面向方面软件开发在J2EE企业应用系统中的实现

赵 艳,刘同明
(江苏科技大学 电子信息学院,江苏 镇江 212003)

摘 要:面向方面编程(AOP)是面向对象编程(OOP)的扩展和延续,能够很好地解决横切关注点问题,实现了业务逻辑与非业务逻辑的解耦合。目前大部分研究还主要是把AOP作为一种程序设计方法加以研究的,实际上,AOP对于软件设计和开发过程的影响是全面的。因此在AOP的基础上,从概念、规约、实现三个视角定义了方面,并从概念层、规约层、实现层三个层次研究了面向方面软件开发(AOSD)的开发过程,在Spring框架下给出了J2EE企业应用系统的实现过程,与只使用OOP方法相比,不但简化了设计,也使代码更具可读性。

关键词:面向方面编程;面向方面软件开发;关注点;企业应用
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2008)10-0225-05

Implementation of AOSD in J2EE Enterprise Application System

ZHAO Yan, LIU Tong-ming
(College of Electronic and Information of Jiangsu University of Science and Technology, Zhenjiang 212003, China)

Abstract: Aspect-oriented programming is extension and continuation of object-oriented programming, it can commendably deal with the problem of system crosscutting concern, to implement loosing coupling between business logic and non-business logic. Now the majority of research is on the level of treating AOP as a method of programming, actually, the influence is all-sides on software design and developing process. So based on AOP, defines the aspect in three different perspectives (concept, specification and implementation), try the developing process of AOSD from three different layers-conceptual perspective, specification and implementation, at last, the implemented process of J2EE enterprise application system is displayed with Spring framework, compared with OOP method, it is not only easy for design, but also is more readable.

Key words: aspect oriented programming; aspect oriented software development; concern; enterprise application

0 引 言

目前企业应用系统大多采用面向对象程序设计语言(OOPL,如Java,C++等)进行开发,通过对对象的封装,实现了模块化和信息隐藏。面向对象(OO)方法有诸多优点,如便于软件构件化、软件复用和软件维护,软件体系结构良好等[1],但是,人们在实践中也发现了面向对象方法的不足。例如,软件系统除了需要实现一些核心业务功能外,还有一些辅助业务或公共需求(如日志记录、事务完整性、安全性、线程池等)需要处理,虽然OOP针对这些公共需求进行了封装,但在每一个核心关注点(系统要完成的业务逻辑功能)代码中都要对它们进行显式调用,这使得横切关注点[2](系统要完成的非业务逻辑功能)代码分散在各个核心业务方法中,无法很好地封装在单个模块中。

针对横切关注点这种需求,1997年的欧洲面向对象会议提出了面向方面编程(Aspect-Oriented Programming, AOP)方法,现在,AOP已经得到了软件设计和开发人员的认可。AOP从更高层次上对软件系统进行抽象,将传统的按功能或按对象纵向划分程序模块的方法转化为按系统特征横向划分。作为一种新的编程方法学,目前已经出现了许多支持方面的语言和工具,面向方面作为一种新的思想也开始渗入到软件设计开发的各个领域。

然而,仅仅从程序设计角度研究AOP是远远不够的,程序设计方法涉及用于指导程序设计工作的原理和原则,以及基于这些原理和原则的设计方法和技术[3]。AOP技术还没有形成一个成熟的统一标准,因此从工程化方面研究AOP是其发展的需要。现在也有很多关于AOP的架构和框架的研究,但都还没有大量用于实际的企业应用开发。面向方面软件开发(As-

