一种轻量级的权限控制开发框架

郭克华，常亮
（中南大学信息科学与工程学院，长沙 410083）

摘要：传统权限控制机制，对于运行时可能进行的权限修改，不具备可扩展性和可配置性。本文利用配置文件来描述软件需要进行控制的权限，引入代理模式思想，将权限控制模块和软件实现模块本身分开，构造出权限控制映射；通过对配置文件的修改，可以动态改变应用程序需要控制的权限类。实验表明，在不修改软件源代码的情况下，基于代理模式的软件权限控制框架，和传统方法相比，能够使得软件的安全保障更加有扩展性。

关键词：代理模式；权限控制；框架；软件安全
中图分类号：TP391

A Lightweight Development Framework for Permission Control

GUO Kehua, CHANG Liang
(School of Information Science & Engineering, Central South University, ChangSha 410083)

Abstract: Traditional privilege control mechanism has no extensibility and configurability for the runtime change of privilege. In this paper, configuration file is used to describe the permission, and proxy pattern is employed to separate permission control module and software implementation. The permission control class can be dynamically changed by modify the configuration file. In comparison of the traditional approaches, experiment indicates the extensibility of this lightweight development framework for software security.

Keywords: proxy pattern; permission control; framework; software security

0 引言

软件开发中，对用户权限进行控制的应用已经越来越广泛，科学的权限控制方法，保证了资源访问的安全性，成为系统安全的重要保障。因此，对于软件来说，权限控制如何进行开发，显得非常重要。软件权限控制是软件安全领域中的一个重要课题，权限控制的准确性，关系到整个系统的安全运行[1-2]。

目前的软件开发，大都使用面向对象技术。在面向对象的语言中，权限一般使用单独的模块进行处理，权限控制通常存在与被封装的对象中。但是，由于权限处理是一个可能经常被改变的业务逻辑，好的权限控制机制除了需要针对权限进行准确处理，还需要具有较好的可扩展性，对于在运行中临时改变的权限，在不改变源代码的情况下，也能进行有效的处理。

传统权限控制方法可以分为两大类。一类是在具体功能前加入权限操作检验的实现方式，这类方法，能够将权限的粒度控制到具体的业务方法[3]；但是，这种情况下，每个功能类都需要相应的权限检验代码，项目中分布着大量的权限控制代码，程序功能和权限检验混淆在一起，存在紧密的耦合性，扩展修改难度大[4-5]。另一类是利用已有的一些框架，如 Spring AOP[6]等，这种情况下，可以将权限控制模块和业务逻辑模块分开，实现单独开发，很方便地对权限控制模块进行修改，具有良好的可维护性；但是，项目的运行必须依赖于框架本身，脱离该框架，项目则无法运行。

针对以上问题，本文利用配置文件来表达软件需要进行控制的权限，降低修改的成本；引入代理模式思想，将权限控制模块和软件模块本身分开，构造出权限控制映射。这样，在

基金项目：教育部博士点基金(20090162120069)，湖南省科技计划(2009FJ3016)，中南大学博士后基金
作者简介：郭克华，(1980-)，男，副教授，主要研究方向：人工智能、模式识别。E-mail:guokehua@csu.edu.cn

修改权限模块时，只需要通过对配置文件进行，就可以动态改变应用程序需要控制的权限类，使得软件的安全保障更加有扩展性。

1 权限控制

权限主要是指在对某个资源进行某种操作时，对操作者的身份要进行的限制。在一般的系统中，操作者的身份是以用户的形式进行表达的。在权限的概念中，一般有如下几个概念：（1）用户：对资源的操作者身份；（2）功能权限：用户能否执行某个功能。在一些软件中，对于权限控制，还引入了“角色”或者“用户组”等概念，归根结底，其目的是为了将以上三个概念进行更好、更方便的管理。权限控制，最核心的方法是确定：对于某个操作，该用户是否能够进行；具有某种权限的用户称为该功能的合法用户，事先已经被授予了某种权限[7]。

