基于 Decorator AOP 框架的一卡通管理系统①

Smart Card Management System Based on Decorator AOP FrameWork

李纯玉 张祖平 (中南大学 信息科学与工程学院 湖南 长沙 410083)

摘 要：传统 MVC 模式下横切关注点问题是一个关系系统性能优化的问题。论文分析了 MVC 模式的性能目标，建立了基于装饰器设计模式（Decorator Pattern）精简 AOP 框架的理论模型，实现了权限管理等横切关注点织入方案，并基于实际应用进行了系统架构模式优化，在系统功能与业务功能分离方面得到有效解决，AOP 多重拦截方案得到有效建立。实验与实际运行系统表明，Decorator AOP 框架在实际应用中降低了系统复杂性、提高了组件重用性并优化了业务流程，具有良好的实用性。

关键词： MVC 装饰器模式 多重拦截器 AOP 一卡通

横切关注点就是在多个不同的模块中要实现同一个需求。面向方面编程(Aspect Oriented Programming)技术是鉴于面向对象编程技术在处理横切关注点时的弊端而出现的，它以解决横断现象为出发点，用最小的耦合处理每个关注点，使横切关注点也是模块化的，将贯穿系统横切关注点提取出来，形成方面层并声明方面的织入点，以提高重用率，提高代码的可维护性。

1 引言

在现代软件开发领域，基于 OOP 的 MVC 模式为应用系统的开发环境提供了一种分层的体系结构，即模型层、视图层和控制层，有效解决了系统复杂性与维护性的困难。但是由于面向对象技术在解决跨越多个模块的横切关注点问题上固有的缺陷，使得采用面向对象技术的开发模式，在处理横切关注点问题时具有较大的局限性。

随着科学技术的迅速发展及企业竞争越来越激烈，一卡通管理系统的应用也越来越广泛，用户对系统要求也越来越高，企业需要不断应付用户即时需求变化，虽然当前 MVC 模式为此提供了较好的解决方案，但是在日志管理、权限管理、异常处理等横切方面的可扩展性还不能满足用户需求，系统低耦合性还没有得到完全解决，文献[1]提出的 AOP 编程技术正是为处理横切关注点而出现的。

但是文献[2-3]指出，AOP 是一种新兴编程方法，目前还处在研究和发展阶段，它需要特殊的语言处理，AOP 在 Java 环境下已经拥了较为成熟的框架如 AspectJ 等，但是在.Net Framework 环境中还没公认的最佳解决方案。作者在实际项目开发过程中通过对日志处理、权限管理模块以及 AOP 技术研究分析后，基于对 Struts[2]框架的思考，在.NET Framework 环境下设计出一种能实现 MVC 模式下基于装饰器设计模式(Decorator Pattern)的 AOP 框架，设计了类拦截器模型实现多重拦截，相对于传统使用代理技术实现简单的 AOP 框架，Decorator AOP 框架在提高系统可扩展性方面具有突出的优势，如对日志记录功能扩充记录错误严重级别，又需要记录优先级功能，降低了系统复杂性及耦合度，提高了系统整体性能及执行高效性，并完全分离系统功能和业务功能，使基

① 基金项目:中南大学博士后基金(60873081)
收稿时间:2008-11-04

于面向对象技术的 MVC 模式可以较好的解决横切关注点及其带来的众多问题。论文较全面地分析了这些特点与问题，并基于一卡通的具体实施过程对 Decorator AOP 框架的有效性进行了验证。

2 装饰器模式的AOP框架设计方案

2.1 .Net 平台下 AOP 拦截器的提出及分析

对于传统 MVC 模式下的 Web 应用系统，当用户点击一个链接需要权限验证时，通常在 Controller 层需要接受用户请求，根据用户权限告诉 Model 层执行相应的权限管理业务，最后 Viewer 层根据用户权限显示相应功能界面给用户，可见 MVC 模式下每一层都执行了权限管理，这样系统仍然存在高耦合度。同样，如果系统存在日志管理功能，使用传统 OOP 方法在每一层仍然都需要进行日志处理，高耦合度仍然存在， MVC 模式优点发挥得并不明显。因此，我们把类似于权限管理、日志记录等功能划分为系统横切关注点，利用 AOP 思想处理这些横切关注点。作者将 JAVA 中 AspectJ[4,5]框架的成熟的思想应用到.Net AOP 上，通过对 Struts 的研究，设计了一种类拦截器模型。图 1 的工作原理图说明了拦截过滤方法的过程和原理。

