文章编号:1000-5013(2016)01-0092-06 **doi**:10.11830/ISSN.1000-5013.2016.01.0092

结合 AOP 思想和依赖注入技术的轻量级 MVC 框架

姜林美,李国刚,杜勇前

(华侨大学 计算机科学与技术学院,福建 厦门 361021)

摘要: 为了解决目前 Java 主流模型-视图-控制器(MVC)框架日益庞大,从而导致性能低下的问题,提出了一个轻量级的 MVC 框架.该框架利用面向方面编程(AOP)技术实现了横向业务的剥离,采用控制反转设计模式实现了模块间的最低耦合,并通过 Java 反射技术实现了数据库记录与 Java 对象的自动转换.实验结果表明:该框架以仅仅 70 KB 左右的 JAR 包实现了比其庞大数百倍的类似框架的主要功能,同时支持网络应用和移动应用的开发,具有更高的执行效率.

关键词: Java 反射;依赖注入;控制反转;模型-视图-控制器框架;面向方面编程;移动应用开发

中图分类号: TP 393.01 **文献标志码**: A

在面向对象设计领域,框架是由一组相互协作的可重用设计的类构成,对这些类进行特化(specialized),即可创建不同的客户应用程序[1].目前,框架设计已成为加速信息系统开发和软件重用的一项关键技术[2],尽可能地降低模块之间的耦合性.模型-视图-控制器(model view controller,MVC)框架的核心是能够实现三层甚至多层的松散耦合[3].面向方面编程(aspect oriented programming,AOP)的核心思想是对软件系统中各个互相独立的横切关系(如日志记录、权限检查、缓存和持久化等)加以模块化,使之可以有效地集中被管理,而不会让其分散到程序代码的各个地方[4].AOP 把一个系统看作一批关注点,强调调用者与被调用者之间的解耦,并能够自动将横切关注点织入到面向对象的软件系统中[5],将业务关注点和横切关注点分离[6-7].依赖注入是 Martin Fowler 对 IoC(inversion of control)模式的一种扩展解释,即高层不应依赖于底层,两者都应依赖于抽象;抽象不应依赖于细节,细节应依赖于抽象[8].在 Java 中,依赖注入一般通过反射机制来实现.Java 反射机制允许在运行时加载、探知和使用编译期完全不知道的类[9].目前,SSH(struts2 spring hibernate)框架在 Java Web 轻量级应用开发领域得到了广泛的应用.SSH 是由 Struts2,Spring 和 Hibernate 三个独立组件组合而成,它们各有自己的不同的版本,这些版本相互组合存在兼容性问题和执行效率低下的问题.另外,Hibernate 因采用对象关系映射(ORM)技术导致效率比精心编写的 JDBC(java data base connectivity)差[10].最后,SSH 的各个组件的尺寸越来越大,所带来的问题是应用系统的可维护性差,运行效率的降低.为此,本文提出应用 AOP 和 Java 依赖注入技术的思想的轻量级 MVC 框架.

1 框架的设计

文中提出的框架运行于 Web 容器之上,其总体结构如图 1 所示.

1.1 派遣过滤器

派遣过滤器是框架与 Web 容器交互的界面.派遣过滤器实现了 javax.servlet.Filter 接口.Filter 接

收稿日期: 2014-07-30

通信作者: 姜林美(1976-),男,讲师,博士研究生,主要从事网络安全与网络应用软件的研究.E-mail:clough@hqu.edu.cn.

基金项目: 福建省厦门市重大科技计划项目(3502Z20131019)

口的配置管理器由派遣过滤器创建并调用，负责读取并解析系统配置文件. frame. xml. init()方法中，通过配置管理器读取框架的配置文件 frame. xml，并进行解析. Filter 接口的 doFilter()方法中，利用 Action 调度器对 Web 容器发来的 HttpServletRequest 请求进行处理，并将处理结果通过 HttpServletResponse 反馈给 Web 容器.

