# 基于基本 RBAC 模型的权限管理框架的设计与实现①

吴 波，王 晶

(北京邮电大学 网络与交换技术国家重点实验室，北京 100876)
(东信北邮信息技术有限公司，北京 100191)

**摘 要**：基于角色的访问控制(RBAC，Role Based Access Control)最直观的描述就是权限被授予角色，角色被授予用户，权限是访问资源的唯一凭证，用户与权限没有直接关联。将采用基本 RBAC 模型以及 JavaEE(Java Platform, Enterprise Edition，Java 企业级应用)的一些主流选型，设计并实现由用户登录模块、各种管理操作模块组成的具有用户管理、角色管理以及权限管理功能的框架结构。

**关键词**：RBAC；JavaEE；权限管理

## Design and Implementation of Privilege Management Framework Based on Core RBAC Model

WU Bo, WANG Jing

(State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China)
(EBUPT Information Technology Co. Ltd., Beijing 100191, China)

**Abstract:** The most intuitive description of RBAC is that permissions are granted to roles, and roles are granted to users. The permissions are the only credits to access resources, but users can't directly associate with permissions. With core RBAC and some mainstream selections in JavaEE, design and implement the privilege management framework which consists of user login modules and a variety of management operations modules that can manage users, roles and privileges.

**Keywords:** RBAC; JavaEE; privilege management

在基于 Web 的管理信息系统中，由于网络资源的开放与共享特性，系统安全一直是需要认真对待的问题，其中基于用户行为[1]对用户访问进行控制是保证系统安全的主要措施之一。当前主要有三种访问控制策略：自主访问控制、强制访问控制和基于角色的访问控制(RBAC，Role Based Access Control)[2]。

自主访问控制(DAC，Discretionary Access Control)是根据访问者和它所属组的身份来控制对客体目标的授权访问。它的优点是具有相当的灵活性，但是访问许可的转移很容易产生安全漏洞，所以这种访问控制策略的安全级别较低。

强制访问控制(MAC，Mandatory Access Control)是基于主体和客体的安全标记来实现的一种访问控制策略。它的优点是管理集中，根据事先定义好的安全级别实现严格的权限管理，因此适宜于对安全性要求较高的应用环境，但这种强制访问控制太严格， 实现工作量太大，管理不方便，不适用于主体或者客体经常更新的应用环境。

相比较而言，前两种访问控制模型都存在的不足是将主体和客体直接绑定在一起，授权时需要为每对主体和客体指定访问许可，这样存在的问题是当主体和客体达到较高的数量级之后，授权工作将非常困难。20 世纪 90 年代以来，随着对在线的多用户、多系统研究的不断深入，角色的概念逐渐形成，并产生了以角色为中心的访问控制模型(RBAC，Role Based Access Control),被广泛应用在各种计算机系统中。由

① 基金项目:国家杰出青年科学基金(60525110);国家 973 计划(2007CB307100,2007CB307103);国家自然科学基金(60902051);中央高校基本科研业务费专项资金(BUPT2009RC0505);电子信息产业发展基金
收稿时间:2010-08-16;收到修改稿时间:2010-09-19

于角色和权限之间的变化比角色和用户关系之间的变化相对要慢得多，减小了授权管理的复杂性，降低管理销，而且灵活地支持企业的安全策略，并对企业的变化有很大的伸缩性[3]。

## 1 RBAC模型

### 1.1 简介

RBAC 的基本思想就是：管理员授予给用户的访问权限，通常由用户在一个组织中担当的角色来确定，RBAC 中权限被授予给角色，角色被授权给用户，用户不直接与权限关联。RBAC 对访问权限的授予由管理员统一管理，而系统将根据用户在组织内所拥有的角色来做出访问控制，授权规定是强加给用户的，用户不能自主地将访问权限传递给他人，这是一种非自主型集中式访问控制方式。举例来说，在医院里，医生这个角色可以开处方，但他无权将开处方的权力传给护士。

