ASP. NET MVC 下基于 RBAC 权限认证的设计与实现

梁 智[a],沈 倩[b]

(重庆理工大学 a. 信息与教育技术中心; b. 材料科学与工程学院,重庆 400054)

摘 要:介绍了基于角色的访问控制模型,阐述了用户、角色、权限之间的关系。讨论了MVC模型及其三大组成部分的作用与职能,介绍了以MVC模型为基础的微软ASP. NET MVC框架。针对ASP. NET MVC下基于RBAC权限认证提出了一种实用可行的解决方案。该方案引入了角色的概念,实现了用户与权限的逻辑分离,以便适应大多数应用系统对用户访问控制的需求。

关 键 词:访问控制;RBAC;MVC模式;ASP. NET MVC

中图分类号:TP391 **文献标识码**:A **文章编号**:1674-8425(2011)09-0075-06

Design and Implementation of Access Control Based on RBAC in ASP. NET MVC

LIANG Zhi[a], SHEN Qian[b]

(a. Center of Information and Education Technique, b. School of material Science and Engineering, Chongqing University of Technology, Chongqing 400054, China)

Abstract: This paper first described the role-based access control pattern and then expounds the relationship among user, role and permissions. This paper secondly discussed MVC pattern and its three components, then introduced MVC pattern as the foundation of Microsoft ASP. NET MVC framework. Finally it put forward a practical and feasible solution of access control based on RBAC pattern in ASP. NET MVC

Key words: Access control; RBAC; MVC; ASP. NET MVC

收稿日期:2011-05-15

作者简介:梁智(1982—),男,重庆人,主要从事信息系统开发与设计研究。

随着计算机网络信息技术的发展和企业信息化建设的不断推进,越来越多的企业和单位通过网络平台构建起自己的网络信息系统。MVC 模式作为一种灵活的软件设计模式已被广泛地运用到了网络信息系统的开发中。与此同时,由于网络是一种开放式的架构,与网络连接的资源并没有充分得到保护,用户可以随时随地地通过网络获得自己需要的各种各样的资源信息,有些甚至是关系到企业或单位自身利益与发展的重要信息,因此防止信息遭受非法用户的获取已逐渐成为一个关键性问题,所以,如何为网络信息系统规划制定一套良好的访问控制机制显得尤为重要。

1 基于角色的访问控制

访问控制技术起源于 20 世纪 70 年代,是一种针对越权使用资源的防御措施。该技术的基本目标:为了限制访问主体(用户、进程、服务等)对访问客体(文件、系统等)的访问权限,从而使计算机系统在合法范围内使用;决定用户能做什么,也决定代表一定用户利益的程序能做什么[1]。常见的访问控制技术有自主访问控制 DAC(discretionary access control)、强制访问控制 MAC(mandatory access control)、基于任务的授权控制 TBAC(task-based access control)和基于角色的访问控制 RBAC(role-based access control)。

基于角色的访问控制早在 20 世纪 70 年代就已经提出,直到 20 世纪 90 年代才得以发展。公认并被广泛采用的是由 George Mansion 大学信息安全技术实验室(LIST)的 Ravi Sandhu 教授等于 1996 年提出的 RBAC96 模型。

基于角色的访问控制在用户和访问许可权之间引入角色(Role)的概念,其目的是为了隔离用户与用户所具有的权限。分配给每个用户一个合适的角色,每一个角色都具有其对应的权限。一个用户可以有多个角色,一个角色也可以有多个用户。一个角色可以有多个权限,相同的权限也可以赋予多个角色[2]。角色作为中间桥梁把用户和权限联系起来,一个角色与权限的关联可以看作是该角色拥有的一组权限的集合,与用户的关联可以看作是若干具有相同身份的用户集合。权限赋予角色,角色分配给用户。RBAC 访问控制基本模型如图 1 所示。

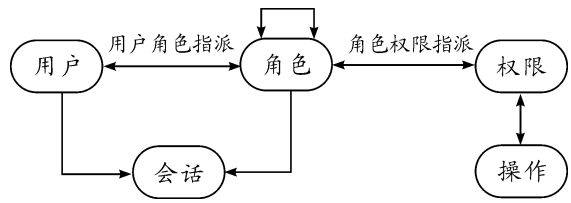

图 1 RBAC 访问控制基本模型

2 MVC 模式

早期的软件开发主要依靠事件及动作的驱动来设计,使得编制出来的软件在数据处理、程序功能与显示代码等各部分逻辑不很清晰,系统的耦合度也过高,不利于开发和维护[3]。MVC 模式的提出较好地解决了这一问题。MVC 模式是“model-view-controller”的缩写,中文翻译为“模式 - 视图 - 控制器”,是 20 世纪 70 年代末由 Trygve Reenskaug 提出,运用于 Smalltalk 平台上[4-5]。