1.2　配置管理器

配置管理器由派遣过滤器创建并调用，它负责读取并解析系统配置文件 frame. xml. 根据配置文件中的配置项生成各类映射表，包括 Action 映射表、拦截器映射表、终结器映射表、DAO (data access object)映射表、结果视图映射表和常量映射表. 前 4 种映射表是 Action 调度器进行依赖注入的依据. 另外，配置管理器还通过反射机制创建数据库连接池. 该连接池在 DAO 对象调用数据库操作工具包进行数据库访问时被自动引用，并在派遣过滤器销毁时自动销毁.

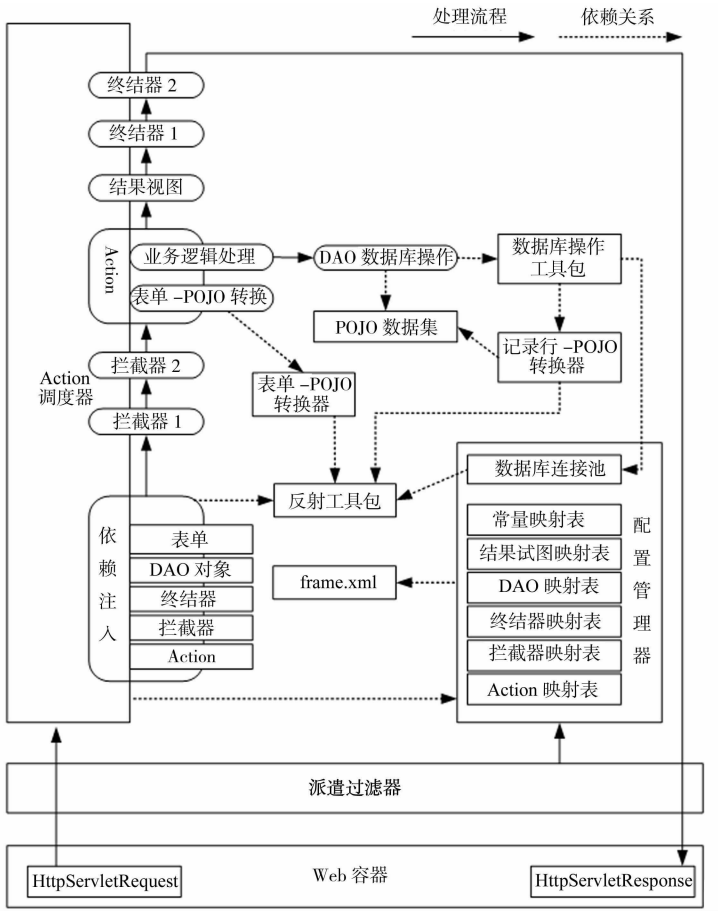


图 1　框架结构

Fig. 1　Structure of the framework

1.3　Action 调度器

当客户端的 HTTP 请求经 Web 服务器送达 Web 容器后，Web 容器生成相应 HttpServletRequest 请求，派遣过滤器则会在其 doFilter() 方法中截获该请求，并通过 Action 调度器对该请求进行处理，并反馈处理结果. 因此，Action 调度器是整个框架的核心.

1.3.1　*依赖注入*　依赖注入是 Action 调度器对一次 HttpServletRequest 请求所做的第一个操作，需要调用反射工具包中的反射处理函数配合完成. 首先，Action 调度器会根据请求的 URL(universal resource locator)信息，在配置管理器的 Action 映射表中查找相应的 Action 类名. 如果找到，则通过依赖注入创建相应的 Action 实例；否则，立即终止请求，并输出错误提示. Action 实例成功创建之后，调度器继续在配置管理器的映射表中查找与该 Action 相关的拦截器和终结器，查找成功，即通过依赖注入创建相应的实例. 此外，调度器还将在配置管理器的 DAO 映射表中查找与该 Action 相关联的 DAO，如果查找成功，则创建 DAO 实例，并将创建的 DAO 实例注入该 Action 对象. 最后，调度器通过 Java 反射功能检查该 Action 中的成员变量中是否有表单类型的对象，如果有，则创建表单实例，并为表单对象的各成员变量注入值. 所注入的值来自 HttpServletRequest 中与成员变量同名的请求参数，表单类型指的是直接或间接从 Form 类继承的类.