收稿日期:2008-01-22
基金项目:中国船舶工业基金项目(2007HD011G)
作者简介:赵 艳(1983-),女,河南濮阳人,硕士研究生,研究方向为应用软件框架;刘同明,教授,研究方向为智能信息处理、软件工程。

pect - Oriented Software Development, AOSD)是一个新兴的技术规范,用于提取横切于多个系统组件中的关注点,并通过新的组合技术来组合方面和组件。现在基于AOSD的研究基本上始于软件开发周期的编码阶段,实现了横切关注点的模块化,还没有全面体现在软件开发的各个阶段,AOSD还应在需求工程、架构设计和详细设计阶段进行完善[4],文中将从AOSD的完整过程方面讨论AOP问题。使用AOSD方法进行的架构设计是问题空间域到二维解空间域纵横两个方向上的映射,解决了使用OOP方法时多维问题空间域到一维解空间域单维方向上的映射所带来的关注点模糊问题。

1 AOP概念及AOSD的目标

1.1 AOP基本概念

(1)关注点(concern)。一个关注点(concern)就是一个特定的目的,或是一块感兴趣的区域,一段需要的逻辑行为。典型软件系统包含核心级关注点和系统级关注点即横切关注点。在OO方法中,关注点被模型化为类和对象;结构化方法则将关注点声明为程序段;在AOP中,系统级关注点被扩展为横切关注点概念,核心关注点的实现采用OO方法实现,横切关注点则被封装在方面中。

(2)横切关注点(crosscutting concern)。系统中有许多相互独立的系统级需求,它们一般横切于多个核心模块,这种用于完成系统级需求的关注点称为横切关注点。在传统编程语言中,一个软件模块对应一个可执行的代码块,一个横切关注点被局部化到多个模块中,其实现处于缠绕状态[5]。

(3)方面(aspect)。从抽象意义上讲,Aspect是对系统组件性能和语法产生一定影响的一些属性;从设计上讲,Aspect是横切软件系统的一些系统级关注点[3];从实现上讲,Aspect是一种程序单元,它支持将横切关注点封装到单独的模块单元中。

(4)切入点(pointcut)。这是为详细说明方面横切的点而引入的概念,实质上是对方面横切的连接点集合的声明,对连接点集合的抽象。

(5)连接点(join point)。是程序执行过程中明确定义的点,可能定义在方法调用、条件检测、循环的开始或是赋值动作处。每个连接点都有一个与之相关联的上下文。

(6)通知(advice)。通知是方面在一个特定的连接点执行的动作,根据不同的语法定义,这些功能可以在切入点之前、之后执行,也可以替换切入点执行。许多AOP框架(如Spring)都将advice模型化为一个拦截器,基于一个连接点维持一个拦截器堆栈。

(7)AOP代理(AOP Proxy)。这是AOP框架为实现方面规约而创建的代理对象。

1.2 AOP两个特性

多量化和不知觉性[6]是面向方面技术的两个最基本的特性,多量化指的是方面的单个表示会影响到多个程序模块;不知觉性是指人们无法通过检查基础代码的程序体来分辨出方面代码是否执行,这种特性允许在系统创建的过程中更大范围地分离关注点。

1.3 AOSD的目标

AOSD的目标是使整个系统更好地模块化,包括使功能性/非功能性需求等许多不同的关注点更好地模块化,从而保持各关注点相互独立,构建出易于扩展、易于维护、可重用的软件系统模块,在整个软件生命周期中提供系统化标识和组合横切关注点的手段。

AOSD描述的面向方面是一个完整的实施过程,AOP的发展空间不应该仅仅局限于代码级,面向方面的分析与设计(AOA/AOD)对面向方面技术的未来起到了主导作用。在AOSD的基础上得到的架构必须有效地保持关注点相互分离,首先在初始阶段定义合理的关注点,然后定位合理的横切关注点、完成方面以及切入点的粒度控制,多个横切关注点的交互问题,接下来才是AOP部分。

2 面向方面软件开发过程的视角层面架构

Martin Fowler提出了对象设计的三个层面:概念层、规约层(Specification)、实现层。文中引用三种视角[7]对AOSD分析了方面在这三个层次中的体现。方面设计的过程可以简单地表示为图1。

