文章编号：1005-8451（2006）11-0004-04

基于RBAC的B/S权限管理的研究与应用

马 亮，罗省贤
（成都理工大学 信息工程学院，成都 610059）

摘 要：介绍基于角色的访问控制模型的理论基础与特点，结合宽带增值业务管理系统的开发，阐述系统在权限管理方面的设计思路和实现方法，并对该系统的用户识别和权限验证进行了研究。

关键词：基于角色的访问控制；权限管理；J2EE；模型—视图—控制器
中图分类号：TP39 文献标识码：A

Research and application on B/S System permission management based on RBAC

MA Liang, LUO Sheng-xian
(School of Information Engeneering, Chengdu University of Technology, Chengdu 610059, China)

Abstract: It was introduced the theory and features of RBAC (Role-Based Access Control) model which was based on the development of the Management System for broadband increment business, described the design and implementation in permission management, researched on authentication of user and permission.

Key words: RBAC; permission management; J2EE; model-view-controller

在许多应用系统中，权限管理模块对敏感资源的保护有着至关重要的作用。基于Web的管理信息系统的每一个角落都会涉及到权限问题。RBAC模型作为一种实用的权限存取控制策略，在用户层和

收稿日期：2006-05-06
作者简介：马 亮，在读硕士研究生；罗省贤，教授。

```
<action name="loginForm" path="/loginAction"scope="request" type="user.LoginAction" input="/index.jsp ">
  </action>
</action-mappings>
.....
</struts-config>
```

从此配置文件中可以很清晰看出，本系统登录功能模块主要流程为：用户进入index.jsp页面，输入用户名与密码后向Web服务器提交登录验证请求（Login.do），由LoginAction负责将前端用户发出的请求进行处理，实现持久层逻辑的调用，并将处理结果返回给LoginAction，其中需要交给user包中Login.java的实例进行处理，并将LoginFrom.java中捕获到的网页表单数据传递进去。如果验证通过，进入main.jsp页面，否则进入loginWrong.jsp页面。

5 结束语

经过初步试运行，从数据准确性、安全性、使用方便与灵活性、易维护性、系统结构的合理性、功能的完备性、软件可靠性及资源共享的可行性等方面进行了测试，系统都表现了良好的特性。该合同管理信息系统主要定位在转制的科研院所以及大中型科技型企业。

通过该项目的实施，我们得到如下结论：采用Struts可以简化遵循MVC设计模式的Web应用的开发工作，很好地实现代码复用，使开发人员从繁琐的工作中解脱出来，快速开发具有很强的可扩展性的Web应用。基于Struts的CMIS基本满足了日常合同管理的需求，使企业合同管理工作向信息化和规范化方向发展。

参考文献：

[1] Cay S.Horstmann, Gary Cornell.Java2 核心技术（Core Java2）[M].程 峰，黄若波. 北京：机械工业出版社，2004.
[2] 方葆青. 如何进行合同管理[M]. 北京：北京大学出版社，2003.
[3] 曹广鑫，王谢华，王建凤，等. Struts数据库项目开发宝典[M]. 北京：电子工业出版社，2006.

权限层增加了一个角色层，实现了用户与访问权限的逻辑分离和构造角色之间的层次关系，从而方便了数据和系统资源的安全管理。

本文结合宽带增值业务管理系统的开发，介绍一种基于角色控制权限管理的设计和实现方法。

1　RBAC基本思想

权限问题究其本质是一个“主体对于客体做什么操作”的问题，其中主体代表了可以向应用系统发出应用请求的任何实体。客体代表了权限系统中需要被保护的资源，显然，客体（资源）+操作的组合就代表了权限。

RBAC模型引入角色将用户和访问权限逻辑分开。首先根据用户所在岗位和工作职责设置角色，并对角色授予相应的访问权限，然后再为用户分配角色，并通过会话激活角色。RBAC基本模型如图1所示。

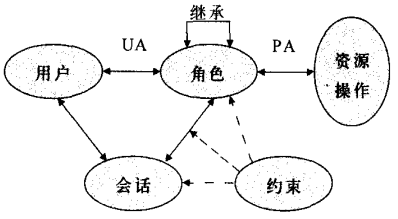

图1　RBAC基本模型

2　权限管理模型设计

2.1　设计目标

根据系统的规模和实际的需求，在设计权限管理系统时主要考虑以下几个因素：（1）直观性原则。因为系统会由最终用户来使用，权限分配直观并易理解就显得比较重要，除了功能上的需要，更主要的是这种权限分配比较直观；（2）简单性原则。包括概念上、意义上和功能上的简单。试图用一个权限系统解决所有的权限问题是不现实的。设计中将常常变化且“定制”特点比较强的部分判断为业务逻辑，而将“通用”特点比较强的部分判断为权限逻辑，权限系统就是基于这样的思路设计和实现的。（3）可扩展原则。采用可继承的方式实现扩展性。角色组的继承概念使得权限以组方式定义的同时也有效地避免了角色重定义。

