基于 RBAC 模型的权限组件①

王志瑞, 顾 问, 刘正涛

(三江学院 计算机科学与工程学院, 南京 210012)

摘 要: 基于信息化系统的特点及其对权限控制的需要, 在.NET 框架下, 基于 RBAC 模型设计与实现了一套权限控制组件, 并在设计过程中对 RBAC 模型进行了扩展, 引入了权限委托、分级授权, UBAC 等特性, 并将该组件应用于多个 EIS 系统的开发过程中, 结果表明该组件能够很好的集成于信息化系统中, 能够减轻权限开发的复杂度, 提高权限管理工作的效率.

关键词: RBAC; 权限委托; UBAC; NET; 权限组件

Access Component Based on RBAC

WANG Zhi-Rui, GU Wen, LIU Zheng-Tao

(Department of Computer and Information Engineering, Sanjiang University, Nanjing 210012, China)

Abstract: In this paper, based on the characteristics of information system and the need for access control, a set of permissions control component is designed and implementated using RBAC model on the platform of .Net framework. Furthermore, with the introduction of the authority commission, hierarchical authorization and UBAC features, the RBAC model is expanded in the design process, and is applied to the component of multiple EIS system development process. Finally, the results show that the component can be integrated quite well in information system. It reduces the complexity of the development, and improves the efficiency of permissions management work.

Key words: RBAC; inheritance of role; UBAC; .NET; access component

在信息系统中, 权限控制组件实现对登陆用户的身份认证, 并在用户进入系统后, 对用户所能够进行的操作和能够查看的数据进行控制, 是系统信息安全的重要保证.

传统的权限控制, 如强类型访问控制、自主类型访问控制普遍存在着开发工作量大, 用户功能单一, 授权不灵活、不方便等问题, 不能给系统数据提供充分的数据安全保护, 难以满足复杂的系统环境要求[1,2]. 基于RBAC的权限模型, 利用角色来控制用户与操作的逻辑分隔, 简化了权限管理, 特别适合大型管理信息系统的特点, 因此被誉为最有前景的访问控制策略[3,4].

1 RBAC模型

RBAC 权限模型共提出了 4 个版本, 分别是基本模型 RBAC0 (Core RBAC), 角色继承模型 RBAC1(Hierarchal RBAC)、 角色限制模型 RBAC2(Constraint RBAC) 和统一模型 RBAC3(Combines RBAC) [5].

RBAC0 定了 RABC 控制系统最小的集合, 主要包括四个基本元素: 用户集合、角色集合、许可权集合和会话集合. 许可权与角色之间是多对多的指派关系, 角色与用户之间是多对多的指派关系, 用户通过角色关系拥有了许可权集合, 实现了用户与许可权之间的分割, 方便了授权的管理. 会话由用户控制, 用户可以创建会话并激活多个角色, 从而获得相应的权限, 用户可以在会话中更改激活的角色, 并可以主动终止一个会话, RBAC0 模型如图 1.

RBAC1、RBAC2、RBAC3 都是在 RBAC0 的基础

① 收稿时间:2014-06-17;收到修改稿时间:2014-08-04

上进行的扩展. RBAC1 中加入了角色继承的思想, 继承关系遵循偏序关系, 并且加入了继承层次的限制. RBAC2 是在 RBAC0 的基础上增加限制形成的, 这些限制是用于确定 RBAC0 中各个组件的值是否是可接受的, 只有那些可接受的值才是允许的. RBAC2 中引入的限制可以施加到 RBAC0 模型中的所有关系和组件上. 比如限制两个角色之间互斥关系, 不能同时指派给某一个用户; 或者限制某一个角色的最大成员数等. RBAC3 是把 RBAC1 和 RBAC2 组合在一起, 提供角色的继承和限制的能力, RBAC3 模型如图 2.