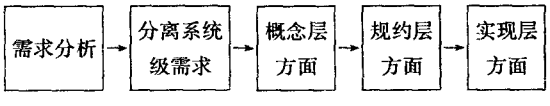

图1 方面设计的层次

首先识别系统需求,分离出功能需求和非功能需求,然后将模块中的关注点分为核心关注点和横切关注点。目前可参考的面向方面需求工程(Aspect - Oriented Requirements Engineering, AORE)方法[4]主要有基于视点的AORE方法,面向方面的需求分析(AORA)方法,关注点多维分离方法(Multi - dimensional approach to separation of Concerns, MDSOC),以及基于目标的方法等。

当需求变化时如何使设计改动最小,是设计者极为关注也是非常头痛的问题。AOP先驱提出了一些用于指导AO设计和开发的原则。下面是与方面设计相关的设计原则[1],基于这些原则可以设计出具有弹

性的AO架构:

* 开闭原则,方面是可以扩展的,但是不可以修改源文件和配置文件。

* Liskov替换原则,方面与基类型一块必须能够替换掉它们的基类型。

* 通知替换原则,通知必须遵从连接点所规定的契约。

* 导言(Introduction)替换原则,一个导言必须符合类的规定。

* 切入点倒置原则,切入点不应该依赖细节,细节应该依赖抽象。

* 切入点作用域原则,切入点的作用域越大就越抽象。

2.1 概念层

概念视角呈现了所研究领域中的各种概念,用于建立问题域的概念模型。该视角回答了"软件要负责什么?"的问题。概念层中的方面是负责完成一定任务的实体,在本层次中对分离出的系统级关注点进行架构,对其进行概念建模,即对所有方面实体进行架构,这些实体通过横切于主业务功能来实现应用系统。可以通过识别系统中的关键用例来构建架构。

2.2 规约层

规约视角回答了"怎么使用软件?",即软件提供了什么样的接口和方法。在规约层,方面作为一个单独的模块与OOP中的类相对应,在本层应该从抽象层次上考虑如何将方面织入(Weaving)到其它核心业务模块中,即选择连接点、创建通知、构建切入点等;还要考虑方面与功能类之间的关系以及方面之间的交互关系。对于横切关注点的设计实现有大量方法提供支持,如面向方面组件工程(Aspect－Oriented Component Engineering, AOCE)、超空间方法等。另外还可以借助模式简化设计,使用模式已经成功地解决了OO系统中出现的很多问题,很多人已经把AOP成功地运用到了OOP的设计模式上(如代理模式),解决了现有设计模式中的横切问题,同时也可以借用一些AOP特有的设计模式,如:Cuckoo's Egg等,这些设计模式都可以简化设计。根据织入的时间将织入分为三类:

① 静态织入:即编译时织入,借助预编译器对源代码进行增强,在编译前进行预处理,将方面代码自动织入到功能模块代码的合适位置,也可以在编译后,对编译后的代码进行操作,其代表是Aspect J。

② 载入时织入:指在代码载入时实现代码的织入,其代表是J Boss AOP。

③ 动态织入:即运行时织入,利用Java动态代理(dynamic proxy)机制在运行时拦截方法调用,根据对方法的调用执行适当的方面代码以实现方面的逻辑织入。大多数AOP框架均以这种织入机制实现,灵活性强,且对被织入模块的侵入性较小,其代表有Spring AOP。

2.3 实现层

实现指的是软件的代码,该视角回答的问题是:"软件怎样履行自己的责任?"实现层中的方面是用面向方面语言编写的代码(如Aspect J是对Java编程语言的面向方面的一个无缝扩展,具有完全的Java平台兼容性),方面通过代码来实现非业务逻辑功能。横切关注点的实现代码被模块化到一个方面模块里,解决了由于横切关注点的缠结而引起的软件复杂性。这些被模块化的Aspect必须实现与业务功能模块的集成,这里会使用AOP提供的织入机制具体实现方面的织入。

