Zend_Acl_Db - Ben Dauphinee

<ac:macro ac:name="note"><ac:parameter ac:name="title">Under Construction</ac:parameter><ac:rich-text-body>
This proposal is under construction and is not ready for review. </ac:rich-text-body></ac:macro>

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

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

Zend Framework: Zend_Acl_Db Component Proposal

Proposed Component Name	Zend_Acl_Db
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Acl_Db
Proposers	Ben Dauphinee
Zend Liaison	TBD
Revision	1.0 - 3 June 2010: Initial Draft. (wiki revision: 8)

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_Acl_Db is a complex ACL class that requires and uses a database for all records, allowing more flexibility in use and management.

2. References

- Role-based access control Wikipedia Entry

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** use a database for all rule storage.
- The config file **will** use Zend_Db.

4. Dependencies on Other Framework Components

- Zend_DB

5. Theory of Operation

This component is used as part of an access control scheme, as well as for management of acl data.

6. Milestones / Tasks

- Milestone 1: [DONE] Working prototype written and tested.
- Milestone 2: Unit testing converted to PHPUnit.
- Milestone 3: Create initial documentation.
- Milestone 4: Working code and tests checked into SVN.
- Milestone 5: Submitted to Ready for Review.

7. Class Index

- Zend_Acl_Db

8. Use Cases

UC-01

...