由于 RBAC 实现了用户与访问权限的逻辑分离，因此它极大的方便了权限管理。因为如果一个用户的权限需要发生变化，只要将用户当前的角色去掉，加入代表新权限的角色即可，角色相对权限之间的变化比角色相对用户关系之间的变化相对要慢得多，并且委派用户到角色不需要很多技术，可以由行政管理人员来执行，而配置权限到角色的工作比较复杂，需要一定的技术，可以由专门的技术人员来承担，但是不允许他们委派用户的权限，这与现实中情况正好一致。

为了将用户和权限解耦，RBAC 模型引入了“角色”的概念，整个 RBAC 参考模型都就是围绕角色来建立的。根据对不同复杂度权限的需求，RBAC 参考模型定义了三部分组件——基本 RBAC 模型（Core RBAC）、角色分级 RBAC 模型（Hierarchy RBAC）和角色限制 RBAC 模型（Constrained RBAC），本次设计中采用了三种 RBAC 组件模型中的基本 RBAC 模型。

### 1.2 基本 RBAC 模型

基本 RBAC 模型定义了 RBAC 模型最基本的五个元素：用户、角色、目标、操作以及许可权，还包括一些其他元素：用户角色分配（User Assignment）、角色许可分配（Permission Assignment）以及会话，如图 1 所示。

用户代表人，但是也可以是一台机器或其他任何智能型物品，只要是能够发出动作的主体即可。

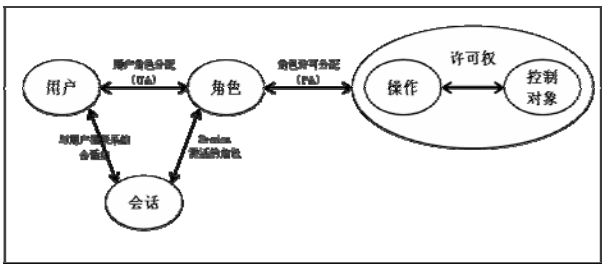


图 1 基本 RBAC 模型

角色表示一个工作职责，在一个组织机构环境中的工作职责。该职责可以关联一些关于权力和责任的语义。

目标表示资源或对象，任何访问控制机制都是为了保护系统的资源。目标可能包括文件，目录，数据库表等，甚至于磁盘空间，打印机等都是目标所指的范畴。

操作是程序的可执行的动作，被用户调用和执行。操作的类型取决于实现系统的类型。例如文件系统可能的操作时读写，执行等，数据库系统则是增加、删除、修改或查询操作。

许可权是一个许可，对在一个或多个目标上执行操作的许可。RBAC 标准中定义的权限均指正权限，但是并没有禁止负权限[4]。

用户角色分配：用户到角色的一种映射，表示一个用户所具有的角色，在本次设计中将是单向多对多的关系。

角色许可分配：角色到权限的一种映射，表示一个角色所具有的权限集合，在本次设计中将是单向多对多的关系。

会话集：是指用户激活角色时建立的动态的会话集合[5]。

RBAC 的关注点在于角色和用户以及角色和权限的关系，称为用户角色分配和角色权限分配。正如前面提到的，关系的左右两侧都是单向的多对多关系。在常见的关系型数据库中，只要两个表存在多对多关系，就必须要使用一个额外的关系表来记录两者之间的映射，这样无形中给数据的一致性维护带来了麻烦。

会话在基本 RBAC 模型中是比较隐晦的一个元

素。本次设计中并没有涉及到，所以此处不加以详细介绍。

## 2 权限管理框架的设计

### 2.1 设计思路

基于 RBAC 的权限控制框架需要满足实用性、可靠性和安全性等系统性能，需要完成包括用户登录控制，用户鉴权，用户详细信息、角色详细信息、权限详细信息、用户所拥有角色、角色所分配的权限的增加、删除、修改和查询等具体功能。

由于项目实际需求，从安全性、可靠性和实用性方面综合考虑，本次设计中采用了基本 RBAC 模型，该模型中共包含了五大基本元素，分别设计使用 UserBean 实体类实现模型中的用户对象，RoleBean 实体类实现模型中的角色对象，AuthorityBean 实体类实现模型中的角色许可分配过程，Action 类实现操作过程以及使用 Dao（Data Access Object，数据访问对象）类实现目标对象。