2.2　权限粒度的问题及设计原则

权限逻辑的设计应配合业务逻辑，即权限管理的目标是为系统业务逻辑提供服务。

权限粒度可分为粗粒度和细粒度。粗粒度仅考虑对象的类别，不考虑对象的某个特定实例；细粒度则需要考虑具体对象的实例。许多细粒度的权限问题因其太独特而不具备通用意义，因此，应该被理解为是“业务逻辑”的一部分，在权限管理的架构设计之中不予过多考虑。即权限逻辑的设计原则可归结为：“系统只提供粗粒度的权限控制，细粒度的权限被认为是业务逻辑的职责”。

3　权限管理实现方法

宽带增殖业务管理系统作为一个大型的管理信息系统，人机交互界面设计的关键是使用户与计算机之间能够准确地交流信息。前台是用户和计算机交流信息的界面，针对前台的权限管理负责授权用户对信息的访问，也负责非授权用户的试用等等；后台是信息处理的中心，针对后台的权限管理主要负责系统管理员对角色的定义、用户的授权和信息的处理。

3.1　权限定义与配置

宽带增殖业务管理系统包括客户管理、内容管理、业务管理、供应商管理、设备管理和系统管理等子系统。系统采用MVC（Model-View-Controller）设计模式，并遵循J2EE体系规范。数据库和服务器分别采用Mysql和Apache Tomcat。

各个子系统又包含了多个子功能模块，所有子模块构成了系统受保护的资源，我们采用对子模块编码的方式来标识各子模块。

在对各子系统模块编码时制定如下规则：

（1）以1个4位十进制数来描述某子系统及其下属的模块。例如，对于客户管理中的客户申请受理模块，它的编码是1001。最高位的1代表第1个子系统（即客户管理子系统），后3位代表客户管理子系统中的第一个模块（即客户申请受理模块）；

（2）系统对模块的操作定义了4种通用操作：浏览、增加、修改和删除，分别用0、1、2、3编码表示。例如，对于客户管理中的客户申请受理模块授予删除权限，相应的权限定义编码为10013。

3.2　角色管理的实现方法

角色是某类权限集的抽象，由最终用户根据实际的业务活动及业务组织情况自行定义，系统提供

角色定义工具，实现角色的权限分配，完成角色到权限的映射。

3.2.1 角色存储的数据结构

在给用户分配角色时，考虑到角色之间的互斥关系。在角色管理模块中，角色存储的数据结构被设计成一个无向图。图2中的节点代表角色的标识RoleID（惟一标识角色的编号），图中的边存储角色的动态互斥关系和静态互斥关系。动态互斥指的是同一个用户会话期间不能同时激活两个角色；静态互斥是指角色之间的冲突。如果两个角色之间存在互斥关系，联系这两个角色的边的值设置为1，否则设置为0。例如，某个用户的角色包括Role1，Role2，Role3和Role4，且角色Role1和角色Role2，角色Role2和角色Role4，角色Role2和角色Role3之间存在静态互斥，则相应的角色存储数据结构如图2所示。

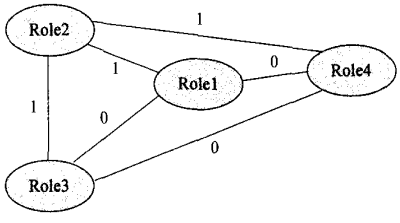

图2 角色存储的数据结构

3.2.2 角色互斥的处理

在分配角色的时候，权限管理系统提供的用户定义工具会遍历整个角色图，找出存在互斥的角色并将其删除。以图3为例，在删除过程中可能有多种选择，但是删除角色冲突时应遵守的一个重要原则是尽可能保留用户最多的角色。对于图2中的角色互斥关系，删除存在冲突的角色后，其结果如图3所示。

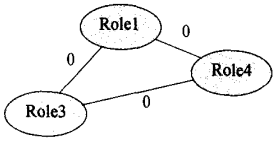

图3 删除冲突角色后的数据结构

在完成静态角色互斥定义后，就可以为该用户的角色定义动态互斥。如果两个角色之间存在动态互斥关系，联系这两个角色的边的值设置为1，否则设置为0。以图3为例，增加角色Role1和Role3，Role1和Role4之间的动态互斥后，相应的角色存储数据结构如图4所示。

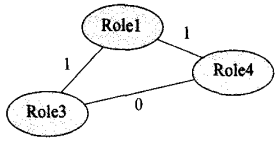

图4 增加动态互斥关系后的数据结构

为一个用户配置好角色后，当该用户登录系统时，系统将遍历整个角色图，根据角色的动态互斥关系，要求用户激活本次会话的角色。以图4为例，Role1和Role3，Role1和Role4之间存在动态互斥，当用户选择激活Role1和Role3时，系统在遍历整个图后，会提示用户Role1和Role3不能在同一个会话中激活，要求用户选择其中之一。