9. Class Skeletons

```
class Zend_Acl_Db{
    protected $db = null;
    protected $dataQuery = null;

    // Build class with base config and database connection
    public function __construct(Zend_Db_Adapter_Abstract $db, $config = array()){
        $this->db = $db;
        if(!empty($config)){$this->config = $config;}
    }

    // Function to build the acl query, or to select extra data
    protected function _buildQuery($selectType = null, $vars = array()){
    }

    protected function _bq_attachWhere($whereID = null, $vars = array()){
```

```
}

// Function to grab permission keys
private function _helper_setpermkey(&$vars){
}

private function _helper_group_setquerytype($vars){
}

// Sets the query type to group, user, role
private function _helper_gru_setquerytype($vars){
}

private function _helper_gru_sanitygru_norole($qt, $vars){
}

// Builds a recursive array of permission keys, traversing either up or down
private function _permission_tree($permission = null, $direction = 'd'){
}

// Checks that required variables are set
protected function _sanityRun_acl($sanitycheck, &$vars, $qt = null){
}

// Set operator uid as uid
private function _helper_resowneroverride($qt, $vars){
}

// Set operator uid as uid
private function _helper_set_op_uasu($vars){
}

// Set supplicant uid as uid
private function _helper_set_sup_uasu($vars){
}

public function check_permission_uaccess($permission = null, $uid = null){
}

public function check_resource_uaccess($resid, $uid = null){
}

public function check_resource_uperm($resid = null, $uid = null, $permkey = null){
}

// All group functions such as add, addmember, deactivate, etc
private function group($action = null, $vars = array()){
}

// All permission functionality such as tree, existance, access, etc
private function permission($action = null, $vars = array()){
}

// All resource functionality, such as info, access, existance, etc
private function resource($action = null, $vars = array()){
}

// All role functionality, such as create, grant, permadd, etc
private function role($action = null, $vars = array()){
}

// All user functionality, currently only existance check
private function user($action = null, $vars = array()){
}

// Public Interfaces (Many here for testing only)
public function group_create($groupname, $uid){
    return($this->group('create', array('groupname' => $groupname, 'uid' => $uid)));
} // integer groupid // Create group
public function group_deactivate($groupid, $uid){
```

```
        return($this->group('deactivate', array('groupid' => $groupid, 'uid' => $uid)));
    } // Deactivate group
    public function group_exists($groupid){
        return($this->group('exists', array('groupid' => $groupid)));
    } // Group exists
    public function group_getid($groupname){
        return($this->group('getid', array('groupname' => $groupname)));
    } // groupid // Get group id from name
    public function group_umember($groupid, $supplicant_uid){
        return($this->group('umember', array('groupid' => $groupid, 'supplicant_uid' =>
$supplicant_uid)));
    } // Is user member of
    public function group_umemberships($supplicant_uid){
        return($this->group('umemberships', array('supplicant_uid' => $supplicant_uid)));
    } // array groupids // Groups that user is member of
    public function group_uadd($groupid, $supplicant_uid, $operator_uid){
        return($this->group('uadd', array('groupid' => $groupid, 'supplicant_uid' => $supplicant_uid,
'operator_uid' => $operator_uid)));
    } // Add a user to group
    public function group_urem($groupid, $supplicant_uid, $operator_uid){
        return($this->group('urem', array('groupid' => $groupid, 'supplicant_uid' => $supplicant_uid,
'operator_uid' => $operator_uid)));
    } // Remove a user from group
    public function permission_exists($permkey){
        return($this->permission('exists', array('permkey' => $permkey)));
    } // bool checkresult // Check permkey exists
    public function permission_gaccess($permkey, $groupid){
        return($this->permission('access', array('permkey' => $permkey, 'groupid' => $groupid)));
    } // bool state // Have permission globally
    public function permission_glist($permkey){
        return($this->permission('list', array('permkey' => $permkey, 'groupid' => 1)));
    } // array allow/deny group // List that have this permkey
    public function permission_raccess($permkey, $roleid){
        return($this->permission('access', array('permkey' => $permkey, 'roleid' => $roleid)));
    } // bool state // Have permission globally
    public function permission_rlist($permkey){
        return($this->permission('list', array('permkey' => $permkey, 'roleid' => 1)));
    } // array allow/deny role // List that have this permkey
    public function permission_tree($permkey, $type = null){
        return($this->permission('tree', array('permkey' => $permkey, 'type' => $type)));
    } // array keys // Get tree
    public function permission_uaccess($permkey, $uid){
        return($this->permission('access', array('permkey' => $permkey, 'uid' => $uid)));
    } // bool state // Have permission globally
    public function permission_ulist($permkey){
        return($this->permission('list', array('permkey' => $permkey, 'uid' => 1)));
    } // array allow/deny user // List that have this permkey
    public function resource_exists($resid){
        return($this->resource('exists', array('resid' => $resid)));
    } // bool exists // Resource exists
    public function resource_gaccess($groupid, $resid){
        return($this->resource('access', array('resid' => $resid, 'groupid' => $groupid)));
    } // bool allowed // Have access
    public function resource_glist($resid){
        return($this->resource('list', array('resid' => $resid, 'groupid' => 1)));