3 企业级应用系统示例

基于J2EE平台的企业应用系统的体系结构一般采用四层或五层模式,参见图2。客户端层用来与用户交互,并把来自系统的信息显示给用户。表现层由Web服务器和Web组件构成,主要是处理客户请求,调用相应的逻辑模块,并把结果以动态网页形式返回到客户端。业务层处理应用的核心业务逻辑。企业信息系统层处理企业系统软件,包括企业基础系统、数据库系统和其他遗留系统。业务层代码的逻辑主要用来满足特定商务领域的需要。

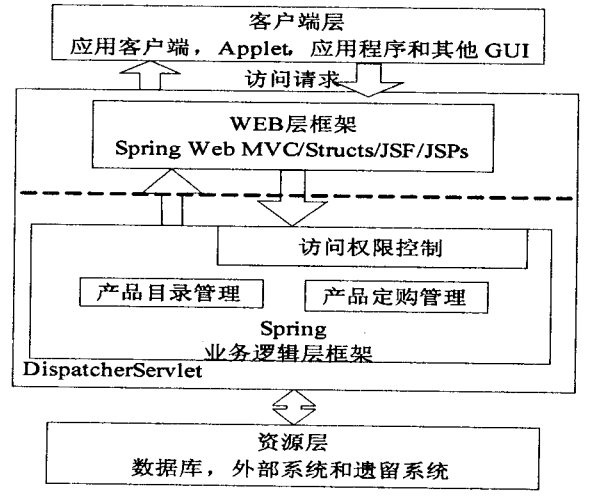

图2　计算机产品销售系统的架构

图2表示一个计算机产品销售系统,它有两类用户:一是消费者,能够浏览计算机产品目录以及在线订购产品;另一个是生产商/零售商用户,是直接销售计算机产品的商家,能够通过自己的Web发布产品目录。不同的用户对于同一个资源可能具有不同的访问

权限,这就需要一种基于业务逻辑方法的权限控制实现方案。

本例基于 Spring 的 AOP 框架进行开发,在业务功能模块都已经实现的基础上,完成不同用户拥有不同权限的设置。该应用有两个核心业务逻辑,即产品目录管理和产品定购管理。

图 2 系统的架构设计基于 MVC 模式,通过控制器发送命令后调用业务逻辑,访问权限控制在业务逻辑层实现,在调用业务逻辑前进行权限认证,从而判断访问者的权限,把权限信息返回给控制器。这里的访问权限控制就是要实现的横切关注点。

在 J2EE 企业应用开发中,主要用到 AOP 的拦截能力,可以在调用对象的方法前/后加入自定义行为,使我们可以处理企业应用中的横切关注点,并且仍然保持强类型。基于 AOP 的访问权限控制对于业务功能逻辑是透明的,访问控制可以在业务功能代码完成后动态地织入到应用程序中,这就可以实现业务逻辑和访问权限控制的解耦。

3.1 概念模型

要构建访问权限认证问题的概念模型,首先是识别系统的关键用例,然后架构权限控制这个关注点。根据图 3 的简化用例图,借助元素结构图可以更清楚地表达 checkPower 这个横切关注点所处的上下文(参见图 4)。应用层包含了主要的参与者:消费者客户和销售商(含总经销商、销售代表等),领域层中的元素有产品管理、目录管理、定购管理三个包,参与者访问领域层的元素时都会涉及到访问权限认证问题。

