一种新颖的面向方面的 Web 应用访问控制方法

邹 玲,黄 杰,贾 焰
ZOU Ling,HUANG Jie,JIA Yan

国防科技大学 计算机学院,长沙 410073
Department of Computer Science,National University of Defense Technology,Changsha 410073,China
E-mail:hj_nudt@yahoo.com.cn

ZOU Ling,HUANG Jie,JIA Yan.Novel aspect-oriented Web application access control method.Computer Engineering and Applications,2007,43(36):110-113.

Abstract: Role-based access control is a traditional software security technology;Web application framework technology comes out continually,such as struts and spring,they decouple Web application by MVC design pattern.It is a challenge to make full use of these frameworks,and implement a flexibly configured,scalable and maintainable access control mechanism.We present a novel access control method based on AOP,reflection,context propagation,XML technology.The method can work with MVC framework seamlessly.It not only makes the access control code be centrally controlled,but also keep Web application loose coupled at the same time.

Key words: access control;Web application;AOP;context propagating

摘 要:基于角色的访问控制是一种传统的软件安全技术;支持 Web 应用开发的框架技术层出不穷,如 struts 和 spring 框架基于 MVC 设计模式对 Web 应用进行了有效地解耦合。在这些框架技术下,如何充分使用这些框架带来的优势,实现一种配置灵活、扩展性强、易于维护的访问控制机制成为一个新的挑战。结合 AOP、反射、上下文传播、XML 技术给出了一种新颖的访问控制实现方法,这种方法能够同基于 MVC 设计模式的框架有机地结合起来,不仅使访问控制代码集中管理,而且在实现访问控制的同时,保持了原有 Web 应用的松耦合结构。

关键词:访问控制;Web 应用;AOP;上下文传递
文章编号:1002-8331(2007)36-0110-04 **文献标识码:**A **中图分类号:**TP311.52

1 引言

随着 Web 技术的快速发展,新的 Web 应用构造技术层出不穷,与此同时,Web 技术在表现力、对应用有效解耦合方面独有的优势使其被广泛地应用于系统监视、控制、管理等多种领域,越来越多的金融、电信、国防、电子政务等系统基于 Web 技术进行开发。基于 MVC 设计模式的 Web 应用架构技术独领风骚,服务器端 Model 2 体系结构和浏览器端 Ajax 技术的发展无不展示出 MVC 设计模式在 Web 应用体系结构中的重要性。与此同时,基于 MVC 的服务器端 Web 框架技术迅猛发展,struts、spring、Web works 等一大批软件框架的出现对于规范 Web 应用的体系结构,快速构造 Web 应用提供了强有力的支持。这些框架有着各自的特点,它们被应用的问题领域尽管相似,但它们的具体实现和解决问题的侧重点有所不同。struts 侧重于控制器和视图的支持,对业务逻辑层的模型支持不够。spring 对 MVC 的各个部分都有较好的支持,尤其是对业务逻辑层构件的管理具有其它框架无法比拟的优势。尽管 spring 对于控制器和视图也有十分灵活的支持,但对于大多数 Web 应用来说,spring 的灵活性却成为易用性的负担。越来越多的开发者开始同时使用 struts 和 spring 两种框架开发新的 Web 应用,这种结合充分发挥了两个框架的特长,为快速高质量的构造复杂 Web 应用提供了重要的技术保证。

另外,Java 语言拥有强大的反射能力,为 AOP 技术的发展提供了技术基础。在此基础上发展起来的多种 AOP[1]技术被 spring 的开发者有效地集成到了业务逻辑层的框架之中,应用的开发者们可以有效地利用这种技术灵活地处理诸如事务、安全、并发控制等复杂的企业计算问题。

尽管 Web 应用发展迅速,但 Web 应用中的访问控制问题[2]仍然是一个永恒的话题。如何充分利用新出现的框架技术实现一个灵活、扩展性强、结构清晰、易于维护的访问控制机制成为一个新的挑战。