站在软件开发者的角度，主要关心的是权限控制怎样去通过编程来实现。由于权限处理是一个可能经常被改变的业务逻辑，好的权限控制机制除了需要针对权限进行准确处理，还需要具有较好的可扩展性，对于在运行中临时改变的权限，在不改变源代码的情况下，也能进行有效的处理。而传统方法要么是程序功能和权限检验混淆在一起，存在紧密的耦合性，扩展修改难度大，或者项目的运行依赖于具体框架本身，脱离框架项目就无法运行。因此，设计一种轻量级的，并能够在不修改软件源代码的情况下，使得软件的安全保障更加有扩展性的方法，就显得十分必要。

2 代理模式

代理模式[8-9](Proxy Pattern)是一种常见的软件设计模式，它的经典定义是：给某一个对象提供一个代理，并由代理对象控制对原对象的引用。代理模式属于结构型设计模式，一般可以作为与对象交互的接口，这些对象包括：网络连接对象、远程对象、创建一次需要很大开销的对象、需要保护的对象、内存中的大对象、需要附加线程安全的对象、需要在原来的功能上附加或扩展的对象等等。

虽然代理模式可能使得请求的处理速度会变慢，但是能够将代理模块的功能从整个系统中分离出来。代理模式基本结构如下：

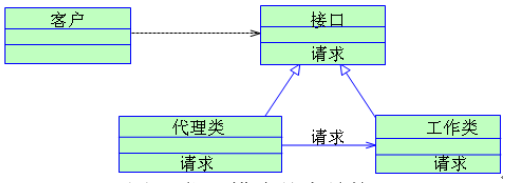

图 1 代理模式基本结构

Fig.1 Structure of Proxy Pattern

从图 1 中可以看出，代理模式有如下特点：

（1）代理类和工作类实现同样的接口。

（2）客户端和代理类打交道。

（3）代理类如果通过验证，则将请求发给工作类。

（4）客户和代理类打交道，其感觉就好像在调用工作类一样，也就是说，代理类实际上对客户透明。

3 框架实现

3.1 基本结构

本框架参考传统的代理模式[8]，提供四个模块：Collaborator，ISubject、ProxySubject 和 RealSubject，结构如图 2 所示：

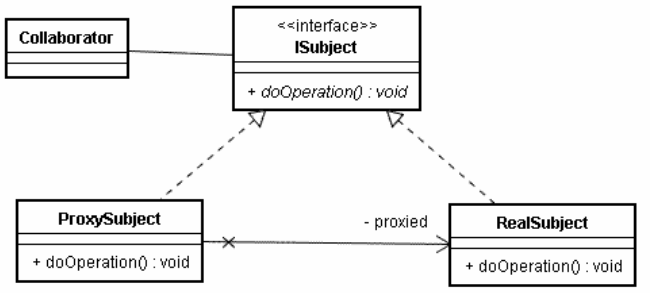

图 2 本框架基本结构
Fig. 2 Structure of Proposed Framework

代理类 ProxySubject 和被代理的类 RealSubject 都实现了相同的接口 ISubject。ProxySubject 负责权限控制，RealSubject 负责实现真正的业务逻辑。在这种方式中，代理类和被代理类之间存在关联。协作者 Collaborator 实际上和 ProxySubject 交互，不管是出于自身的利益还是被代理对象的利益，ProxySubject 在 RealSubject 行为的执行之前和之后都可以加入自己的特定的行为。

因此，在本框架中，权限控制可以单独写在 ProxySubject 模块中，和实际业务逻辑 RealSubject 分开，它们都实现了 ISubject 接口，Collaborator 在调用时，模块对他们来说是透明的。

开发者在尽量多的预测到程序运行过程中可能进行的权限控制之后，在 ProxySubject 中使用自定义的方法来进行权限控制，根据函数返回的值判断，如果权限检查通过，则调用 RealSubject 中相应的处理方法，反之，则抛出异常。

3.2 利用配置文件降低耦合性

为了降低耦合性，在本框架中，ProxySubject 类和 RealSubject 类的耦合信息不直接写在源代码中，需要从外部文件读入到系统中。所以这些信息应独立存储在文件中，由于 XML 文件以其跨平台性，已经成为处理结构化文档的流行工具，因此，本框架采用 XML 文件来描述。

在 XML 配置文件中，配置了权限控制类 ProxySubject 的业务逻辑类 RealSubject 的映射，以下是 XML 文档结构的片段：

```
<authority>
<authorityClass name="ProxySubject">
   <businessLogic>RealSubject</businessLogic>
      ……
</ authorityClass >
……
</ authority >
```