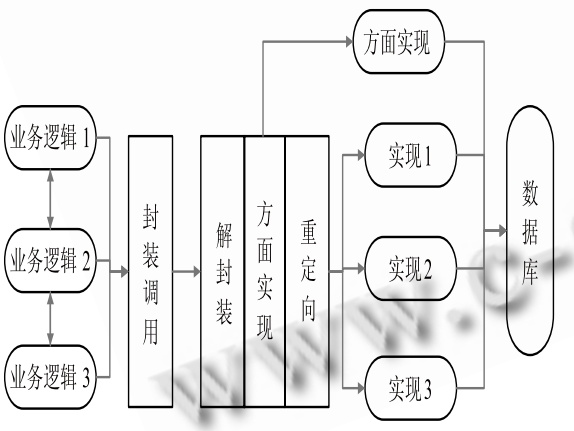

图 1　类拦截器工作原理

该模型在对 MVC 控制器的请求中注入拦截代码，在 Controller 和它的 Action 方法执行的前后执行，促成一些非常棒的封装场景，在其中，能以一种非常明确的声明方式轻松地封装和重用功能。但是，目前 AOP 织入方法还不能实现高效多重拦截，本文在此基础上通过设计 Decorator AOP 框架有效地解决了该难题。

2.2 AOP 织入方式研究与缺点分析

使用 AOP 技术最终要对关注点进行织入，通常织入的方法有两大类[6]：静态织入和动态织入，然而通过分析使用，静态织入方法一般都是需要扩展编译器的功能，将需要织入的代码，通过修改 IL 代码，直接添加到相应的被织入点；或者需要为原来语言添加新的语法结构，从语法上支持 AOP。这背离了简单实用的特点，是不可取的。在.Net 平台上，实现 AOP 的动态织入，主要是靠继承(或对“基接口”的实现)实现代理的，如果一个类不可以被继承，这么代理模式就不成立了，也就是说，如果你用代理方法对一个 sealed 类进行 Mixin 的话(混合以便动态增添新功能)那么它就会失败，因为无法创建动态代理。这是一个很严重的问题，如果非要改动代码来适应动态代理的话就会破坏面向对象的设计，严重的甚至可以带来安全隐患，因此不可以把这种方法应用到大规模项目上，所以设计一种精简 AOP 框架解决以上问题迫在眉睫。

目前常用 AOP 框架的主要优点在于当我们要为一个已经写好的类的方法植入新的处理时，我们不需要对该类的方法实现做修改。只需要实现一个类的拦截代理(如 AspectServiceProxy)，并对类加[Aspect(typeof(AspectServiceProxy))]特性，对要被拦截的方法加 [Interception(true)]特性。最重要的是被拦截的类的调用方式没有任何的改变，例如，Aspect ServiceClass asc=new AspectServiceClass(5)这样的调用可能散布在系统的许多地方。如果被经过 AOP 改造后的类的调用方式发生改变那么就必须修改系统中每一个调用该类的地方，这样造成的代码的变动可能不是我们所期望的，也于 AOP 理念的初衷是不相符合的。当然这个 AOP 框架的缺点也是显而易见的。首先，所有要被代理的类都必须继承自 Context BoundObject，而事实上许多的类可能要实现其他的继承，其中一个办法就是把 ContextBoundObject 作为系统中所有类的基类。其次，就是不能实现方法的多重拦截。图 2 是以日志记录管理为例的类图。

此时可以看到，如果需要相应的功能，直接使用这些子类就可以了。这里我们可以采用类的继承方式来解决了对象功能的扩展问题然而，这种方式却带来了一系列的问题。首先，前面的分析只是进行了一种功能的扩展，如果既需要记录错误严重级别，还需要记录优先级时，子类就需要进行接口的多重继承，这

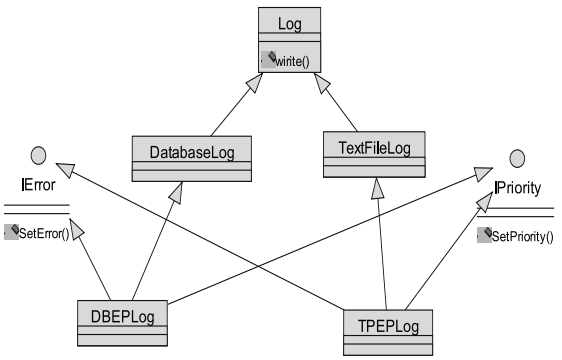

图 2 日志记录类图

在某些情况下会违反了类的单一职责原则，如上图中的 DBEPLog 和 TPEPLog，因此我们设计了 Decorator AOP 框架来解决此问题。

2.3 Decorator AOP 框架设计

由于上述 Realproxy 的效率较低，且被拦截的对象必须继承自 ContextBoundObject，如果这些需要拓展的功能种类繁多，那么势必生成很多子类，增加系统的复杂性，同时，使用继承来实现功能拓展，必须预见这些拓展功能，这些功能在编译时就确定了，是静态的。所以其有自身的局限性。实际上，这些功能需要由用户动态决定加入的方式和时机。设计模式[6]中的装饰器模式(Decorator Pattern)提供了“即插即用”的方法，在运行期间决定何时增加何种功能，所以为了更高效和在实际项目具有更好的实用性，作者提出使用 Decorator 模式以面向接口的方式设计了一个简单的 AOP 框架，并在实际项目里广泛地应用，取得较好的实践效果。