2 基于反射技术的 RBAC

在 Web 应用中,基于角色的访问控制[3](RABC)目标是对 Web 应用提供的功能进行控制,Web 应用的功能由对象或方

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.90104020);国家高技术发展计划(863)(the National High-Tech Research and Development Plan of China under Grant No.2001AA113020);国家重点基础研究发展规划(973)(the National Grand Fundamental Research 973 Program of China under Grant No.G1999032703)。

作者简介:邹玲(1986-),女,主要研究领域为面向方面的程序设计方法,Web 框架技术;黄杰(1976-),男,博士,助理研究员,主要研究领域为分布构件,卫星导航定位技术;贾焰(1962-),女,博士,教授,博士生导师,主要研究领域为大规模事务处理,分布构件技术。

法构成。在运行时进行 RABC 的前提是了解运行时的程序信息,代码中对象和方法的信息可以通过两种方式获得,一种通过硬编码完成,另一种则可以通过更加高级的反射机制提供。前者工作量大,访问控制代码分散在业务逻辑代码中,难于维护,易于出错,不利于提高系统的扩展性和灵活性。后者需要程序的元模型信息,Java 语言的 class 文件中存储了这样的信息,并提供了一组反射 API 用于访问程序的元模型。

起初,将反射的概念引入计算机领域的动机是希望计算系统能够根据自身状态以及外界条件的变化随时调整自身以达到最理想的状态[4]。而与反射理论相关的开放实现技术和元程序设计技术都为计算机系统能够更好地实现自省和自我调整提供了具体的研究方法。在反射系统中,用于描述系统本身的数据结构被称为系统的元数据(Meta-data),它是反射实现的基础。反射系统就是在系统运行时通过监视系统的行为和状态,维护正确的元数据,从而达到根据环境和应用程序的需要对系统进行优化和更新的目的。

在探索一种新的 Web 应用 RBAC 过程中,并不打算通过软件元模型或者元数据的反射对运行的 Web 系统进行优化和更新,而是通过元模型让软件很好了解自身的执行情况,并根据用户的安全配置和运行时的角色对 Web 应用程序代码进行访问控制。另外,对于一个 Web 应用可能会有多个用户,每个用户的角色也有所不同。用户的改变被认为是频繁的,而系统中的角色划分却是相对稳定的,在很长一段时间内,系统中的角色不会发生较大的变化,尽管角色和系统权限的关系复杂,但却是相对稳定的。根据上述分析,将系统中用户和角色的关系存入关系数据库以便于用户的更新,用户角色的关系定义如下:

```
create table t_user
(
username varchar(32) not null,
passwd varchar(32) not null,
…
);
create table t_user_role
(
username varchar(32) not null,
role varchar(32) not null
);
```

用户和权限的关系复杂,相对稳定,同时还需要易于维护,XML 为这种需求提供了一种非常便利的描述。XML 的自描述和扩展能力可以很好的被用于定义角色和权限之间的关系,在系统的角色变化时也很容易使用工具进行维护。“方法”是能够表达功能的最小元模型单位,可以将方法看作是系统权限的最小单元,只要说明角色和可以访问的方法,就定义了一个角色拥有的权限。基于该技术的角色描述如下(security.xml):

```
<role>
 <administrator description="系统管理员">
  <privilege>logic.ControlLogic.shutdown</privilege>
  <privilege>logic.ControlLogic.cutout</privilege>
  <privilege>logic.ControlLogic.stop</privilege>
  <privilege>logic.ControlLogic.switchpb</privilege>
  <privilege>logic.ControlLogic.killall</privilege>
  <privilege>logic.setting.UserRoleBean</privilege>
  <include>operator</include>
 </administrator>
 <operator description="操作员">
  <exclude>administrator</exclude>
  <privilege>logic.ControlLogic.startup</privilege>
  <privilege>logic.ControlLogic.cutin</privilege>
  <privilege>logic.ControlLogic.start</privilege>
  <include>normal</include>
 </operator>
 <normal description="普通用户">
  <exclude>operator</exclude>
  <privilege>logic.*</privilege>
 </normal>
</role>
```