用伪代码描述的角色冲突检测算法如下：

输入：用户访问请求（用户u）。

相关函数：

getRoles（u），返回用户u对应的角色集r；

traversal（r），遍历角色图，返回所有角色图的边集合；

getExlRoles（r,exls），返回冲突角色集。

输出：角色冲突集(exlRoles)。

```
{
  exlRoles = null;
  e = null;
 // 获取用户 u 对应的角色集
  r = getRoles ( u );// 建立用户角色的数据结构
  // 从角色集中获取角色冲突集
  if ( r.size > 1 )// 如果用户的角色个数大于 1
    e = traversal (r);
 for (each ∈ e ){
       if (e.value = 1 )
            exlRoles = getExlRoles ( r,exls);
 }
  return exlRoles;
}
```

系统数据访问权限关系如图5所示。

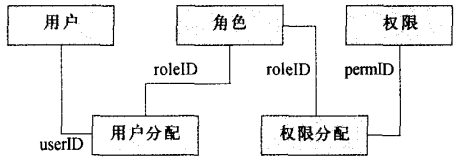

图5 系统数据访问权限关系

用户的所有角色被存储在用户分配表中，用户

分配表通过roleID对应用户的角色，在角色表中，通过roleID对应权限分配表中的permID，在权限分配表中以permID对应权限表中的permCode。在角色表中存储了与这个角色存在静态和动态互斥的角色，在处理角色互斥的过程中，系统从角色表读出与这个角色互斥的角色，并在用户角色存储的数据结构中删除相应的结点。

3.3 用户访问控制的实现方法

为了系统能具备良好的扩展性和复用性，系统设计采用了MVC模式，即把一个应用的输入、输出、处理流程按照Model、View和Controller的方式进行分离，这样，一个应用被分成3层：模型层、视图层和控制层。模型（Model）：负责业务流程/状态的处理以及业务规则的制定，是MVC的核心；视图（View）：代表用户交互界面，也就是展现信息处理结果的视图界面；控制器（Controller）：接收用户的请求，将模型与视图匹配在一起，共同完成用户的请求。

系统权限验证是基于表单方式通过Tomcat服务器来处理的。在Tomcat应用配置文件Web.xml中，对Sererlet的映射配置加入权限编码参数，以控制系统对Servlet使用。在J2EE体系中，Servlet起着MVC模式中控制器的作用，客户页面提交信息给Servlet，Servlet将信息转发给对应的实体类处理。对Servlet进行控制也就实现了对具体用户的访问控制。

当用户登录系统并提交表单后，在相应的Servlet中会生成AccessController类的实例对象，并调用AccessController.CheckPermission方法读取用户Session中的权限信息，判断该用户是否具备访问权限，并将权限检查结果返回至Servlet。

下面给出AccessController.CheckPermission权限验证伪码。

```
输入：（1）会话session；
      （2）用户user；
      （3）权限集perm；
输出：用户权限判断结果flag，如果为真，
      表明用户具有该权限，如果为假，
      表明用户没有该权限；
// 获取用户的会话并得到用户名
HttpSession session = request.getSession();
UserBean user = new UserBean ();
User = session.getAttribute("user");
ArrayList perm = new ArrayList();
// 从会话对象中获取所有关于用户的权限
perm = session.getAttribute("perm")
if (perm空) {
 flag=false;
 return();
 }
else if (perm不空) {
 for (perm中每一个权限) {
    if (perm中包含与输入相同的权限编码) {
flag = true;
break;
}
```

系统通过上述权限验证，控制用户被赋予的不同角色的操作权限。

4 结束语

本文采用比较成熟的基于角色的访问控制模型，结合J2EE框架技术，应用MVC设计模式设计并实现了一种以模块为中心的权限控制信息管理系统，希望能对基于权限管理的系统开发有所借鉴。目前，对于多个应用服务采用一个公共认证授权服务器的情况[4]，为了避免安全成为性能的瓶颈，在集中式安全管理环境中对授权模块可以采用分层控制的策略，根据资源类别或资源的敏感程度分层，同时还需要对权限粒度进一步研究。此外，访问控制模型基本上是静态权限模型，不能很好地支持动态的访问权限管理。诸如工作流的权限管理就需要采用TBAC基于任务的访问模型，以满足不同的权限需求。

参考文献：

[1] Ravi Sandhu,Venkata Bhamidipati,Qamar Munaver.The ARBAC96 model for role-Based administration of roles[J].ACM Transactions on Information and System Security, 1999, 2 (1): 105—153.

[2] Michael E. Shin, Gail-Joon Ahn. 基于角色访问控制的UML表示[J]. 非程序员, 2001 (25): 20—22.

[3] [美]Hans Bergsten. JSP设计[M].（第3版）北京：中国电力出版社，2004.

[4] 段云所.信息安全概论[M]. 北京：高等教育出版社，2003.

[5] 裘　炅，谭建荣，张树有，等. 应用角色访问控制的工作流动态授权模型[J]. 计算机辅助设计与图形学学报，2004，16 (7): 992—998.