1.3.2　*拦截器*　调度器在完成依赖注入之后，在配置文件中，以拦截器配置的先后顺序依次调用拦截器对象的 intercept()方法，该方法以接收请求相应的 Action 对象作为其参数. 在拦截器中可以对请求做任何横切事务处理，并根据横切处理结果决定是否要继续执行 Action 对象中的目标方法.

1.3.3　*Action*　Action 是框架提供给上层应用的主要接口，应用程序应当是在 Action 中或 Action 的交互类中实现业务处理的核心代码.

如节 1.3.1 所述，Action 中若有 DAO 类型的成员变量，并在 frame. xml 中进行了相应的配置，该成员变量的值会在依赖注入阶段进行注入. 另外，Action 中表单类型的成员变量的值也会在依赖注入阶段进行注入. 因此，在 Action 中可以直接使用 DAO 进行数据库访问，也可以通过表单类型的成员变

量直接获得客户端提交的数据.

在 Action 中,可通过"表单-POJO(plain ordinary java objects)转换器"将表单对象转换成 POJO 对象.转换的原则是同名成员变量做类型转换后,进行相应的赋值,转换后的 POJO 对象即可供业务处理使用.此后,若业务处理过程需要进行数据库操作,可使用 DAO 对象来完成.在 DAO 中只需编写 SQL 语句,然后调用数据库操作工具包中的相应方法执行该 SQL 语句,即可完成任何数据库操作.若执行的是有结果集(Resultset)的 SQL 语句,所取得所有记录行(结果集中的)会通过"记录行-POJO 转换器"转换成 POJO 对象,然后整个结果集中的数据将被封装成一个 List 列表(POJO 数据集)返回.

1.3.4 结果视图 结果视图对应 MVC 框架的视图部分,在 Action 中对业务逻辑进行处理之后,可根据处理结果选择在 frame.xml 中配置好的属于 Action 的任一结果视图.框架支持的结果类型包括 FORWARD,REDIRECT,REDIRECT_TOP,CHAIN,PLAINTEXT,JSON 和 XML.前 3 种类型一般用于 JSP 页面,其中,FORWARD 类型用于直接在服务端转发请求的 JSP 页面;RIDRECT 类型则用于重新转发到一个新的 URL 页面;REDIRECT_TOP 和 RIDRECT 一样,但在原始页面由多个 frame(指 html 的〈frame〉或〈iframe〉)构成时,会在顶层显示转发后的结果页面.CHAIN 类型用于直接调用另一个 Action 作后续处理的页面;使用 PLAINTEXT 类型会向客户端传送普通文本;使用 JSON 类型则会将 Action 中的处理结果格式化成 JSON 字符串,再传送给客户端.最后,XML 类型意味着向客户端传送 XML 文档.

1.3.5 终结器 Action 调度器处理的最后一步,在配置文件中,对终结器配置的先后顺序依次调用终结器对象的 intercept()方法,该方法同样以接收请求相应的 Action 对象作为其参数,可对请求做任何横切事务处理.

2 框架的实现

框架源码实现时含 4 个包,分别是工具包 util、数据库包 db、核心包 core 和活动包 action.整个框架最终被打包成 xxdw-platform.jar 提供给上层应用程序来使用,框架的源码组织结构,如图 2 所示.