其中 administrator、operator、normal 为系统的角色,logic.ControlLogic.shutdown 为方法的全路径名,其中包括所属的包、类以及方法自身的名称,logic.* 是一个表达式,它表示所有 logic 包中的方法,<include/>和<exclude/>标记分别表示角色权限的包含和拒绝关系。

在上述设计的基础上就可以很好地利用 Java 语言的反射机制对 Web 应用的功能进行访问控制。进行访问控制的示意代码如下:

```
String className=target.getClass().getName();
String methodName=method.getName();
String fullName=className + "." +
                methodName;
String[] roles=SecurityContext.getRoles();
if(checkRole(roles[i],operation))
{
      return;
}
else
{
throw new NoPrivilegeException();
}
```

为了能够捕获 NoPrivilegeException 异常并在视图中通知 Web 应用的访问者,需要在 struts 的配置文件中增加全局异常处理的相关内容:

```
<global-exceptions>
    <exception
     key=""
     type="acl.NoPrivilegeException"
     path="/jsp/acl/noprivilege.jsp"
    scope="request"/>
    …
</global-exceptions>
```

3　业务逻辑层面向方面的访问控制方法

在基于 struts 和 spring 的 Web 应用架构中,spring 管理业务逻辑层的模型构件,只要对业务逻辑层构件进行有效、灵活地控制就完全可以实现对 Web 应用的访问控制。在 Web 应用的同一个视图上,可能会组织有多个功能,这些功能的权限应当独立控制,如果通过视图和控制器进行访问控制管理,不仅粒度过粗,而且不能对后台功能和视图有效解耦合,在一定程度上影响了 MVC 设计模式对 Web 应用的解耦合效果。因此,这里提出了一种新颖的访问控制实现方法,这种访问控制方法

在模型层实现，对业务逻辑层代码进行基于方法的细粒度控制。

另外,在传统的访问控制方法中,访问控制代码被分布在应用代码的各个角落，相同的控制代码在应用中反复出现,为代码的开发增加了一定的工作量,并且在访问控制代码发生变化时需要对源代码的很多地方进行修改。AOP 技术通过面向软件方面的编程可以将分布在软件中的方面特性集中统一管理,访问控制恰恰是软件安全的一个方面。基于构件的软件开发技术使得软件代码被有效的解耦合和提高代码的复用程度。在设计一种新颖的访问控制方法时,提出将访问控制作为一种面向方面的构件进行实现和管理。Spring 框架为 AOP 和软件构件化都提供了强有力的支持。这里提出的访问控制技术将基于 spring 来实现。

为了能够对方法进行访问控制管理,需要在访问一个方法之前进行合法性检查。对于 AOP 这里需要实现一个“咨询”,这个咨询在业务逻辑方法之前被执行,如果不能通过检查,则抛出异常,阻止访问继续执行。

```
public class AclAdvice
        implements MethodBeforeAdvice
{
public void before(Method method,
                   Object[] args,Object target)
      throws Throwable
{
       …
       if(checkRole(roles[i],operation))
       {
           return;
       }
       else
       {
           throw new NoPrivilegeException();
       }
}
 …
}
```

checkRole 通过访问 security.xml 在内存中的 DOM 进行访问控制检查。security.xml 可以通过 Web 应用的 servlet 上下文侦听器(servlet context listener)加载到内存,并形成 DOM 对象。

```
public class SecurityContextListener
        implements ServletContextListener
{
public static Document document_=null;
public void contextInitialized
        (ServletContextEvent event)
{
      ServletContext servletContext=
                  event.getServletContext();
      String securityConfigPath=
          servletContext.getRealPath("/WEB-INF/security.xml");
      DOMParser parser = new
         org.apache.xerces.parsers.DOMParser();
         parser.parse(securityConfigPath);
         document_ = parser.getDocument();
}
…
}
```