MVC 模式将系统划分为 3 个部分:模型(model)、视图(view)和控制器(controller)。模型与领域模型的概念不同,包含完成业务所需的数据、行为和逻辑规则,是处理逻辑问题的一种抽象和用于存储数据的组件,也是整个模型的核心部分。视图是根据模型数据进行内容展示的组件和模型的外在表示以及应用系统与外界的接口。控制器用于接收用户输入的指令,并调用操作相应模型,选择一种视图并输出内容,它是模型与视图

之间的联系纽带。如同一般的程序结构一样,MVC 有输入、处理、输出 3 个部分:Controller 对应于输入;Model 对应于数据处理;View 对应于输出。其中控制器与平台和操作系统的关系最为密切,View 与之部分相关,Model 与平台无关[6]。

ASP. NET MVC 是微软官方提供的 MVC 模式编写 ASP. NET Web 应用程序的一个框架,是继 ASP. NET WebForm 后的又一种开发方式,使 ASP. NET开发人员拥有了另一个选择。ASP. NET MVC 框架由 Castle 的 MonoRail 而来,历经 9 个版本。2009 年发布了该框架的 1.0 正式版,目前仍在不断的完善与扩展其功能 ,最新版已更新到 ASP. NET MVC 2.0。相对 ASP. NET MVC 1.0 版的改善主要表现在区域化支持(areas)、强化验证功能、强类型 UI 辅助方法、UI 辅助方法模板化等方面。新特性促进开发人员将工作重点放在业务需求的分解和业务逻辑的设计上,简化和缩短枯燥而又乏味的编码实现过程。ASP. NET MVC 彻底改变了使用微软平台的 Web 开发者,全新框架设计方面更加重视纯净的架构、设计模式的运用以及可测试的能力,将所有代码的可控性还原给开发者,因此,使得 MVC 模式被迅速广泛地应用于大型可扩展的 Web 项目开发中。

3　RBAC 在 ASP. NET MVC 框架下的实现

在 ASP. NET WebForm 的架构下可以通过一定的配置即可实现用户身份验证与授权。特别是在 ASP. NET 2.0 的 Membership 的功能的支持下,可以建立更加简洁可复用的用户认证系统。通过配置文件 web. config 可以做到对页面或目录以及不同用户身份可见性的定制,但是它是基于物理文件和目录[7-8]。而在 ASP. NET MVC 架构下,用户访问的每个页面在磁盘中并没有一个固定的物理文件,它是通过 Controller 控制数据与视图的组合来生成 HTML 代码,进而向客户端输出,因此,在 MVC 模式下,权限的粒度可以做到 MVC 的 Action 上,这无疑是一个很好的选择,因为请求的功能入口是 Controller 相对应的 Action,无论是一个页面,一个按钮,还是一次查询,都是通过 Action 请求来实现的。这样只需要在每个控制器的 Action 请求执行之前进行权限判断就可以做到用户的访问控制。

3.1　数据库设计

RBAC 数据库实体关系如图 2 所示,其中主要包括如下几张表:用户表,包含用户 ID、用户名、登陆密码等,其存放所有的系统用户信息;角色表,包含角色 ID、角色名称等,其存放系统所有角色信息;权限表,包含权限 ID、控制器名称、动作名称,其存放系统中所有需要授权的行为权限;用户角色表,包含用户 ID、角色 ID,其用于存放用户与角色之间的关系;角色权限表,包含权限 ID、角色 ID,其用于存放角色与权限之间的关系。通过以上各表对应关系的建立实现用户与 ControllerName 与 ActionName 之间的映射,从而实现用户访问控制。