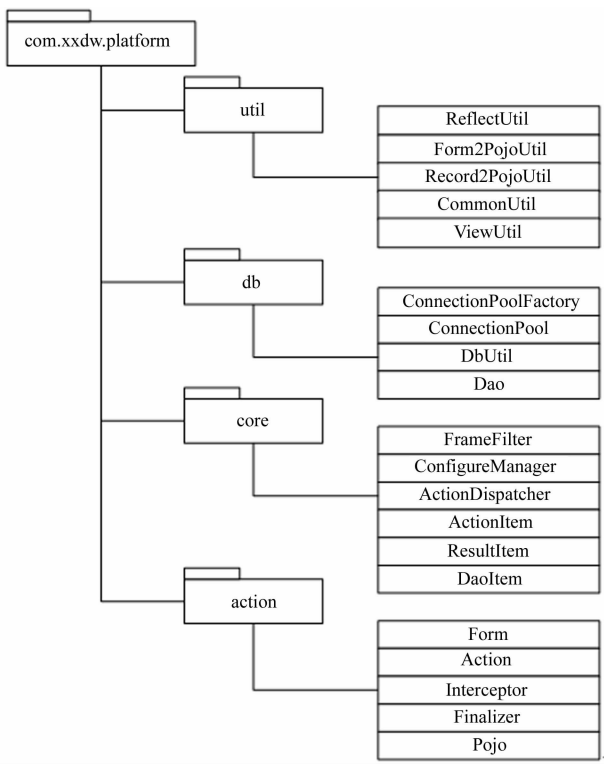


图 2 源码组织结构

Fig. 2 Organization of the source codes

工具包 util 提供整个框架的一些工具类,工具类只有一些静态的方法,以便通过类名直接调用.其中:ReflectUtil 类是实现依赖注入的基础,同时也为 Form2PojoUtil 类进行表单-POJO 的转换,以及为 Record2PojoUtil 进行记录行-POJO 的转换提供支持.ReflectUtil 的核心操作是依赖注入和数据类型的转换.

ViewUtil 定义了和视图操作相关的一些方法,如取上下文路径、视图重定向等.CommonUtil 则定义了其他一些常用的公共方法,如设置调试模式、获取远端 IP 地址、特殊字符转义处理等.

数据库包 db 中的类均与数据库处理相关.其中:ConnectionPoolFactory 是用于创建连接池的工厂类;ConnectionPool 是数据库连接池.配置管理器在读入并解析 frame.xml 时,实例化连接池工厂,并创建连接池;Dao 是一个抽象类,应用程序应当继承该类以实现自身的各种数据库访问操作;DbUtil 类提供一系列静态方法以执行任何 SQL 查询操作,使用 Record2PojoUtil 工具可将数据库记录转换成 Java 对象.

核心包 core 实现整个框架的"控制"(即 MVC 中的 Controller)部分,其中:FrameFilter 是派遣过滤器,它被配置到 Web 容器的 web.xml 文件中,以截获客户端的 ServletRequest 请求;ConfigureManager

即配置管理器，管理 frame. xml 中的各配置项；ActionItem，ResultItem 和 DaoItem 分别用于其中的 Action 配置项、结果视图配置项和 Dao 配置项；ActionDispatcher 对应框架中的 Action 调度器，对整个 HttpServletRequest 的请求处理过程进行调度处理.

活动包 action 中实现了一些应用程序所必须继承的基础类或应当实现的接口，应用程序主要通过这些类使用框架的各项功能. 其中：Action 类是业务活动基类，所有继承该类的类应当在 frame. xml 中进行配置，并会在框架的依赖注入阶段被实例化；Form 是所有表单类应继承的基类，用于接收客户端提交的数据，表单类也会在依赖注入阶段被实例化，并被注入数据；Interceptor 和 Finalizer 分别为拦截器和终结器接口；Pojo 类则是应用程序中的简单对象类.

3　实验结果与分析

为验证所提出的轻量级 MVC 框架的性能，建立了两个 Web 项目，分别采用 struts 2 框架和所提出的轻量级 MVC 框架对客户端的请求进行处理. 客户端通过多线程来模拟并发连接，每个线程可连续向服务端发起多个请求. 线程数称为并发数；每个线程连续发起的请求数称为每并发请求数，简称请求数. 因此，总请求数＝并发数×请求数，客户端向服务端提交请求的核心代码如下所示.

```
//建立一个 Socket
Socket socket=new Socket(InetAddress. getByName("localhost"),8080);
//提交的数据
String para="username=alice&password=568998";
StringBuffer command=new StringBuffer()
    . append("POST" +path+ "HTTP/1. 1\r\n" )
    . append("Accept: * / * \r\n")
    . append("Content-Type: application/x-www-form-urlencoded\r\n")
    . append("HOST:localhost:8080\r\n")
    . append("Connection:Close\r\n")
    . append("Content-Length: "+para. length() + "\r\n")
    . append("\r\n"+para);
//发送命令
BufferedWriter writer=new BufferedWriter(new OutputStreamWriter(socket. getOutputStream
()));
writer. write(command. toString());
writer. flush();
//接收返回的结果
BufferedReader reader = new BufferedReader(new InputStreamReader(socket. getInputStream
()));
while (reader. readLine() ! = null);
```

两个 Web 项目所构建的服务端接收相同的请求数据（即 username＝alice&password＝123456）并对其进行读取，然后向客户端返回相同的 jsp 响应页面.