对于访问控制的 AOP 实现,除了定义咨询代码,还需要指明哪些方法需要进行控制,特别需要指明的是登录代码和访问控制代码本身不应包含在受控代码之中。为了请求 spring 的 AOP 框架对某些方法进行切入,需要定义 AOP 的切入点,与此同时,还需要通过动态代理来实现方法的切入。通过 spring 的配置文件 applicationContext.xml 就可以声明“切入点”和“咨询”。

```
<bean id="proxyCreator"
class="org.springframework.aop.framework.autoproxy.BeanNameAuto-
ProxyCreator">
      <property name="beanNames">
          <list>
              <value>controlled_*</value>
          </list>
      </property>
      <property name="interceptorNames">
          <list>
              <value>aclAdvice</value>
          </list>
      </property>
</bean>
<bean id="aclAdvice" class="acl.AclAdvice"/>
```

id 为 aclAdvice 的构件声明了一个 AOP 的咨询类,id 名为 proxyCreator 的构件为所有 id 匹配“controlled_*”的构件创建动态代理,并通过代理对它们进行方法切入,在切入点咨询构件 aclAdvice 来进行访问控制。

这种基于 AOP 的访问控制方法配置灵活,代码被构件化,与应用的业务逻辑相对独立。通过 AOP 实现 Web 的访问控制,在很大程度上减少了代码开发的工作量,提高了软件代码后期的可维护性。

4 Web 应用中角色上下文的传播

业务逻辑层同 Web 层的分离有利于 Web 应用核心功能和视图层的解耦合。在 Web 层,用户的角色信息在登录后需要被放置在会话中保存,才能够跟踪用户所拥有的角色,然而如果 spring 所管理的业务逻辑无法访问 Web 上下文，也就无法获取 Web 会话中的登录角色信息。如何跨越 Web 层和业务逻辑层的屏障传递角色信息,而不破坏这种松耦合的结构对于有效实现我们提出的基于 AOP 的 Web 访问控制方法十分重要。

为了达到上述目标，将 Web 应用的上下文划分为请求级上下文和线程级上下文[5]。通过 HttpServletRequest 可以访问的上下文,称为请求级上下文。请求级上下文在 Web 层可以通过 struts 的 action 进行访问。为了将角色信息传递到 spring 的 AOP 咨询构件,需要设计一个线程级上下文。

```
public class SecurityContext
{
private static ThreadLocal threadLocal_ =
                    new ThreadLocal();
public static void setRoles(String[] roles)
{
      threadLocal_.set(roles);
}
public static String[] getRoles()
{
      if(threadLocal_.get() ! = null)
```

```
        {
            return (String[])(threadLocal_.get());
        }
        else
        {
            return new String[0];
        }
    }
}
```

线程级上下文通过线程本地存储技术保存角色信息,请求的处理实体为 servlet 和 jsp 容器中的某个线程,AOP 进行访问控制咨询时可以通过线程级上下文获取角色信息。

这个过程还需要能够将请求级上下文转换为线程级上下文才能真正实现。在提出的方法中,使用 servlet 的过滤器机制可以截获请求级上下文并将其转换为线程级上下文,以便为 AOP 的咨询提供用户角色信息。

```
public class SecurityFilter implements Filter
{
    public void doFilter(ServletRequest request,
                         ServletResponse response,
                         FilterChain chain)
    throws IOException,ServletException
{
    HttpServletRequest httpRequest=
                (HttpServletRequest)request;
    if(httpRequest ! =null)
    {
        String[] roles=
        (String[])httpRequest.getSession().getAttribute("roles");
        SecurityContext.setRoles(roles);
        chain.doFilter(request,response);
        SecurityContext.setRoles(null);
    }
}
...
}
```

这种基于请求级上下文和线程级上下文的结构,被灵活地用于在 Web 应用和业务逻辑层传递应用上下文,同时这种结构不破坏原有的软件体系结构,对于减少软件的耦合度,灵活地提供面向方面的上下文信息提供了有效的支持。