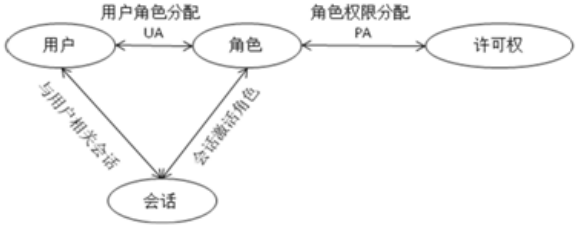

图 1 RBAC0 模型

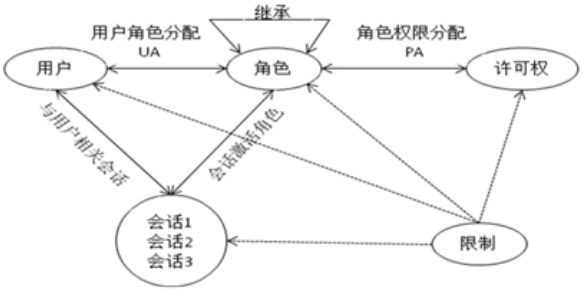

图 2 RBAC3 模型

RBAC 模型只是一个理论模型, 它为权限组件的设计提供了有力的理论支持, 指导我们完成权限组件的设计工作. 但是在实际的使用过程中, 还是会有一些方面的问题需要我们去扩展. 在 OA 系统中, 某个人需要外出出差, 在指定时间段, 他想把自己所负责的部分工作委托给其他人员代办, 到期之后能够自动回收权限; 再者就是当一个企业的用户规模特别多的时候, 超级管理员会将对应的权限授权给对应的下级部门管理员, 下级部门管理员可以再将自己的权限授权给其所管理的具体用户, 诸如此类的问题在 RBAC 中没有给出具体的说明, 本文在权限组件的设计过程中对传统的 RBAC 模型进行了改进, 引入了权限委托, 分级授权, UBAC 等特性, 扩展了 RBAC 模型.

2 权限组件设计

权限控制组件在EIS系统中是必不可少的组成部分, 设计一个可复用的权限控制组件对于 EIS 系统的开发至关重要. RBAC 模型是被公认的权限模型, 因此按照 RBAC模型设计权限控制组件能够提高权限组件通用性.

2.1 数据模型设计

按照 RBAC 模型设计的权限组件, 用户权限是可以动态设定的, 而不是通过程序逻辑写死在代码中, 其关键点就是将权限设定信息都保存在数据库中, 可以通过授权维护程序修改数据库内的数据, 实现用户授权控制的改变. 因此在控制组件设计与开发过程中, 需要将 RBAC 模型进行数据模型化, 将 RBAC 模型中的概念与软件设计中的概念进行对应, 将 RBAC 模型转化为对应的数据模型.

在 RBAC 模型中提出了用户、角色、许可权、限制、会话几个概念, 本文在权限组件的设计过程中根据系统开发的需要定义了如下几个数据模型: 用户, 角色, 用户角色关系, 模块, 资源, 操作, 角色授权, 用户授权. RBAC 模型中的用户、角色对应到数据模型中的用户, 角色, 用户角色关系; RBAC 模型中的许可权对应的数据模型中的模块, 资源, 操作; RBAC 中的权限分配和限制对应到数据模型中的角色授权, 用户授权. 下面给出该组件中相对关键的三个表的结构, 见表 1 角色表(Role), 表 2 资源表(Resource), 表 3 授权表(Privilege).

表 1 角色表(Role)

字段名	数据类型	含义
RoleID	int	角色编号, 主键
RoleDes	varchar(50)	角色描述
UpID	int	父角色, 角色继承

表 2 资源表(Resource)

字段名	数据类型	含义
ResCode	varchar(50)	资源编码, 主键
ResDes	varchar(50)	资源描述
ResURL	varchar(100)	资源路径
ResHasOps	varchar(100)	资源拥有的操作
ModID	int	所属模块

表 3 角色/用户授权表(Role/User Privilege)

字段名	数据类型	含义
RoleID/UserID	int	角色/用户编号
ResCode	varchar(50)	资源编码
OpeCode	int	所授操作编码
DueDate	datetime	到期时间
AuthUserID	int	授权用户 ID

AuthRoleID	int	授权角色 ID
isGift	bit	是否允许授予别人

其中相关字段的作用如下:

Role 表内的 UpID 字段: 用来保存此节点的父节点 ID, 形成角色之间的继承关系, 上层角色具有底层角色所拥有的权限; Resource 表内的 ResHasOPs 字段: 用来保存资源所拥有的多个操作编码, 多个操作编码之间用分隔符分割; 角色/用户授权表内的 DueDate 字段: 用来设定权限委托情况下, 权限的到期期限, 到期后将自动回收; 角色/用户授权表内的 AuthUserID、AuthRoleID、isGift 字段: 在分级授权中, 当授权人的权限发生减少时, 用来控制所减少的权限也从其所授出的下级用户或角色那里删除掉, 防止出现上级管理员所不具有的权限, 下级管理员却有的现象, 另一个功能可以查看某个角色所有权限的授权细节信息.

2.2 框架结构设计

RBAC 权限组件只包含一套数据表无法实现权限的控制, 还需要配套的业务逻辑处理类和数据维护程序共同构成权限控制组件, 具体的权限控制组件结构图如图 3 所示:

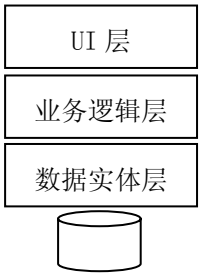

图 3 权限组件结构

数据实体层用来实现将数据库中的表转化为面向对象的类, 实现 O/R 转换, 方便与业务逻辑层的结合, 本文采用的是.Net 环境下的 EntityFramework 框架实现的数据实体层.

