基于 AOP 技术的 Web 系统日志管理

Log Management of Web System based on Aspect—oriented Programming

葛 峰 张云华 赵国平 （浙江理工大学 杭州 310018）

摘要：为了把影响多个模块的行为封装到一个单独的可重用模块中，使系统拥有更好的模块化、可扩展性和可维护性，面向方面编程（AOP：Aspect—Oriented Programming）是一种行之有效的方法。本文介绍 AOP 的基本概念并举例说明 AspectJ 在 Web 系统中日志管理开发的具体应用。实践表明，在 Web 系统中应用 AOP 后，系统的核心业务逻辑和日志管理逻辑分开，从而使系统的耦合性大大降低且易于扩展。

关键词：Web 系统 AOP 日志管理 AspectJ 横切关注点

1 引言

当前，Web 系统的应用越来越广泛。随着网络技术的快速发展，Web 系统已经成为了目前最重要的基于文档的分布式系统[1]。在社会的各个领域都发挥不可取代的作用。

日志管理（Log management）在现代软件系统中有着重要的地位，从各种操作系统到一般的应用程序，都能发现有关日志的模块或者功能。它可以记录下系统所产生的所有行为，并按照某种规范表达出来。我们可以使用日志系统所记录的信息为系统进行排错，优化系统的性能，或者根据这些信息调整系统的行为。在 Web 系统中，日志系统是必不可少的功能模块。

2 传统的日志管理实现及其问题

以往我们在进行系统设计的时候都采用 OOP[2]（Object—Oriented Programming 面向对象编程）把系统按对象进行划分，然后用包和类的方法来组织系统。从而实现核心业务的模块化。虽然这种方法取得巨大的成功，优点很明显。但是也存在以下两点不足[3]。

（1）代码交织（Code Tangling）。指一个软件系统中的核心模块可能需要同时与多种非核心需求交互，如此多的需求会导致最终的代码混乱。

（2）代码分散（Code scattering）。指由于非核心关注的模块贯穿于整个核心模块，使得相关的实现也遍布于其中。

由于这些缺点的存在造成系统模块间的耦和度过高，代码重用性不强等问题，从而使开发出来的系统不便于维护和理解。AOP 作为一种全新的设计技术，把贯穿于核心模块的非核心需求（日志、权限管理、容错处理等）提取出来，形成一个单独的横切模块。从而避免上述问题的发生。

3 AOP 技术概述

面向方面编程（AOP）是施乐公司帕洛阿尔托研究中心（Xerox PARC）在上世纪 90 年代发明的一种编程范式。由于软件系统越来越复杂，大型的企业级应用越来越需要人们将核心业务与公共业务分离。AOP 技术正是通过编写横切关注点的代码，即“方面”。分离出通用的服务以形成统一的功能架构。它能够将应用程序中的商业逻辑和对其提供支持的通用服务进行分离，使得开发人员从重复解决通用服务的劳动中解脱出来，而仅专注于企业的核心商业逻辑[4]。

OOP 使用继承、封装、多态等特性建立一种自上而下的对象层次结构来模拟现实社会。但是当对这些对象层次引入公共的行为时，OOP 就显得无能为力了。也就是说 OOP 只适合自上而下的关系，并不适合自左向右的横切关系。如图 1 所示。

AOP 作为一种全新的编程思想，可将 Web 系统中非核心业务模块：日志模块代码抽象成为单独模块，然后通过连接将此单独模块作用于相关的业务核心模

块，从而大大提高了安全模块的重用性和程序的安全性。图1所示的横切关注点就是AOP技术的应用所在。

下面简单介绍一下AOP的术语[5]：

(1) join point(连接点)。是程序执行中的一个精确执行点，例如类中的一个方法。它是一个抽象的概念，在实现AOP时，并不需要去定义一个join point。

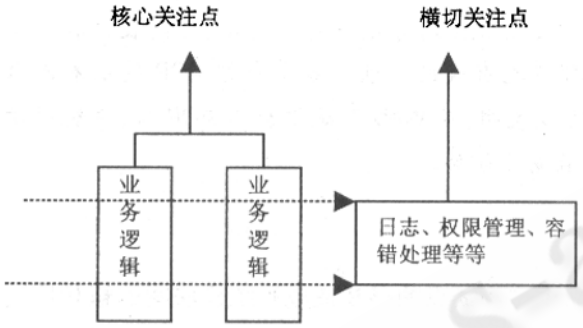

图 1　系统核心关注点和横切关注点

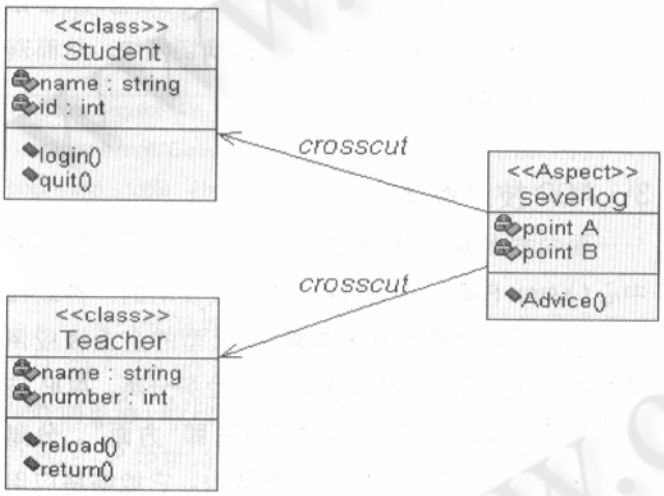

图 2　方面横切多个类的图示

(2) point cut(切入点)。本质上是一个捕获连接点的结构。在AOP中，可以定义一个point cut，来捕获相关方法的调用。

(3) advice(通知)。是增加到point cut(切入点)的可执行代码。实现横切关注点的功能。例如日志记录就是一个通知，切入点捕捉方法的调用，然后动态插入通知，即日志记录代码。