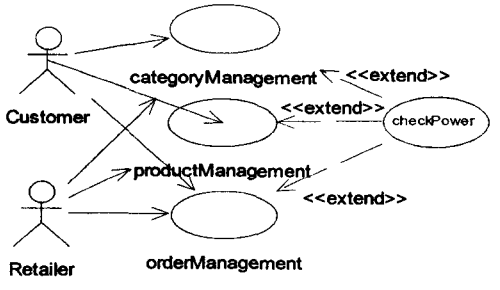

图 3 销售系统的简化用例图

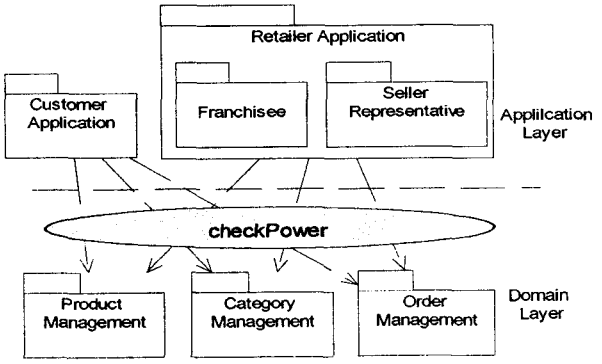

图 4 销售系统的元素结构图

3.2 规约层的方面设计

需要传输的数据的逻辑视图如图 5 所示,权限认证机制横切于各个类中。在本层次中可借助类图识别出连接点。

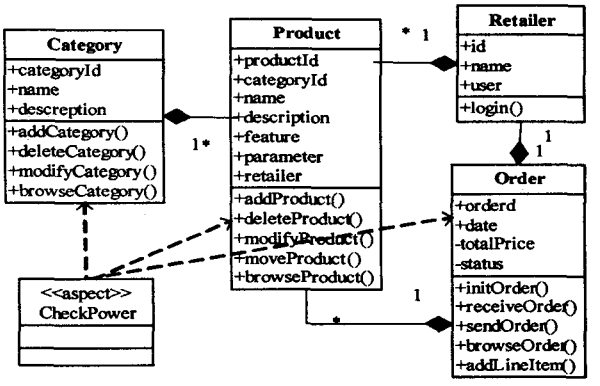

图 5 数据实体关系图

如图 5 所示,当调用类 Category 的 addCategory() 方法时需要权限认证,这里就识别为一个连接点。将识别出的连接点的集合声明为一个切入点 controlledAccess,权限控制都是在访问之前进行,这里应该使用 Before 通知,使用面向方面的方法把安全认证机制封装到一个 CheckPower 方面模块中。图 6 是 CheckPower 的方面图。

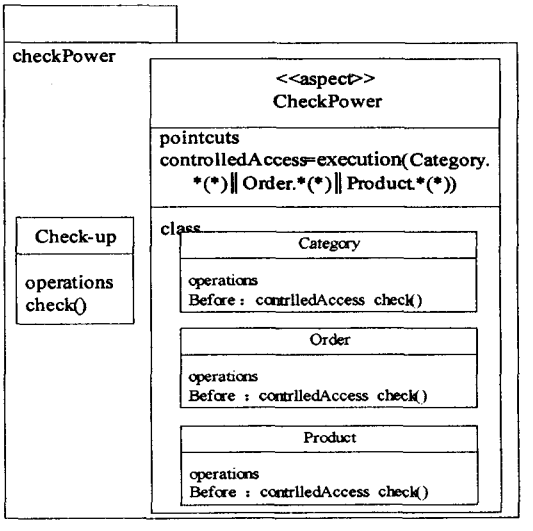

图 6 权限认证模块图

3.3 代码实现

Spring 是一个非常灵活的应用框架,非常容易与其它框架集成,可以部分或者全部引入其它开源应用框架的功能。Spring2 中的 AOP 部分引入了 AspectJ 的切入点语法,基于模式并使用@AspectJ 标记实现方面的自定义,从而可以使用基于 AspectJ 的切入点表达式来描述切入点[8]。Spring AOP 框架是基于代理模式的,缺省时使用 JDK 的动态代理,JDK 代理可以代理任何接口。而 CGLIB 代理主要用来代理类而不是

接口,如果业务对象没有实现一个接口,Spring将使用CGLIB进行代理。

基于权限认证模块图来完成代码实现,本例中可以通过三个方面实现所需的安全性逻辑非功能需求:方面Validity用以认证用户的身份;方面UserManagement负责Retailer的初始化;方面CheckPower实现各种访问控制,根据用户身份的不同,拥有的操作权限也不同。

4　结束语

从一个更高的层面上研究了面向方面的软件开发过程,并基于AOSD的思想,提出了方面开发的三个层次,在Spring AOP框架下设计了J2EE应用系统的访问权限控制方案,实现了业务功能模块与访问控制代码的解耦。