业务逻辑层主要与数据实体层和 UI 层进行交互, 实现基本的权限表数据维护和重要的逻辑处理, 重要的业务逻辑主要包括:

1) 权限获取: 根据输入的用户名信息, 通过角色继承, 角色授权, 用户授权(权限委托)等多方面因素进行权限合并, 计算出此用户所拥有的所有权限.

2) 登录验证: 根据用户输入的用户名和密码进行验证, 如果成功了, 则调用“权限获取”获取到登录用户的所有权限, 并将其保存到状态对象中.

3) 登录检测: 用来判断某个用户是否登录, 如果没有登录, 则跳到登录页面.

4) 权限授权: 当授权的时候, 调用“权限获取”获取到授权用户所拥有的权限(自己有的权限才允许授权), 对选中的角色进行权限授权, 根据授权项的变化情况进行处理, 并更新角色授权表内的权限数据.

5) 权限判断: 当对某具体资源的某个具体操作进行权限判断时, 利用状态对象中保存的当前登录用户的权限信息进行判断, 无需每次都访问数据库进行判断, 减少数据库的访问操作.

6) 权限委托: 权限委托就是权限授权的一种特例, 不进行角色授权, 直接进行用户授权(UBAC), 将自己需要委托的权限授权给被委托人, 并设定所授权限的结束时间, 当时间到期后, 被授权人就无法再享有此权限的使用权利.

UI 层主要用来提供便利的数据维护和数据展示界面, 方便管理员准确的进行权限的分配和控制.

3 权限组件实现

权限组件的实现包括很多的内容, 包括数据实体层的生成, 业务逻辑层的编写, UI 层的构建等内容, 本文只介绍其中相对复杂的业务逻辑的实现思路.

3.1 权限获取

RBAC 权限模型的所有权限信息都保存在数据库中, 因此要获取某个用户的权限信息, 需要利用数据库内的数据进行计算. 由于权限授权过程中包括角色继承, 角色授权, 权限委托等多方面的权限因素, 因此进行权限计算时需要将多面的权限进行合并计算得到用户的最终权限, 具体方法如下:

1) 在数据库中编写一个获取某个角色下所有子角色的函数 GetSubRole.

2) 利用角色授权表和 GetSubRole 函数获取此用户所属角色所拥有的权限信息(资源和操作).

3) 利用用户授权表(权限委托)获取此用户所拥有的权限信息(资源和操作), 获取权限时需要考虑权限的到期时间, 之后有效期内的权限才是合法的权限.

4) 将角色角度的权限和用户角度(权限委托)的权限进行合并操作, 得到用户的最终权限.

3.2 登录控制

基于 B/S 的 EIS 系统[6], 由于其工作特点, 所有的

网页用户都可以直接通过浏览器输入打开, 因此对于一些需要登录认证的页面, 需要在用户访问的时候首先做登录检测, 如果检测不通过, 则直接跳转到登录页面让用户登录.

由于网页的无状态特性, 在 B/S 系统中登录控制主要依靠在内置对象 Cookie 或 Session 中保存登录信息, 在需要登录认证的页面中通过检测内置对象中是否保存有登录信息来实现. 但是 EIS 系统中页面很多, 不可能在每个页面中都直接加入这些检测代码, 如果代码需要更改, 则需要为每个页面修改, 本文根据.NET 的特点, 设计了一种基于母版页的设计方法和一种基于页面基类的设计方法. 在基于母版页的设计方法中, 首先创建一个母版页, 在母版页的 Pre_Load 事件中编写登录检测代码, 所有需要登录认证的页面套用此母版页即可; 在基于基类的检测方法中, 首先从 Page 类派生出一个子类 BasePage, 在子类 BasePage 中重写 OnPreLoad 方法, 在此方法中完成登录检测, 所有需要登录认证的页面继承 BasePage 页面即可.

在登录控制逻辑中, 如果用户登录成功了, 则需要调用权限获取逻辑获取此用户的权限信息, 并将权限信息保存到内置对象中, 便于权限判断时使用, 免去了权限判断时不停的访问数据库.

3.3 权限判断

权限判断的主要功能就是查看当前用户对某个资源的某个操作是否有权限. 权限分为资源的访问权限和资源内的操作权限, 首先拥有了资源的访问权限才可以访问此资源, 进而才可以进行资源内的相关操作.

由于权限判断会被多次的使用, 因此可以将权限判断封装成一个业务类, 在类中提供一个权限判断的方法 bool checkPrivilege(string ResCode, string OpCode) 来实现权限的判断逻辑, 此方法通过读取内置对象中保存的权限信息来判断某用户对某资源的某操作是否有权限.