从以上片段中可以看出，在配置文件中，头结点 authority 下可以含有多个子节点 authorityClass，authorityClass 节点代表权限控制类，它的 name 属性是类的路径；authorityClass

节点下包含有多个子节点 businessLogic，节点代表业务逻辑，由此可见，一个权限处理类可以对应多个业务逻辑。

4 实验

以论坛访问的权限控制为例，来测试本框架。某个论坛上有多个功能：如发帖子，发表评论等(简便起见，只列出发表评论功能)，客户端要用这些功能，必须验证一定的权限。传统情况下，我们必须在各个方法中编写权限控制的代码，这样的话，权限控制代码和实际业务代码混合在一起，不好维护；或者利用已有框架，又会对框架具有依赖性。

利用此处框架，就可以将权限验证的模块写成代理。以下是用 Java 语言伪代码为例，展示的各个类的结构：

```
// ISubject
interface Forum {
   public void writeComment();
}
// ProxySubject 代理，负责检查权限
class ForumProxy implements Forum {
private Forum forum; // //配置文件注入
public void writeComment() {
 //查用户权限,如果检查通过,则
//调用 forum.writeComment()，否则抛出异常
}
}
// RealSubject，负责业务逻辑
class ForumOpe implements Forum {
public void writeComment(){
      // 做发表评论的工作
}
}
```

而配置文件结构如下：

```
<authority>
<authorityClass name=" ForumProxy ">
        <businessLogic> ForumOpe </businessLogic>
</ authorityClass >
</ authority >
```

从上面的代码可以看出，权限控制模块完全独立出来了，并且和实际工作模块降低了耦合性。客户类直接和配置文件打交道，方法如下：

Collaborator

```
Forum forum;
//载入配置文件，获取 authorityClass，并组装
//返回 ForumProxy，赋值给 forum 引用
//调用代理类
forum.writeArticle();
```

从本实验可以看出，在代理模式的实现过程中，将每个功能类实现一个相应的代理类，

在代理类中进行权限检查，并利用配置文件，解耦了程序功能和权限检查模块。

5 结论

本文首先对权限控制进行了概述，阐述了权限控制的基本概念和基本思想；接下来阐述了常见的一些权限控制方法，并对这些方法的优缺点进行了阐述。利用配置文件来表达软件需要进行控制的权限，引入代理模式思想，将权限控制模块和软件模块本身分开，构造出权限控制映射。通过对配置文件的修改，可以动态改变应用程序需要控制的权限类。实验表明，在不修改软件源代码的情况下，基于代理模式的软件权限控制框架，和传统方法相比，能够使得软件的安全保障更加有扩展性。不过，如果系统足够复杂，可能带来的问题是代理类太多，这也是将来需要解决的问题。

[参考文献]

[1] 廖俊国, 洪帆, 朱更明, 等. 基于信任度的授权委托模型[J] . 计算机学报, 2006, 29(8) : 1265- 1270.
[2] B. Chess,B.Arkin. Software security in practice[J].IEEE Security & Privacy,2011,9(2):89-92.
[3] Barka E, Sandhu R. Framework for Role- based delegation models[C] In: Proceedings of the l6th Annual Computer Security Application Conference. Washington, DC, USA: IEEE Computer Society, 2000: 168-176.
[4] 沈海波, 洪帆. 访问控制模型研究综述[J]. 计算机应用研究, 2005(6): 9-11.
[5] L.Veiga, P.Pereira,P. Ferreira. Complete distributed garbage collection using DGC-consistent cuts and .NET AOP-support[J].IET Software, 2007,1(6):263-279.
[6] 张献.董威.齐治昌. 基于 AOP 的运行时验证中的冲突检测[J].软件学报,2011,22(6):1224−1235.
[7] E. T. Ueda, W.V. Ruggiero. A systematic mapping on the Role-Permission relationship in role based access control models[J]. IEEE Latin America Transactions, 2012, 1(10):1243-1250.
[8] Huawen Li,Qingjie Wang.Proxy Pattern Informatization Research Based On SaaS[C].IEEE International Conference on e-Business Engineering, USA:IEEE Society, 2009. 518 - 521.
[9] 阎宏. Java 与模式.[M] 北京:电子工业出版社,2002.