5　结束语

随着 Web 应用中基于 MVC 设计模式的框架技术发展,诸如 struts、spring 等框架驱动的管理软件给 Web 应用的开发带来了新的思路,本文基于上述框架,结合使用 AOP、反射、上下文传播、XML 等技术提出了一个配置灵活、扩展性强、易于维护的访问控制实现方法。该方法在对业务逻辑层构件方法访问之前进行权限检查,实现了一种细粒度的访问控制,通过 XML 可以灵活地配置角色和权限之间的关系;访问控制的代码被作为构件实现,不破坏原有 MVC 设计模式的松散耦合结构,对于软件的易维护性具有重要意义。

参考文献:

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

[2] 张静,赵雷,杨季文.一个基于 Web 方式的访问控制的解决方案[J].苏州大学学报:工科版,2006,26(2).

[3] 史永昌,陈和平.基于角色和 Web Service 的访问控制方法应用研究[J].武汉科技大学学报:自然科学版,2006,29.

[4] Fabio Kon,Fabio Costa,Gordon Blair,et al.The case for reflective Middleware[J].Communications of ACM,2002,45(6).

[5] Object Management Group.CORBA 3.0 Specification[S].U S,OMG,2001.

(上接 15 页)

Pattern Analysis and Machine Intelligence,1991,13(3):252-264.

[2] Martinez A M,Kak A C.PCA versus LDA[J].IEEE Trans on Pattern Analysis and Machine Intelligence,2001,23(2):228-233.

[3] Belhumeur P N,Hespanha J P,Kriegman D J.Eigenfaces vs Fisherfaces:using class specific linear projection[J].IEEE Trans on Pattern Analysis and Machine Intelligence,1997,19(7):711-720.

[4] Turk M,Pentland A.Eigenfaces for recognition[J].Journal of Cognitive Neuroscience,1991,3(1):71-86.

[5] Zuo W,Zhang D,Wang K.Bidirectional PCA with assembled matrix distance metric for image recognition[J].IEEE Trans on Systems,Man,and Cybernetics-Part B:Cybernetics,2006,36(4):863-872.

[6] Yang J,Yang J.From image vector to matrix:a straightforward image projection technique—IMPCA vs PCA [J].Pattern Recognition,2002,35:1997-1999.

[7] Yang J,Zhang D,Frangi A F,et al.Two-dimensional PCA:a new approach to appearance-based face representation and recognition[J].IEEE Trans on Pattern Analysis and Machine Intelligence,2004,26(1):131-137.

[8] Wang L,Wang X,Zhang X,et al.The equivalence of two-dimensional PCA to line-based PCA [J].Pattern Recognition Letters,2005,26:57-60.

[9] Zuo W,Zhang D,Yang J,et al.BDPCA plus LDA:a novel fast feature extraction technique for face recognition[J].IEEE Trans on Systems,Man,and Cybernetics-Part B:Cybernetics,2006,36(4):946-953.

[10] Liu C.Gabor-based kernel PCA with fractional power polynomial models for face recognition[J].IEEE Trans on Pattern Analysis and Machine Intelligence,2004,26(5):572-581.

[11] Liu C.Capitalize on dimensionality increasing techniques for improving face recognition grand challenge performance [J].IEEE Trans on Pattern Analysis and Machine Intelligence,2006,28(5):725-737.

[12] Nanni L,Maio D.Weighted sub-Gabor for face recognition[J].Pattern Recognition Letters,2007,28:487-492.

[13] Samaria F,Harter A.Parameterisation of a stochastic model for human face identification [C]//Proceedings of the 2nd IEEE Workshop on Applications of Computer Vision,1994.

[14] Philips P J,Wechsler H,Huang J,et al.The FERET database and evaluation procedure for face recognition algorithms [J].Image and Vision Computing,1998,16:295-306.