由于权限控制框架相当于项目的门户，一旦非授权用户获得了高级别的权限，将对项目造成巨大的危害。所以在选用基本 RBAC 模型的基础上，项目总体设计中将安全性放在首要位置。在整个框架开发过程中采用了 Struts 2 + Spring + Hibernate 三种 JavaEE 开发过程中最常用的框架选型，其中 Struts 2 实现从视图层到业务逻辑层的跳转控制、表项参数传递等，Spring 负责解决对象之间的依赖问题，简化对象的获取，Hibernate 简化关系数据库的操作，使得数据库一致性得到了很好的保证。鉴于本次设计的框架并没有特别关注于前台的设计，所以对于登录、控制操作等各个界面仅仅使用 JSP（Java Server Pages）语言实现了基本功能，没有增加具体的样式控制。

### 2.2 整体结构设计

按照 B/S 结构，本次权限控制框架设计分为三个层次：视图层，业务逻辑层，数据接口层。

视图层即用户所能接触到的操作界面。用户所做的各种输入以及后台返回的所有信息都通过视图层呈现给用户，由于本次设计并没有特别关注与前台表现，所以对于 JSP 页面的美化、格式控制等并没有投入太多精力，而是将重点放在了业务逻辑层。

业务逻辑层用来处理前台传递的各种请求以及参数，从而进行相应的方法调用、数据处理以及返回结果等功能。所有的控制操作都是由业务逻辑层完成的，使得用户显示界面与数据库交互分离，这样的设计符合 MVC（Model-View-Controller，模型-视图-控制）模式的思想和模式。

数据接口层用来创建、维护基于 RBAC 权限管理框架所要使用到的所有数据库表。在本次设计中按照纵向分割的思想，将所有的数据交互处理都设计为 Dao 类，实现了各个模块之间的解耦，更加符合软件开发的原则。本次设计选用的是 Derby 数据库，而在数据库交互时选用 Hibernate 来实现关系数据库的所有操作，利用它强大的关系数据库维护功能来避免在代码中显式的维护数据库之间一致性等关系。

总体架构按功能划分为全局配置文件及工具类、登录处理及页面设计、用户操作设计、角色操作设计、权限操作设计五大部分，每一部分中包含 MVC 思想层次划分的各种类。

1）全局配置及详细设计

Java Web 项目最重要的 web.xml 文件，配置 Struts 2 实现的 Struts.xml 文件，Spring 实现的 applicationContext.xml 文件，Hibernate 总体配置的 hibernate.cfg.xml 文件以及调试信息需要的 log4j.properties 文件，工具类包含了 Spring 和 Hibernate 所要用到的类。

2）登录处理及页面设计

主要包括登录页面，使用 JSP 语言实现基本功能，后台由 Struts 2 负责完成将用户名密码传递等信息传递给业务逻辑层，由 LoginAction 类实现用户登录以及鉴权等功能。

3）用户操作设计

主要包括用户控制界面实现增加用户、删除用户、修改用户、查询用户，使用 JSP 语言实现基本功能，用户实体 UserBean，用户实体对应数据库表项的映射配置文件 UserBean.hbm.xml，用来控制调用何种数据接口类中的方法的 UserAction 类，以及和数据库进行交互的 UserDao 类。

4）角色操作设计

主要包括角色控制界面实现增加角色、删除角色、修改角色、查询角色，使用 JSP 语言实现基本功能，

角色实体 RoleBean，角色实体对应数据库表项的映射配置文件 RoleBean.hbm.xml，用来控制调用何种数据接口类中的方法的 RoleAction 类，以及和数据库进行交互的 RoleDao 类。

5）权限操作设计

主要包括权限实体 AuthorityBean，权限实体对应数据库表项的映射配置文件 AuthorityBean.hbm.xml，以及和数据库进行交互的 AuthorityDao 类。

