Hierarchical Group Permissions Theory/Resources?

Asked 16 years, 3 months ago Modified 6 years, 11 months ago

Viewed 4k times

2

Does anyone know of some good resources related to setting up heirarchical user account systems? I'm currently setting one up and am struggling with some of the more complex logic (especially with determining permissions). I was hoping I might be able to find some resources to help me along.

Some Background: I'm building a user account system for a web CMS that allows for a nested group hierarchy. Each group can be allowed/denied access to read, write, add, and delete (either explicitly for that group, or implicitly by one of its parents). As if that weren't complicated enough, the system also allows for users to be members of multiple groups. -- This is where I'm stuck. I've got everything set up, but I'm struggling with the actual logic for determining pemissions for a given user.

permissions theory usergroups

edited Jan 3, 2018 at 9:39

ekad

14.6k • 26 • 46 • 48

Share

Improve this question

Follow

asked Sep 16, 2008 at 18:28

Wilco

33.3k ● 49 ● 132 ● 161

4 Answers

Sorted by: Highest score (default)

3

The manual for CakePHP has an excellent description of how Access Control Lists work.

http://book.cakephp.org/2.0/en/core-libraries/components/access-control-lists.html

Share Improve this answer

Follow

edited May 23, 2014 at 13:02

Brad Knowles

119 ● 1 ● 1 ● 9

answered Sep 16, 2008 at 18:34

davethegr8

11.6k ● 5 ● 40 ● 61

2

Represent the permissions set for a given group as a bit mask. OR-ing the bit masks together will give you the resultant permission set.

Update for @Alex:

I wrote this answer 3 years ago, but I believe I was alluding to the following...

From the question

> a nested group hierarchy. Each group can be allowed/denied access to read, write, add, and delete (either explicitly for that group, or implicitly by one of its parents). As if that weren't complicated enough, the system also allows for users to be members of multiple groups. -- This is where I'm stuck. I've got everything set up, but I'm struggling with the actual logic for determining pemissions for a given user.

Assign a bitmask matching the total permission set of a group (or role) in the system:

e.g. `00` (using two bits keeps it simple here!)

The first bit confers `Permission A` and the second `Permission B`.

Now say Group A confers the following permission set: `01`.

... and say Group B confers the following permission set: `10`.

To get the resultant permission set for a user in an arbitrary set of groups you could perform a logical `OR` on the permission set bit masks:

```
Permission set for Group A   01
Permission set for Group B   10 OR
                             ----
Resultant permission set     11 (i.e. both
permission A and B are conferred)
```

I do not know the details of the questioner's system, but the system outlined here could be augmented to achieve different group-composition behaviors using different logical operators.

Share Improve this answer Follow

edited Apr 1, 2012 at 20:24

answered Jan 19, 2009 at 15:48

Ben Aston

55.6k ● 69 ● 219 ● 344

1 Hi @Ben, this looks interesting, could you elaborate on this perhaps with a mini example? Or perhaps a link to a useful resource? – Alex KeySmith Mar 29, 2012 at 10:58

1

Look at the permissions in the Andrew File System. It allows users to create and administer groups of their own, while selectively assigning admin rights and ACLs. You might find that many of the pesky details are already worked out for you in their model.

Edit: here's a better link to AFS documentation:

http://www.cs.cmu.edu/~help/afs/index.html

Here's the section on groups:

http://www.cs.cmu.edu/~help/afs/afs_groups.html

Share Improve this answer

Follow

answered Sep 16, 2008 at 18:31

Todd Gamblin

59.7k ● 15 ● 91 ● 96

Could you update the links if they still exist somewhere?
– yokto Sep 18, 2018 at 12:26

1

I've done exactly this before and its no trivial implementation. You're going to want to look at the SecurityPermission class.

[http://msdn.microsoft.com/en-us/library/system.security.permissions.securitypermission.aspx][1]

I have done this before by utilizing XML (which I'm not sure I'd do again) and storing that XML as permission list inside of SQL server in an XML column through a CLR stored proc. The XML would have an element called a "permission" and then the permission would actually be a ENUM inside of the code. Each permission was a new implementation of the SecurityPermission class (linked above) Users were tied to groups which were defined in SQL server and then as the user was added/removed to groups, the XML doc would get updated to reflect which groups they were apart of.

As soon as the user logged in, the users credentials would be loaded into the application store (session) and then would be accessed accordingly. When authorization needed to take place the XMl in the application store would be pulled down loaded into the SecurityPermission via the "FromXML" method. At that point I would use the following methods to determine if the user had permission:

- Demand
- Intersect
- Union
- IsUnrestricted
- IsSubSetOf

etc., etc, etc.

At that point after performing the Demand I was able to determine if the caller had access according to how I implemented my security routines in the SecurityPermissions.

Again, this is leaving out a TON of detail, but this should get you going down the right path.

Take a look at this name space as well: [2]: http://msdn.microsoft.com/en-us/library/system.security.permissions.aspx "System.Security.Permissions"

Share Improve this answer

Follow

answered Sep 16, 2008 at 18:39

Donn Felker

9,651 ● 8 ● 50 ● 68

Hi @Donn this looks really interesting, sounds like a great blog post. I noticed you posted this a few years back, from your experience is there anything you would approach differently now? – Alex KeySmith Mar 29, 2012 at 10:47