面向方面的编程允许在松散耦合的方式下,通过实现横切关注点来开发应用,可以实现核心关注点和横切关注点的完全分离,提供了比OO方法更高层次的模块化技术。软件的设计和开发是一个工程问题,必须从软件工程的高度来研究面向方面的思想和方法,并融合OO思想和方法,从而建立起一个更为完善的软件工程学。

参考文献:

[1] Wampler D. Aspect－Oriented Design in Java/Aspect J and Ruby[C]//29th International Conference on Software Engineering. [s.l.]:Computer Society, 2007.

[2] Jacobason I, Ng Pan－Wei. AOSD中文版——基于用例的面向方面软件开发[M]. 徐　锋译. 北京:电子工业出版社,2005.

[3] 程　虎. 面向方面(aspect)的程序设计方法[J]. 信息技术快报,2005,3(1):1－4.

[4] Greenwood P. On the Contributions of an End－to－End AOSD Testbed[C]// Early Aspects at ICSE: Workshop in Aspect－Oriented Requirements Engineering and Architecture Design. [s.l.]:Computer Society, 2007.

[5] Miller S K. Aspect－Oriented Programming Takes Aim at Software Complexity[R]. [s.l.]:Computer Society, 2001:18－21.

[6] Filman R E, Elrad T, Clarke S, et al. 面向方面的软件开发[M]. 莫　倩,王　恺,刘东梅,袁　臻译. 北京:机械工业出版社,2006.

[7] Shalloway A, Trott J R. 设计模式解析[M]. 徐言声译. 北京:人民邮电出版社,2006.

[8] 蔡世友,吴嘉俊,冯　煜,等. 深入Spring 2:轻量级J2EE开发框架原理与实践[M/OL]. 2006. http://www.easyjf.com/spring/spring2－aop.htm#_Toc151404118.

(上接第224页)

为日报表,月报表,年报表,用户可以需要的方式实现历史数据的查询。实时报表和历史报表均具有保存、打印等功能。

3.6　数据库系统

当用户选择"数据库系统"功能项后,系统给出数据管理子菜单,内容包括:数据查询、数据备份、数据打印和退出[6]。

数据查询为用户提供查阅指定时间范围内的空调运行数据的功能。

本系统可为用户保留18个月以上的运行记录数据,用户可在适当的时候及时进行数据备份工作,系统也会在一定的时间范围内,提醒用户进行数据备份。若用户未能在系统提示的时间范围内进行数据备份,历史数据将会丢失。

数据打印功能将用户指定变量、指定时间范围内的数据从打印机输出,供用户使用。

3.7　帮助系统

在帮助系统内,用户可查询本软件的使用方法,所用到的软、硬件设备信息以及系统包含的所有变量的信息。

4　结束语

文中所述的空调监测系统已在广元火车站等单位实际运行,结果表明,本系统在整体工况监侧、局部数据显示、历史数据管理和人机界面等方面,均达到了传统监侧方法无法比拟的运行效果,为计算机在楼宇自动化空调监测系统中的应用提供了成功的例证。

参考文献:

[1] 宁永生,王琪辉,张　英. 大型空调中央监控系统设计[J]. 暖通空调,2004(3):53－56.

[2] 冯建农. 中央空调电脑监测系统设计与实现[J]. 监测技术,1997(6):40－41.

[3] 周雄辉,陈焕新. 中央空调系统远程数据采集和监控系统的开发[J]. 建筑热能通风空调,2003(1):39－43.

[4] 李蜀瑜. 电气监控组态软件的研究与开发[D]. 西安:西北工业大学,2001:194－195.

[5] 敬新益. 智能建筑中央空调监测计量与控制[D]. 南昌:南昌大学,2006:40－42.

[6] 陈海权. 楼宇空调自控系统的应用研究[D]. 武汉:武汉理工大学,2006:25－29.