Decorator AOP 框架只包含三个类，见图 3。

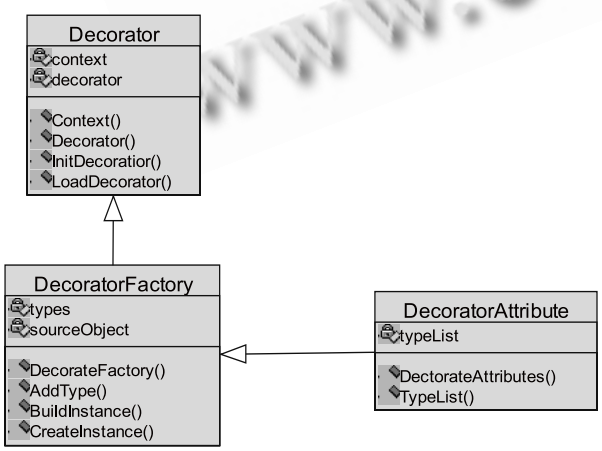

图 3 AOP 框架类图

① Decorate：用来对原来对象的方法、事件进行拦截；

② DecorateFactory：用来构造拦截器的对象链的工厂；

③ DecoratorAttribute：用来对被拦截对象的类施加拦截器标识。

Decorator 模式提供了比继承更加灵活的扩展，通过使用不同的具体装饰类以及这些装饰类的排列组合，可以创造出很多不同行为的组合。解决了主体横切关注点类在多个方向上的扩展功能。这样的 AOP 框架的优点是在于其执行的高效性，几乎没有性能损耗，并且可以实现多重拦截，其对方法和事件的拦截顺序取决于拦截器被 Decorator Attribute 标记的顺序。此框架要求每个被拦截的类和拦截实现同一个接口，并且每个拦截器都需要实现对接口中的每个方法一个至少形式上的拦截。作者将 Decorator AOP 框架应用于一卡通系统验证了该框架的有效性。

3 Decorator AOP在一卡通管理系统中的应用

根据上述对 Decorator AOP 的研究，作者将其应用到广州市鑫奥康科技有限公司第四代一卡通管理系统中，并通过对该系统页面验证与授权来分析利用 Decorator AOP 在 MVC 模式下实现拦截的机制。

一卡通管理系统是指利用先进制造技术和 IT 技术对一卡通，包括发卡，退卡、挂失卡、考勤卡等信息进行管理的信息系统。但是，随着管理信息系统的开发进入成熟阶段，客户对于系统的要求越来越全面，开发人员也开始关注横切关注点实现的优劣，希望能够模块化横切关注点。这就使得传统模式在横切关注点实现方面的不足更加凸显出来，所以作者将以上提出的 Decorator AOP 模式应用于该系统。

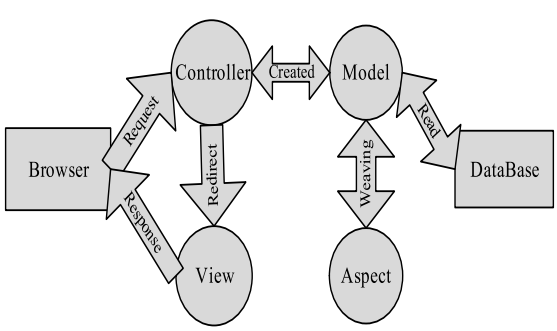

图 4 一卡通管理系统架构图

整个项目的架构设计遵循 Decorator AOP 模式，将视图层、控制层、模型层及持久层分开，其中方面层织入点由控制层拦截并织入，项目架构图如图 4 所示。

在 Decorator AOP 架构下，用户访问的每一个页面在磁盘中并没有一个固定的物理文件，它是通过 Controller 控制数据与视图的组合来生成 HTML 代码，进而向客户端输出。通过复用已有的表单验证授权机制，提高系统可复用性，降低系统复杂度。

在 MVC 中，请求的功能入口是 Controller 相应的 Action 函数，作者在函数执行前去控制请求权限并提供了一个机制对 Action 的 AOP 拦截，这个接口定义如下：

```
Public interface?IActionFilter
{
Void OnActionExecuted(ActionExecuted Context
filterContext);
}
```