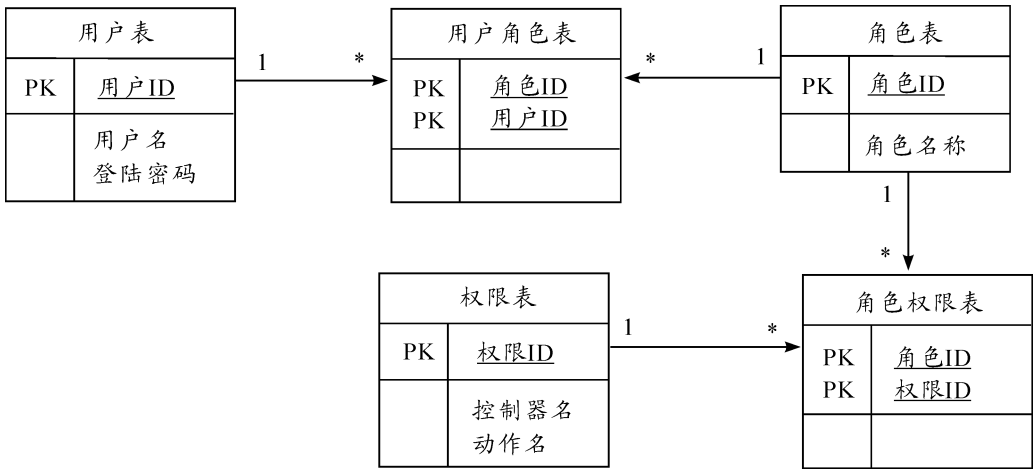

图2 基于RBAC的数据库实体关系

3.2 系统设计

要对系统用户的权限进行认证,首先必须知道当前用户是谁。如果连当前用户的身份都一无所知,那么权限认证也就无从谈起了。所以当用户登陆系统时,在数据库中完成用户名与登陆密码认证之后,必须对用户的身份进行必要的记录。通常可以把这样的用户身份信息记录在Session当中或者是Cookie当中。二者主要的区别在于Session是存放在服务器端而Cookie存放在客户端。Session更加安全,不过在使用过程中容易出现丢失的情况。Cookie的安全性不如Session,但可以通过加密的方式对其安全性进行提升。

这里以Cookie为例,当用户登陆成功后,从数据库中获取其用户名以及其角色集合,并向客户端写入认证Cookie。在这里记录其角色集合是为了在以后的权限认证过程中不必每次都从数据库中获取其角色信息集合,以提升其认证的效率,减轻服务器端的压力。服务器每当接收到客户端请求时,如果系统未能从客户端Cookie中获得相应用户认证信息,系统将拒绝用户的此次请求。其客户端认证Cookie写入代码如下:

```
FormsAuthenticationTicket authTicket = new FormsAuthenticationTicket(
1,
"张三",//登陆用户名
DateTime.Now,
DateTime.Now.AddMinutes(60),
false,
"admin;manager");//登陆用户角色集合
string encryptedTicket = FormsAuthentication.Encrypt(authTicket);
System.Web.HttpCookie authCookie = new System.Web.HttpCookie(
    FormsAuthentication.FormsCookieName, encryptedTicket);
    System.Web.HttpContext.Current.Response.Cookies.Add(authCookie);
```

本文提到MVC模式下权限的粒度可以做到Action上,因为用户的每一个操作都是通过对Action的请求来实现的。那么要判断一个用户是否具有执行指定控制器的Action的权限,实际上就是判断该用户在数据库中所属角色集合当中是否存在某个角色具有完成指定控制器的Action的权力。类似于这样的数据库查询实现起来并不困难,关键问题在于如何做到对每个需要授权认证的Action请求进行拦截。这里提供一种方式来实现这种权限认证拦截。可以通过定义Attribute来实现,在System.Web.Mvc程序集中提供一个Ac-

tionFilterAttribute 抽象类,它实现 IActionFilter 接口,这个接口提供一种可以对 Action 的 AOP 拦截的机制,通过使用 Action Filter,能够轻松实现认证、缓存、异常处理等业务功能。IActionFilter 接口定义如下:

```
public interface IActionFilter
{
    void OnActionExecuted(ActionExecutedContext filterContext);
    void OnActionExecuting(ActionExecutingContext filterContext);
    void OnResultExecuted(ResultExecutedContext filterContext);
    void OnResultExecuting(ResultExecutingContext filterContext);
}
```

其中:OnActionExecuting 方法表示在 Action 执行之前完成指定操作;OnActionExecuted 方法表示在 Action 执行之后完成指定操作; OnResultExecuting 方法表示在结果生成之前完成指定操作;OnResultExecuted 方法表示在结果生成之后完成指定操作。显然,只要实现 OnActionExecuting 方法,在其中执行对当前用户的身份认证即可达到对用户进行访问控制的目的。