(4) aspect(方面)。pointcut和advice结合起来就是aspect，它类似于OOP中定义的一个类，但它代表的更多是对象间横向的关系。

(5) introduce(引入)。为对象引入附加的方法或属性，从而达到修改对象结构的目的。有的AOP工具又将其称为mixin。

上述的技术特性组成了基本的AOP技术，大多数AOP工具均实现了这些技术。它们也可以是研究AOP技术的基本术语。

4　AOP技术在Web系统日志管理中的应用

4.1　基于AOP技术的日志管理模块设计

下面以一个教务管理系统为例。在系统开发的分析阶段，我们首先识别出了系统的核心关注点和横切关注点。对于核心关注点(录入程序，查询课表等)作为主体部分分别进行了类分析、设计，以期最后形成系统的相应主体模块。对于横切关注点日志记录，我们则分别作了方面分析、设计。我们引入一个日志记录方面(aspect)，该方面需将每个访问服务器的方法的调用定义为切点(pointcut)，在切入点点(方法调用)之前和之后做日志记录，这需要使用目志方面的advice来完成。如图2所示。

4.2　基于AOP技术的日志管理模块实现

AOP的实现，我们需借助施乐公司帕洛阿尔托研究中首席软件科学家Gregor Kiczales及其团队开发的AspectJ实现。AspectJ[6]是Xerox PARC开发的基于Java语言的AOP扩展，是一种用的比较广泛的方面描述语言，它实现了对Java语言的无缝扩展并且有自己的语言构造，aspect、pointcut。AspectJ的工作原理是：方面模块定义后，AspectJ通过自己的编织器进行处理，生成java程序源代码，然后将生成的java代码编译(通常的java编译器即可)生成中间字节码。下面给出服务器端日志记录的部分AspectJ实现代码：

```
package Server;
import org.aspectj;
public aspect ServerLog
{
  public void SendMessage(String message)
```

```
    {
      ServerLog.SendMessage(message);
    }
  //定义切入点,截获所有方法的执行
    pointcut publicMethod: execution(public *.
org.apache.severaction.*(..));
    pointcut logCalls(): execution(Logger.*
(..));
  //定义通知(advice),给出调用方法前后应执行
的日志记录操作
    before(): logCalls()
    {
      SendMessage("begin method:" + thisJoin-
Point.getSignature().getName());
    }
    after(): publicMethods()
    {
      SendMessage("end method:" + thisJoin-
Point.getSignature().getName());
    }
}
```

此 ServerLog 方面定义了 2 个切入点来自动捕捉 org.apache.severaction. 包的所有调用,并形成日志记录。这段代码的功能是:通过定义了这样一个完整的 aspect,当系统调用 org.apache.severaction 的相关方法时,就触发了 pointcut,然后调用相应的 advice 逻辑:在核心操作完后执行写入日志操作。这样,各个子系统的设计者就没有必要关心涉及当前模块的日志记录。我们要让此日志记录 Aspect 影响正常的基于类的代码,只须进行方面重组,我们可以借助 AspectJ 的 ajc 编译器将 aspect 和相关类以单元为单位编译,即可将此 aspect 的 advice 加入到它定义的切入点的方法代码中去。

由上可知,采用基于 AOP 的日志管理开发,结构清晰,易于维护。由于日志管理代码被模块化为独立的单元,作为切面应用到相应的连接点中;当日志管理策略需要修改时,只需变动日志管理模块代码,而无需改动核心业务模块;此外,当添加类或类中添加新的方法时,编程人员不必考虑新的安全问题,也不必为加入横切行为而了解它们,而只需改动 aspect 中的连接点标识,日志管理策略将自动应用于它们,即这种独立的管理策略具有更好的重用性。这种对新代码自动应用 aspect 的特性也是 AOP 的主要优势之一。另一方面,由于日志管理模块与核心业务模块分离,减少了由于代码纠缠而引起错误的可能性,由此也大大提高了程序质量。

5 结束语

实际的软件系统往往都是多方面需求"正交"的系统,AOP 将"方面"概念引入到对象中,以模块化的方式解决横切关注点问题,使得传统的 OOP 中较为棘手的安全、调试、资源管理等系统问题变得容易解决。在 Web 系统日志管理开发中的应用 AOP 技术使系统拥有更好的模块化、可扩展性和可维护性。

参考文献

1 冯玉琳、黄涛、金蓓弘,网络分布式计算与软件工程[M],北京科学出版社,200—252.

2 张海藩,软件工程导论[M],第 4 版,北京:清华大学出版社,2003.

3 张广红、陈平,关于 AOP 实现机制和应用的研究[J],计算机工程与设计,2003(8)14-17.

4 Ramnivas Laddad,刘克科译,利用 AOP 分离软件点[J], Programmer,2002(1):57-60.

5 Bertrand Meyer Object-Oriented Software Construction.

6 The AspectJ Team. Aspect-Oriented Programming with AspectJ at: http://aspect.org.