Zend_Rbac or Zend~Access~Rbac - Dolf Schimmel

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

> ⚠ This component will be ZF >=2.0 only and will not be shipped with a (tentative) ZF1.11

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_RBAC Component Proposal

Proposed Component Name	Zend_RBAC
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_RBAC
Proposers	Dolf Schimmel (Freeaqingme)
Zend Liaison	TBD
Revision	0.5 - 17 December 1909: Initial Draft. (wiki revision: 13)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

Zend_RBAC is a component used to determine if a user (subject), member of one or more roles, has access to a resource.

2. References

- 1 RBAC Specification
- 2 Wikipedia

3. Component Requirements, Constraints, and Acceptance Criteria

- A *user* CAN be member of one or more *roles*
- A *role* CAN be the parent of one or more other *roles*

- An *object* CAN be assigned to one or more *roles*
- If a *user* is member of a *role* (or any of the *role's* childs) to which the requested *object* was assigned, the *user* MUST be allowed access. Otherwise, MUST NOT.

4. Dependencies on Other Framework Components

Soft;

- Zend_Exception

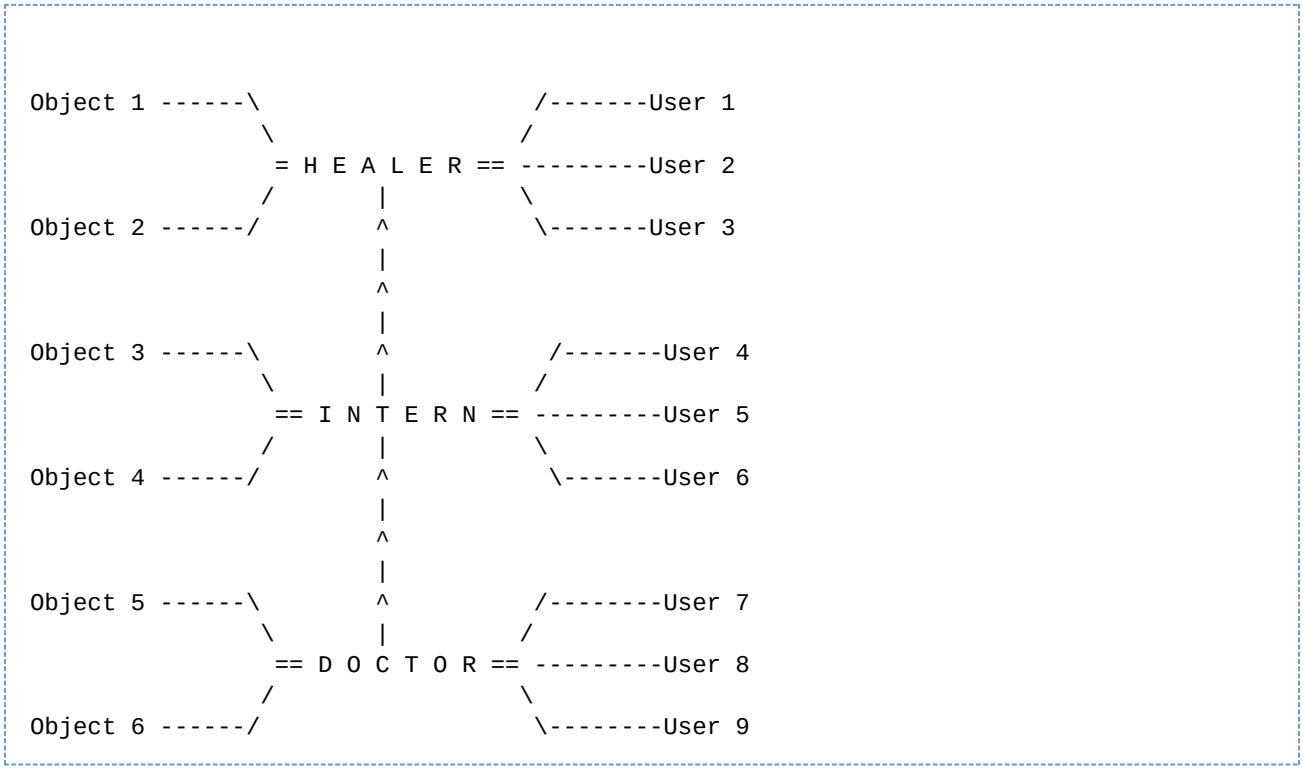
Optional:

- Zend_Db

5. Theory of Operation

Zend_Rbac is a component that can be used to determine whether a subject (user) that has one or more roles, has access to one or more resources, after you have defined the available resources, roles and users. One does only *allow* (not disallow) roles access to resources, assign users to roles, and determine if any roles inherit from other roles.