实验一的客户端和服务器采用同一台主机，该主机的配置如下：CPU 为 Pentium（R）Dual Core CPU E5200@2. 5 GHz，内存 3. 0 G，操作系统为 Windows 7，32 位旗舰版 Service Pack 1，Web 服务器为 Apache Tomcat 6. 0. 26.

实验二的服务端同实验一，客户端则为一台安卓华为 C8813 手机. 华为 C8813 的主要参数：操作系统为 Android OS 4. 1，CPU 为高通骁龙 Snapdragon MSM8625Q 双核 1. 2 GHz，内存 512 M.

两个实验各自的总请求数均为 2 500（基本接近 Tomcat 6 所能接受的并发连接数的上限），各自测试 3 种具有代表性的情形：并发数为 1，每并发连接数为 2 500；并发数为 50，每并发连接数也为 50；并发数为 2 500，每并发连接数为 1. 第 1，3 种情形属于极端情况，中间一种情形代表了普通情况. 3 种情况

下,采用 struts 2 的服务端和采用文中 MVC 框架的服务端的平均耗时(每种情形均进行 1 000 次测试),如表 1 所示. 表 1 中:T 为并发数;R 为请求数. 由表 1 可知:采用轻量级 MVC 框架的服务端所用时间远低于采用 struts 2 的服务端所用时间.

表 1 处理 2 500 个连接的耗时

Tab. 1 Time cost for processing 2 500 connections

客户端	T	R	t(struts 2)/ms	t(轻量级 MVC)/ms
PC 机	1	2 500	8716.9	4 898.9
	50	50	5 722.8	3 802.9
	2 500	1	7 226.5	5 057.1
	1	2 500	9 886.5	7 744.5
手机	50	50	8 203.5	7 246.2
	2 500	1	15 311.2	14 041.5

为更清晰地观察两个框架的性能区别,根据表 1 计算出的两个框架每秒所能处理的请求数,如表 2 所示. 由表 2 可知:在客户端为 PC 机的情况下,相比于 struts 2,采用文中轻量级 MVC 框架可提升处理请求的能力平均达 57.11%;在客户端为手机的情况下,受限于移动终端的并行处理能力,相比于 struts 2,采用文中轻量级 MVC 框架可提升处理请求的能力小一些,但平均仍达 16.64%.

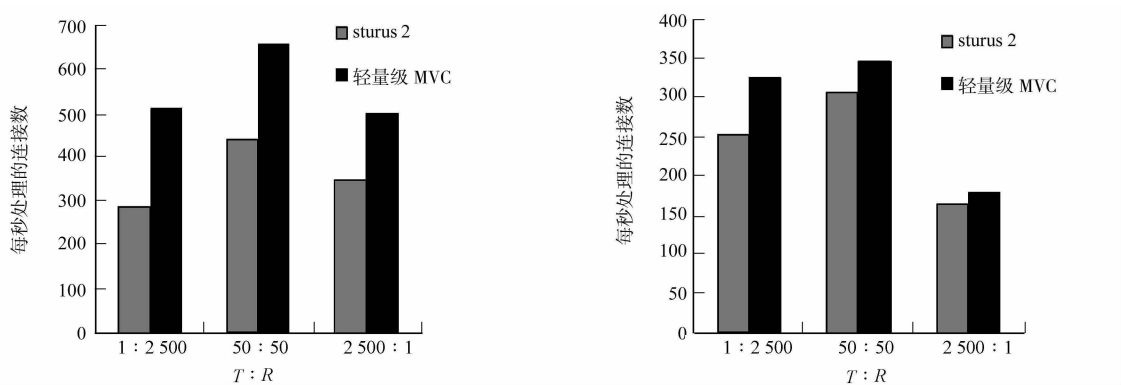
每秒所能处理的连接数对比,如图 3 所示. 图 3 中:横轴标识中的 T 表示并发数;R 表示每并发请求数. 由图 3 可以直观地看出:文中轻量级 MVC 框架的性能优越性.