在需要权限认证的页面中的 Page_Load 事件中, 首先调用此方法判断是否有访问权限, 如果没有则跳转到用户的默认首页, 在有访问权限的前提下再通过此方法判断是否有此资源内某操作的权限, 有则显示, 无则隐藏.

3.4 角色授权

角色授权(RBAC)是权限组件中经常要进行操作的功能, 通过对某个角色授权来实现对此角色内的所有用户授权; 由于授权时只能将自己拥有的权限授权给角色, 因此也是一种分级授权模式, 授权的具体业务逻辑如下:

1) 在 UI 中展示出被授权的角色对象列表, 单击某角色对象后进入角色授权操作.

2) 在角色授权操作中, 首先获取授权者当前可以授出的权限信息和被授权者已经拥有的权限信息, 并在 UI 中展示出授权者可以授出的权限列表, 并根据被授权用户所拥有的权限让权限列表中对应的权限自动选中, 标示出哪些权限是被授权者已有的权限.

3) 勾选结束后, 输入权限到期时间, 保存时, 逐条处理每一项资源权限, 如果是给被授权人新增某项权限, 则直接新增到数据库中即可; 如果是给被授权人删除某项权限, 则需要遵循分级授权的原则, 首先判断此被授权者是否将此权限授权给别的角色或者用户, 如果有此现象, 则利用递归的方式将相关的授权关系从数据库中删除.

3.5 权限委托

权限委托就是将自己的权限在指定的时间范围内授权给别的用户, 是一种受限的用户授权(UBAC), 为了方便控制, 被委托的权限不允许再授权给别人, 因此业务逻辑比角色授权简单, 具体的业务逻辑如下:

1) 在 UI 中展示出被授权的用户对象列表, 单击某用户对象后进入权限委托操作.

2) 在权限委托操作中, 首先获取授权者当前可以授出的权限信息, 并在 UI 中展示出来, 查看被授权人是否有已委托的权限, 如果有, 则让对应的权限自动选中.

3) 勾选结束后, 输入权限到期时间, 保存时, 逐条处理每一项资源权限, 如果新增一项权限, 则新增到用户授权表中, 如果删除了一项权限, 则从用户授权表内直接删除即可(无需递归删除).

4 组件应用

该权限组件自完成之后, 已在多个基于 B/S 的 EIS 项目中得到了应用, 具体方法如下:

1) 将权限组件相关的业务逻辑代码和 Web 应用程序添加到 Web 项目中.

2) 将登陆检测逻辑 checkUserLogin()置入到所有需要登录后才可以访问的程序中.

3) 在需要授权程序的 Page_Load()事件中, 调用

权限判断业务方法 bool checkPrivilege(string ResCode, string OpCode), 判断当前登录用户是否有此程序的访问权限(如果没有, 则自动跳转到用户的默认首页), 并在 Page_Load()事件, 通过调用 checkPrivilege()方法判断当前登录用户是否拥有当前程序中相关按钮的操作权限, 通过返回结果来设定按钮的显示与隐藏(如果没有此按钮的权限, 则按钮会被隐藏).

通过应用发现, 开发一套通用的权限组件后, 能够在以后的项目中很方便的重复使用, 提高了项目开发的效率.

5 结语

本文以RBAC权限模型为理论指导, 按照.NET的技术规范, 设计和实现了一套基于 B/S 的权限管理组件, 该组件能够自成系统, 能够很方便的集成到 EIS 系统中为 EIS 系统提供权限控制, 并能够很方便的在此权限组件的基础上增加业务层面的业务策略, 减轻了 EIS 在权限控制方面的开发量, 提高了权限管理工作的效率.

参考文献

1 杨剑,闪四清.ASP.NET 环境下基于角色的权限控制的实现.计算机技术与发展,2007,17(5):234–237.

2 刘金晓,马素霞,齐林海.Web 应用系统中权限控制的研究与实现.计算机工程与设计,2008,29(10):2550–2553.

3 吴薇.基于角色的访问控制技术的用户权限管理及实现.福建电脑,2008,24(11):176–177.

4 严晓光,等.软件质量保障平台中基于 RBAC 的统一身份认证应用研究.计算机工程与科学,2009,31(3):97–100.

5 吴光成,时云峰.基于RBAC的权限管理系统的实现.电子测试,2008,(5):87–91.

6 宋云,李峰.基于 RBAC 的 B/S 系统访问控制设计与实现.软件导刊,2009,(7):28–30.

7 信科等.基于 RBAC 权限管理系统的优化设计与实现.计算机技术与发展,2011,21(7):172–174.