## 3 权限管理框架的实现

### 3.1 登陆实现

登录实现过程中类之间的调用关系、参数传递过程示意图如图 2 所示：

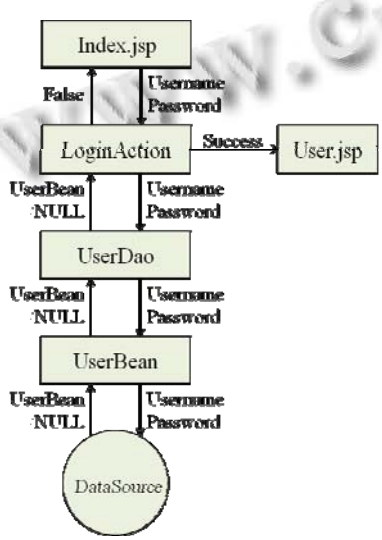


图 2 登录过程示意图

登录页面使用 JSP 实现，采用 HTTP POST 方法向传递 LoginAction 类传递用户名和密码参数。LoginAction 类继承了 ActionSupport 类，并覆盖了父类中的 execute 方法。方法的具体实现为：新建一个用户实体 UserBean，将传递来的参数用户名密码赋值给新建的实体，调用用户数据处理类 UserDao 的 setUser(UserBean)和 getUser()方法返回用户详细信息。如果该用户合法，则返回详细信息后的 User 实体被置入 HttpSession，同一会话中可以直接访问；如果不存在该用户名密码对应的用户则返回为空。

### 3.2 用户操作实现

针对用户操作的所有功能实现，需要进行的类之间的调用关系以及参数、Bean 的传递示意图如图 3 所示：

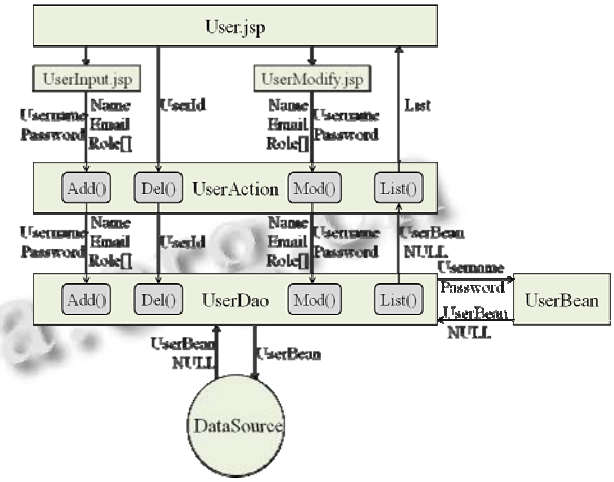


图 3 用户操作过程示意图

UserBean 实体类中定义了整个项目中要使用到的 User 实体的所有私有属性，以及对应的访问和赋值方法，所有针对用户操作的对象都是 UserBean。实体类是整个项目框架中包括传递、判断、调用以及返回等各种操作时的对象，是完整的信息封装体和载体，所以在项目中的利用率是最高的。在实体类的实现中，由于经常涉及到类型转换等会抛出异常的错误，所以在实现的时候在实体类的最外部加上了@SuppressWarnings(“unchecked”)来完成编译时的控制，将所有类型转换的异常全部忽略，这样做有助于提高运行时的效率。

UserAction 类是和前台即视图层交互的接口，负责将视图层传递来的数据加以处理并调用对应的 Dao 类接口实现操作。该类继承了 ActionSupport 类，并覆盖了父类中的 execute 方法。

UserDao 类是所有用户操作与数据库交互的接口，对于数据库的增加、删除、修改和查询操作都是在该类中实现的，本来繁琐的数据库交互 sql 语句因为使用了 Hibernate 而变得简单易懂，代码量也减少很多。

