时间, 分别调用 1 次和调用 100 次, 执行 5 次取平均值. 可以看出在单次执行的情况下, 改进的 JDBC 执行时间较长, 开销主要在两个方面: ① 设计模式增加了对象的调用开销; ② 工厂对象要解析并读取配置文件. 在多次调用的情况下, 改进的 JDBC 节省了执行时间, 主要是因为: ① 传统 JDBC 会多次加载驱动, 改进的 JDBC 只加载一次; ② 改进的 JDBC 在 DAO 工厂中增加了缓存机制. 由此可见, 改进的 JDBC 不但降低了时间复杂度, 并且有效提高了代码的复用率.

4 结论与展望

JDBC 是目前 Java EE 应用中数据持久化操作的一种常见解决方案, 针对传统 JDBC 存在的缺陷, 将策略模式、工厂模式、模板模式和数据库重连机制对传统 JDBC 进行了改进, 并成功地将改进的 JDBC 框架应用于某高校绩效考核系统的数据持久层设计中. 通过项目实践表明改进的 JDBC 框架能够提高数据持久层的开发效率和可复用率, 降低了与业务逻辑层和数据持久层的耦合性, 为 Java EE 数据库持久层的设计与开发提供了相应的借鉴.

参考文献

1 陈小虎, 邓惠俊. 基于 mybatis 的数据持久层研究. 成都工业学院学报, 2020, 23(2): 29–32.
2 王艳清, 陈红. 基于 SSM 框架的智能 Web 系统研发设计. 计算机工程与设计, 2012, 33(12): 4751–4757. [doi: 10.3969/j.issn.1000-7024.2012.12.066]
3 夏赟, 李志蜀. 基于 Hibernate 框架的数据持久化层的研究及其应用. 计算机应用, 2008, 28(9): 2446–2448.
4 申斌, 李利民. 基于 MVC 模式 S2SH 框架的库存管理系统. 实验室研究与探索, 2014, 33(11): 113–117. [doi: 10.3969/j.issn.1006-7167.2014.11.026]
5 郭广军, 陈代武, 胡玉平, 等. 基于 JDBC 的数据库访问技术的研究. 南华大学学报 (自然科学版), 2005, 19(2): 50–54, 57.
6 崔丽群, 宋词. 基于策略模式和静态工厂结合的事务管理. 辽宁工程技术大学学报 (自然科学版), 2009, 28(3): 427–430. [doi: 10.3969/j.issn.1008-0562.2009.03.027]
7 张俐, 张维玺. 改进的 JDBC 框架在数据持久层的应用. 计算机工程与设计, 2010, 31(8): 1746–1749.
8 刘云玉, 段中兴, 原晋鹏. JDBC 数据库重连机制的研究与实现. 计算机应用与软件, 2011, 28(7): 38–40. [doi: 10.3969/j.issn.1000-386X.2011.07.012]
9 韩兵, 沈冲, 方英兰. 基于 JDBC 的缓存数据细粒度管理的研究. 计算机技术与发展, 2019, 29(12): 66–71. [doi: 10.3969/j.issn.1673-629X.2019.12.012]
10 韩兵, 江燕敏, 方英兰. 基于 JDBC 的数据访问优化技术. 计算机工程与设计, 2017, 38(8): 1991–1996, 2031.
11 欧阳宏基, 葛萌, 陈伟. 基于 JDBC 的数据持久化层性能优化研究. 网络新媒体技术, 2016, 5(5): 9–15. [doi: 10.3969/j.issn.2095-347X.2016.05.002]
12 李晓伟, 徐冰霖, 张银发, 等. 设计模式在测控通信构件设计中的应用. 飞行器测控学报, 2012, 31(6): 63–67.
13 李增智, 王宇, 李钢, 等. 面向对象可复用软件设计思想分析. 小型微型计算机系统, 2003, 24(5): 835–839. [doi: 10.3969/j.issn.1000-1220.2003.05.011]
14 欧阳宏基, 葛萌. 基于 Struts2+Ajax+JDBC 的企业级 Java Web 架构. 计算机系统应用, 2017, 26(8): 77–82. [doi: 10.15888/j.cnki.csa.005883]