Following example based on [1]:



Interns can do everything a healer can do
Doctors can do everything an intern can

Healers have access to object 1 & 2 only
Interns have access to object 1, 2, 3 & 4 only
Doctors have access to object 1, 2, 3, 4, 5 & 6 only

This component is different than Zend_Acl because:

- Zend_Rbac does not feature the disallowing of access, only allowing access, meaning that there's no problems by design (as is with Zend_Acl: ZF-5369)
- It makes a distinction between roles and users (subjects) whereas Zend_Acl only has resources and roles
- It's faster and uses less lines of code (that's what I'm aiming at)
- It's adapter based by default
- It does not use privileges *

* If you want to replicate this behavior, you're advised to use several resources to achieve this. Meaning that where with zend_acl you have one resource named 'resource' with the privileges 'add', 'edit' and 'delete', you're expected to add the resources 'resource_add', 'resource_edit', and 'resource_delete'.

Some users (on IRC) have suggested to replace this component with Zend_Acl in ZF2.0 because it essentially does the same: determine if a certain user/role has access to a certain resource (but then done better). If there's a majority in the community that wishes to replace this component with Zend_Acl, it should definitely be considered.

6. Milestones / Tasks

- Milestone 1: [DONE] Design notes will be published here
- Milestone 2: [DONE] Write unittests
- Milestone 3: [DONE] Write working (basic) prototype, commit in userbranch
- Milestone 4: [DONE] Write Resource Plugin, assertions, adapters & FC-lugin => userbranch
- Milestone 5: Drop class interface here (or emphasize link to userbranch, hey, developers are lazy by definition)
- Milestone 6: Get proposal reviewed, updated, accepted
- Milestone 7: Write documentation (or find s/o to do so, volunteers?)
- Milestone 8: Get code reviewed by Matthew/liaison and move to trunk
- Milestone 9: Release immediately, and don't forget to promote Zym meanwhile

7. Class Index

- Zend_Rbac
- Zend_Rbac_Exception
- Zend_Rbac_Object
- Zend_Rbac_ObjectInterface
- Zend_Rbac_Resource (interface)
- Zend_Rbac_Role (interface)
- Zend_Rbac_Subject (interface)

- Zend_Rbac_Object_Resource
- Zend_Rbac_Object_Role
- Zend_Rbac_Object_Subject

Todo (read: not in userbranch):

- Zend_Rbac_Adapter_DbTable
- Zend_Rbac_Adapter_Abstract
- Zend_Rbac_Assert_?
- Zend_App_Resource_Rbac
- Zend_Controller_Front_Plugin_Rbac
- More_To_Come?

8. Use Cases

UC-01

The following code implements the given scenario under 'operation'

```
<?php
$rbac = new Zend_Rbac();
$rbac->assign('healer',array('user1','user2','user3'));

$internRole = new Zend_Rbac_Role('intern');
$user5 = new Zend_Rbac_Subject('user5');

$rbac->assign($internRole, 'user4');
$rbac->assign('intern', $user5);
$rbac->assign('intern', 'user6');

$rbac->assign('doctor', array('user7','user8','user9');

$rbac->setChild('intern', 'healer'); // <parent>, <child>
$rbac->setChild('doctor', 'intern');

$rbac->subscribe('healer', array('object1', 'object2');
$rbac->subscribe('intern', array('object3', 'object4');
$rbac->subscribe('doctor', array('object5', 'object6');

$rbac->isAllowed('user1', 'object1'); // True
$rbac->isAllowed('user1', 'object3'); // False
$rbac->isAllowed('user4', 'object1'); // True
$rbac->isAllowed('user4', 'object3'); // True
$rbac->isAllowed('user4', 'object5'); // False
$rbac->isAllowed('user9', 'object1'); // True
$rbac->isAllowed('user9', 'object3'); // True
$rbac->isAllowed('user9', 'object5'); // True
```

Alternatively you can use your models, by having them implement the correct interface, and adding a __toString() method that returns a unique identifier.

```
Model_User implements Zend_Rbac_Subject {

    protected $_userId = 'userName';

    public function __toString() {
         return $this->_userId;
    }
}

$rbac = new Zend_Rbac();
$rbac->addSubject(new Model_User());
```

9. Class Skeletons

```
class Zend_Rbac_Exception extends Zend_Exception {}
```

For the rest of the code, please see GitHub
]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>