### 3.3 角色操作实现

针对角色操作的所有功能实现，需要进行的类

之间的调用关系以及参数、Bean 的传递示意图如图 4 所示。

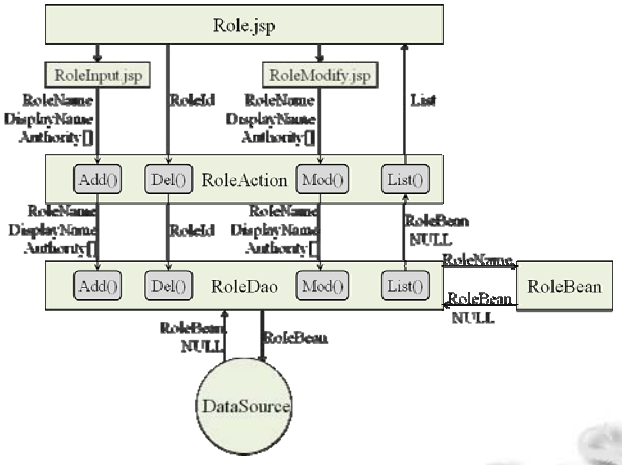


图 4　角色操作过程示意图

RoleBean 实体类中定义了整个项目中要使用到的 Role 实体的所有私有属性，以及对应的访问和赋值方法，所有针对角色操作的对象都是 RoleBean。和用户实体类 UserBean 类似，角色实体类 RoleBean 整个项目框架中包括传递、判断、调用以及返回等各种角色操作时的对象，是完整的角色信息封装和载体。RoleAction 类和 RoleDao 类也与 UserAction 类和 UserDao 类的具体实现一致。

### 3.4 权限操作实现

由于权限操作并不需要用户直接进行交互，所以权限操作没有设计视图层界面，只有对应的实体类、Dao 类等。AuthorityBean 实体类中定义了整个权限实体对应的私有属性，以及对应的访问和赋值方法，所有针对权限操作的对象都是 AuthorityBean。AuthorityDao 类是所有权限操作与数据库交互的接口，无论是对于数据库的增加、删除、修改或是查询都是在该类中实现。

## 4　存在的问题

权限管理框架至此已经实现了所有功能，但是在实现的过程中仍然存在一些问题和可以改进的地方。

关于模型的选用：由于稳定性和安全性的要求，本次设计的权限管理框架采用了最为稳定成熟的基本 RBAC 模型，仅实现了基本功能。

关于数据库访问效率的提高：数据库存储方面，由于本次权限管理框架中涉及到的数据库表仅包含用户、角色和权限三个表以及两个关联关系表，存储所需空间以及数据数量级较小，所以在易用性和查询效率间权衡，使用 Hibernate 读取用户或者角色数据的时候关闭了延迟加载。如果扩展设计框架，数据存储量提升的情况下，则应当考虑延迟加载的应用来避免数据库操作的延迟过大，同时可以考虑为适当的数据库表增加索引来提高数据查询效率。

关于异常的抛出：程序执行中的异常应该尽量早的抛出，尽量晚的捕获。为了准确定位异常抛出的位置，抛出异常的原因以及异常的种类，应当按照这样的原则进行异常处理。本次框架设计中的代码量较小，系统异常位置较为明显，由于异常的抛出和捕获以及调试观察会极大的影响系统效率，所以本次设计实现采取编译时控制的方式，将所有类型转换异常抛出关闭。在异常抛出和捕获的过程中，在捕获方法内添加提示性语句输出到控制台，可以在项目运行时随时监控和观察项目状态，避免切换调试状态，为项目监控提供便利。

系统灵活性和扩展性的要求需要同时满足安全性、可靠性和实用性，并在三者之间取得平衡，本次设计中重点关注的是安全性和可靠性，如果关注的重点变更，则应当对系统中部分代码加以修改以满足要求。

## 参考文献

1 Ni P, Liao JX, Wang C, Ren KY. Web information recommendation based on user behaviors. 2009 WRI World Congress on Computer Science and Information Engineering. 2009.3.31–2009.4.2: 426−430.

2 Ferraiolo DF, Kuhn DR. Role Based Access Control. 15th National Computer Security Conference. Oct 13−16, 1992: 554 − 563.

3 Sandhu RS, Coyne EJ, Feinstein HL, et a1. Role-based access control modules. IEEE Computer, 1996,2:38−47.

4 Ice Cloud, An Introduction to Role-Based Access Control, 2004.

5 丁振国,吴环宇.RBAC 在治理信息系统中的应用.微计算机信息(管控一体化),2007,23(3−6):4−6.