作者通过定义 Attribute 来实现拦截的功能，使用 ExceptFilter 拦截器来记录在请求的执行过程中抛出的异常的细节，继承自 ActionFilterAttribute 类型，覆盖其中的适当方法，在 Controller 的 Action 方法调用之前或之后，或者在 ActionResult 处理进回复之前或之后运行代码：

```
Public class ExceptFilterAttribute: ActionFil-
terAttribut
{
  Public override void OnActionExecuted
(ActionExecutedContext filterContext)
{
if(filterContext.Exception != null)
{  //do:log the exception details
}
}
}
```

在 Decorator AOP Controller 中使用过滤器只要在 Action 方法上将其声明为一个属性，或者在 Controller 类本身之上声明即可(在这个情形下，它将运用于 Controller 中所有的 Action 方法)：

```
[ExceptFilter]
Public object Login()
{
viewData[ “ title ” ]= “ Login page ” ;
return View();
}
```

Controller 这个类也实现了 IActionFilter 这个接口，并且也提供了这四个函数的虚拟方法定义。框架内部，在调用 Action 方法的时候同时来调用这些拦截方法，所有的 Action 的调用都在这个类当中被实现，不需要重写 Controller 里这四个虚方法，便完成本 Controller 里面的所有 Action 的拦截。

作者在项目中使用国外基于角色的 MVC 权限控制的方案。自定义了两个 Attribute，分别为：RequiresAuthentication 和 AttributeRequires Role Attribute。通过这两个 Attribute 来可以作用于 Class 和 Method，用户标记哪些 Controller 或 Action 需要登录后，或者需要拥有哪些角色才能执行。如果用户没有拥有访问当前 Controller 或 Action 权限的时候，就会自动被重定向到登录页面去。

Action 过滤器中定义了 Web 请求的走向，针对 Web 请求的访问安全就可以通过该过滤器进行控制，彻底把安全控制代码从控制器中分离出来。如果用户的安全需求发生变动，只需要修改资源权限定义，从而大大提高了系统的可维护性、可扩展性和可重用性。过滤器根据用户权限自动调度合适的控制器，并由控制器返回用户操作的界面，使用户操作界面变得更加友好。

在处理关注点和主逻辑之间的关系上，它们可以各自利用面向方面的设计思想设计，是完全独立的模块，只有在类拦截器中才完成了主逻辑和关注点的交叉。Decorator AOP 模型及方法简单实用，避免了继承的局限性，可以在大规模项目中使用,回答了引言中提出的问题，并且在广州鑫奥康公司一卡通管理系统

中成功实现，证实了它的有效性和实用性。

4 结束语

AOP 技术在近几年其研究与应用呈上升趋势，原因主要是利用传统 OOP 技术在处理横切关注点、降低系统耦合性问题上还没有得到完全解决，不能满足用户日益增长的需求变化。我们在论文中通过对 Java 中 AspectJ 框架成熟思想的研究，分析了传统 AOP 使用代理技术实现简单 AOP 框架的优势与不足，在此基础上论述了 Decorator AOP 框架的设计与实现，并在广州鑫奥康一卡通管理系统中得到有效应用，证明了采用 Decorator AOP 框架具有如下优势：(1)有效消除 AOP 使用代理织入时出现的灵活性差和多子类衍生问题；(2)避免了继承的局限性，可以满足大规模项目得应用需求；(3)系统功能与业务功能分离降低了系统复杂性，提高了组件重用性并优化了业务流程，具有良好的实用性。

参考文献

1 Chen YJ, Li H, Zhang JJ. A management information system based on AOP. Computer Science and Information Technology, 2008,2(1):583－587.

2 Selfa D.M, Carrillo M, Del Rocio Boone M. A Database and Web Application Based on MVC Architecture Electronics. Electronics, Communications and Computers. Puebla, Mexico: IEEE Computer Society Press, 2006:48－48.

3 Murphy GC, Schwanninger C. Aspect Oriented Programming. IEEE SOFTWARE, 2006,2(1):20－23.

4 熊策,陈志刚.AOP 技术及其在并发访问控制中的应用.计算机工程与应用, 2005,41(16):94－96.

5 Kumar A , Kumar R, Grover PS. Towards a Unified Framework for Cohesion Measurement in Aspect-Oriented Systems. Jeanette Hackett,ed. 19th Australian Software Engineering Conference (ASWEC 2008). Perth, Australia:IEEE Computer Society Press, 2008: 57－65.

6 Ortiz G,Bordbar B, Hernandez J. Evaluating the Use of AOP and MDA in Web Service Development. Reda Reda,ed. Internet and Web Applications and Services. Athens, Greece: IEEE Computer Society Press, 2008: 78－83.