此时需要新建一个类,它继承自 ActionFilterAttribute 抽象类。通过重写这个抽象类的 OnActionExecuting 的虚方法实现对权限的认证,其主要步骤为首先判断当前用户是否登陆了系统,获取当前登陆用户的角色集合和当前所执行的 Controller 名称与 Action 名称,判断用户所属角色集合当中是否有角色被允许执行当前的给定的 Controller 名称与 Action 名称。如果认证失败,则执行相应跳转并终止当前 Action 的执行。部分代码如下:

```
public class RequiresAuthenticationAttribute :ActionFilterAttribute
{
    public override void OnActionExecuting(ActionExecutingContext filterContext)
    {
        //获取 Controller 名称、Action 名称、用户角色集合。
        string controllerName = (string)filterContext.RouteData.Values["controller"];
        string actionName = (string)filterContext.RouteData.Values["action"];
        string[] roles = filterContext.HttpContext.Request.Cookies[FormsAuthentication.FormsCookieName].UserData.Split(new char[]{';'});
        //判断用户是否登陆,数据库中判断角色集合是否能够执行指定 Controller 与 Action。
        if (!filterContext.HttpContext.User.Identity.IsAuthenticated ||
            !DataBase.IsAllowed(roles, controllerName, actionName))
        {
            //用户权限认证失败,跳转到指定页面
            filterContext.HttpContext.Response.Redirect("登陆页面",true);
        }
    }
}
```

至此,主要的工作都已经完成了的。接下来只需要在需要进行权限控制的 Action 或 Controller 前加上[RequiresAuthentication],这些 Action 或 Controller 中的所有 Actions 就会自动被 RequiresAuthentication 这个 Filter 进行处理。如果某一个 Action 被标上了[RequiresAuthentication],而数据库中又不存在请求用户执行该 Action 所对应的权限,那么根据示例的代码,该用户将无法访问这个 Action,并执行相应的页面跳转。添加权限认证 Filter 的代码如下:

```
[RequiresAuthentication]
public class HomeController :Controller
    {
    [RequiresAuthentication]
    public ViewResult Index( )
    {
      …//页面逻辑代码。
    }
}
```

以上示例代码在每次对 Action 请求进行权限认证时,都会请求一次数据库以查询是否具有执行访问的权限。如果系统并发请求较多的话其性能势必受到影响,可以采取缓存的机制来缓解服务器压力。即当系统启动时,将所有系统中角色所对应的 Controller 名称和 Action 名称以全局静态变量的形式存储起来,当拦截器捕获到客户端请求并进行权限验证时,将数据库查询的方式改为在此缓存集合中进行相关查询。通过这样的设计修改之后,系统权限认证的效率将会得到极大的提升。当然,在这里还必须实现该缓存变量与数据库记录的同步更新以防止数据不一致的情况。

4 结束语

随着 MVC 模式在大型网络信息系统中的应用日渐广泛,易用、高效的用户权限访问控制对系统的安全性意义重大。本文所提出的解决方案引入了角色的概念,实现了用户与权限的逻辑分离,能够很好适应大多数应用系统对用户访问控制的需求。当然在具体的实践中,可以考虑对 RBAC 模型进行扩展,采用其扩展模型引入“组”的概念等,这些都有待进一步研究实践。

参考文献:

[1] 段云所. 信息安全概论[M]. 北京:高等教育出版社,2003.

[2] David F F,John F B,Richard K D. A Role-Based Access Control Model and Reference Implementation within a Corporate Interanet[J]. ACM Transactions on Information and System Security,1999,2(1):34 - 43.

[3] 刘克. MVC 架构及其在 Web 应用开发中的应用[J]. 计算机应用与软件,2006,23(7):57 - 59.

[4] 张升平. 基于 MVC 模式的研究生管理系统[J]. 重庆工商大学学报:自然科学版,2006,23(3):277 - 280.

[5] 杜勇前. 基于 MVC 模式的 Web OA 系统设计[J]. 重庆工学院学报:自然科学版,2007,21(11):78 - 81.

[6] 任中方,张华,闫明松,等. MVC 模式研究的综述[J]. 计算机应用研究,2004,24(10):1 - 4.

[7] 涂刚,李建,刘华清. . ASP. NET MVC 的研究[J]. 软件工程师,2010(8):54 - 57.

[8] 周益宏,陈建勋. ASP. NET MVC 中基于 AOP 和 RBAC 的权限控制实现[J]. 现代计算机:专业版,2010(7):206 - 208.

(责任编辑 刘 舸)