从实验结果可以看出:两个框架在并发数和每并发请求数均衡的普通情况下的处理能力均优于各自的极端情况,但是在任何情形下,文中所述框架在性能上均优于 struts 框架.

表 2 每秒处理的请求数和性能比较

Tab. 2 Performance comparison of the processing request numbers per second

客户端	T	R	每秒处理的请求数		性能提升/%
			struts 2	轻量级 MVC	
PC 机	1	2 500	286.80	510.32	77.94
	50	50	436.85	657.39	50.49
	2 500	1	345.95	494.35	42.90
	1	2 500	252.87	322.81	27.66
手机	50	50	304.75	345.01	13.21
	2 500	1	163.28	178.04	9.04



(a) PC 客户端 (b) Android 客户端

图 3 每秒所能处理的连接数对比

Fig. 3 Connection processing performance comparison

4 结束语

鉴于 AOP 程序设计和 MVC 框架在 Web 应用程序开发中日益广泛的应用,而现存的采用 Java 平台的 MVC 框架(如 SSH)日益庞大繁杂,提出了一个精简的轻量级的结合 AOP 和依赖注入的 MVC 框

架.该框架的成品 jar 包库仅有 72.1 KB,远小于 SSH 和 Spring MVC 等框架的 jar 包库.实验数据说明:该框架相比于 struts 2 具有更好的运行性能.另外,该框架已在多个 Web 和移动应用项目中得到应用.实践结果证明:该框架能极大地提升应用开发的效率.最后,增加 Web 服务支持可以更大地提高框架的通用性,这也是进一步的研究方向.

参考文献:

[1] LI Fangxing,BROADWATER R P. Software framework concepts for power distribution system analysis[J]. IEEE Transactions on Power Systems,2004,19(2):948-956.

[2] CHEN Liyan,GAO Qing. Research on framework developing technology based on MVC[J]. Advances in Information Sciences and Service Sciences,2011,3(3):25-31.

[3] 张忻.基于 MVC 模式的 Struts 框架在物流管理信息系统中的应用[D].成都:西南交通大学,2005:7-8.

[4] 徐倩颖,杨宗源.面向方面编程的一种新型设计模式[J].华东师范大学学报(自然科学版),2008,2008(1):68-74.

[5] 史玉珍,李波.一种面向方面的 UML 建模方法研究[J].计算机测量与控制,2009,17(12):2497-2499.

[6] 陈月娟,李慧,刘光远,等.基于 AOP 的信息管理系统的研究与实现[J].计算机应用与软件,2010,27(2):130-132,140.

[7] 卞世晖,李龙澍,陈圣兵,等.基于 AOP 理念的 Struts 2 拦截器的研究与应用[J].电子设计工程,2010,18(1):8-9.

[8] ROBERT C M.敏捷软件开发:原则、模式与实践[M].北京:清华大学出版社,2003:116-118.

[9] 费廷伟,刘淑芬,屈志勇,等.Java 反射驱动的规则引擎技术研究[J].计算机应用,2010,30(5):1324-1326,1330.

[10] 柏银.基于 Struts+Spring+Hibernate 多架构的性能分析系统[D].成都:四川大学,2006:26-30.

A Light-Weight MVC Framework Combining AOP and Dependency Injection

JIANG Linmei, LI Guogang, DU Yongqian

(College of Computer Science and Technology, Huaqiao University, Xiamen 361021, China)

Abstract: To solve the problem of low performance caused by gradually increased size of the most popular Java MVC (Model-View-Control) frameworks, a novel light-weight MVC framework is presented. In the proposed framework, AOP(Aspect Oriented Programming) technology is used to deal with the cross-cutting business logic, and Inversion-of-Control pattern is adopted to achieve the lowest coupling among modules. Meanwhile, Java reflection technology is used to transfer a database record to a Java object automatically. The experimental results show that although the JAR package of the proposed framework takes only around 70 KB, it not only realizes all the primary functions of the similar frameworks whose size are hundreds of times more massive, but also supports the development of both web application and mobile application with higher execution efficiency.

Keywords: Java reflection; dependency injection; inversion of control; model view controller; aspect oriented programming; mobile application development

(责任编辑: 陈志贤　　**英文审校:** 吴逢铁**)**