    } // array allow/deny // List that have this permkey
    public function resource_gperm($resid, $groupid, $permkey){
        return($this->resource('perm', array('resid' => $resid, 'groupid' => $groupid, 'permkey' =>
$permkey)));
    } // bool state // Does have specific permission on resource
    public function resource_gpermgrant($resid, $groupid, $permkey, $operator_uid){
        return($this->resource('permgrant', array('resid' => $resid, 'groupid' => $groupid, 'permkey'
=> $permkey, 'operator_uid' => $operator_uid)));
    } // Grant permission for resource
    public function resource_gpermrevoke($resid, $groupid, $permkey, $operator_uid){
        return($this->resource('permrevoke', array('resid' => $resid, 'groupid' => $groupid, 'permkey'
=> $permkey, 'operator_uid' => $operator_uid)));
    } // bool state // Revoke permission for resource
    public function resource_info($resid){
        return($this->resource('info', array('resid' => $resid)));
```

```
    } // array resinfo // Resource info
    public function resource_ownerid($resid){
        return($this->resource('ownerid', array('resid' => $resid)));
    } // ownerid // Get ownerid
    public function resource_uaccess($resid, $uid){
        return($this->resource('access', array('resid' => $resid, 'supplicant' => $uid, 'uid' =>
$uid)));
    } // bool allowed // Have access
    public function resource_ulist($resid){
        return($this->resource('list', array('resid' => $resid, 'uid' => 1)));
    } // array allow/deny // List that have this permkey
    public function resource_uperm($resid, $uid, $permkey){
        return($this->resource('perm', array('resid' => $resid, 'uid' => $uid, 'permkey' =>
$permkey)));
    } // bool state // Does have specific permission on resource
    public function resource_upermgrant($resid, $permkey, $operator_uid, $supplicant_uid){
        return($this->resource('permgrant', array('resid' => $resid, 'permkey' => $permkey,
'operator_uid' => $operator_uid, 'supplicant_uid' => $supplicant_uid)));
    } // Grant permission for resource
    public function resource_upermrevoke($resid, $permkey, $operator_uid, $supplicant_uid){
        return($this->resource('premrevoke', array('resid' => $resid, 'permkey' => $permkey,
'operator_uid' => $operator_uid, 'supplicant_uid' => $supplicant_uid)));
    } // bool state // Revoke permission for resource
    public function role_create($rolename, $uid){
        return($this->role('create', array('rolename' => $rolename, 'uid' => $uid)));
    } // roleid // Create a role
    public function role_delete($roleid, $uid){
        return($this->role('delete', array('roleid' => $roleid, 'uid' => $uid)));
    } // Delete a role
    public function role_exists($roleid){
        return($this->role('exists', array('roleid' => $roleid)));
    } // bool state // Does role exist
    public function role_ggrant($roleid, $groupid, $operator_uid, $supplicant_uid){
        return($this->role('grant', array('roleid' => $roleid, 'groupid' => $groupid, 'context' =>
'global', 'operator_uid' => $operator_uid, 'supplicant_uid' => $supplicant_uid)));
    } // Add to role
    public function role_glist($roleid){
        return($this->role('list', array('roleid' => $roleid, 'groupid' => 1)));
    } // array allow/deny // Show attached to role
    public function role_gmember($roleid, $groupid){
        return($this->role('member', array('roleid' => $roleid, 'groupid' => $groupid, 'context' =>
'CONTEXTCONTEXTCONTEXT')));
    } // bool member // Is member of
    public function role_gmemberships($groupid){
        return($this->role('memberships', array('groupid' => $groupid)));
    } // array roleids // Roles that is member of
    public function role_grevoke($roleid, $groupid, $operator_uid, $supplicant_uid){
        return($this->role('revoke', array('roleid' => $roleid, 'groupid' => $groupid, 'context' =>
'global', 'operator_uid' => $operator_uid, 'supplicant_uid' => $supplicant_uid)));
    } // Remove from role
    public function role_permadd($roleid, $permkey, $permscope, $uid){
        return($this->role('permadd', array('permkey' => $permkey, 'permscope' => $permscope, 'roleid'
=> $roleid, 'uid' => $uid)));
    } // Add permission to role
    public function role_permlist($roleid){
        return($this->role('permlist', array('roleid' => $roleid)));
    } // array allow/deny // List permissions attached to role
    public function role_permrem($roleid, $permkey, $permscope, $uid){
        return($this->role('permrem', array('roleid' => $roleid, 'permkey' => $permkey, 'permscope' =>
$permscope, 'uid' => $uid)));
    } // Remove permission
    public function role_ulist($roleid){
        return($this->role('list', array('roleid' => $roleid, 'uid' => 1)));
    } // array allow/deny // Show attached to role
    public function role_umember($roleid, $uid){
        return($this->role('member', array('roleid' => $roleid, 'uid' => $uid)));
    } // bool member // Is member of
    public function role_umemberships($uid){
        return($this->role('memberships', array('uid' => $uid)));
    } // array roleids // Roles that is member of
```

```
    public function role_ugrant($roleid, $operator_uid, $supplicant_uid){
        return($this->role('grant', array('roleid' => $roleid, 'operator_uid' => $operator_uid,
'supplicant_uid' => $supplicant_uid, 'context' => 'global')));
    } // Add to role
    public function role_urevoke($roleid, $operator_uid, $supplicant_uid){
        return($this->role('revoke', array('roleid' => $roleid, 'operator_uid' => $operator_uid,
'supplicant_uid' => $supplicant_uid, 'context' => 'global')));
    } // Remove from role
    public function user_exists($uid){
```

```
        return($this->user('exists', array('uid' => $uid)));
    }
}
```

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>