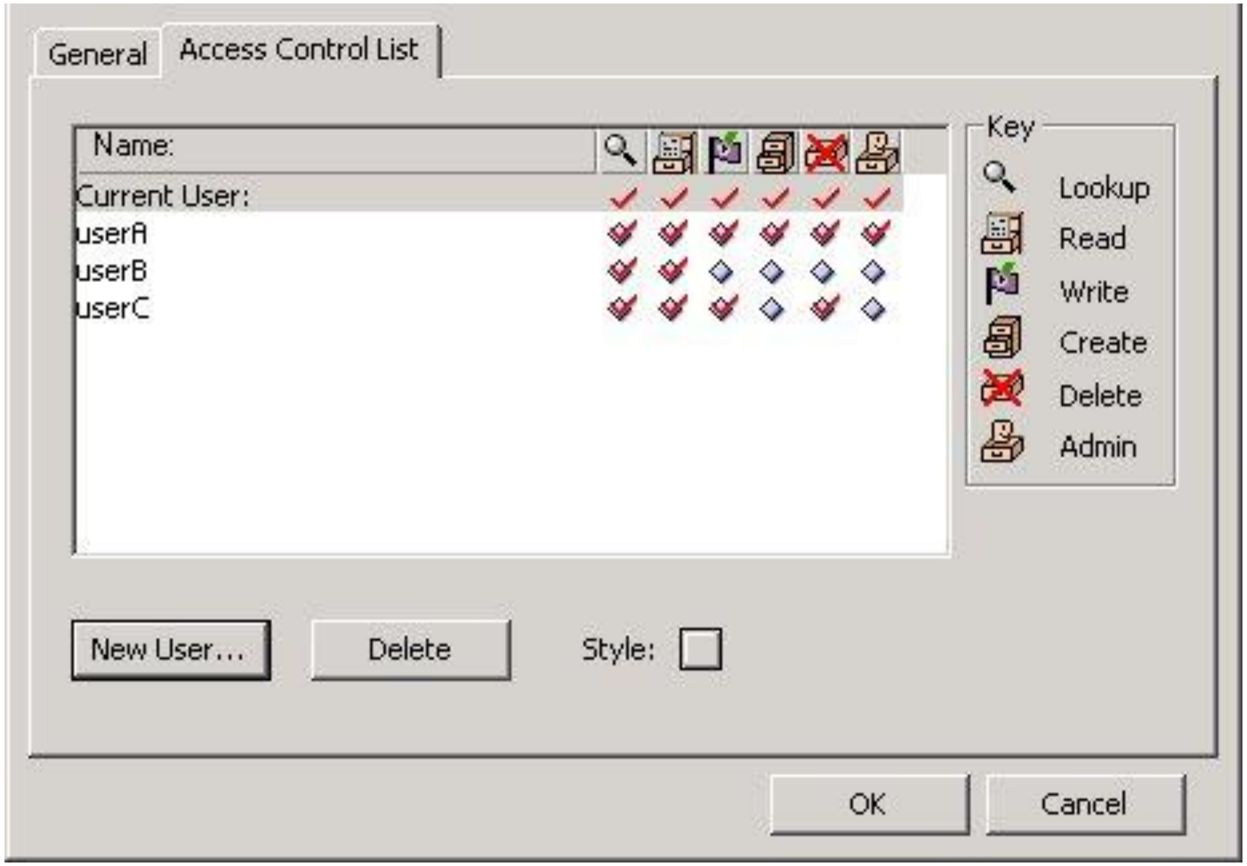
ACL in CodeIgniter

a simpler approach

What is ACL?

- ACL stands for Access Control List
- Used for Authorization purpose (eg: who access what)
- Zend, CakePHP features ACL as key component of their framework
- What about CodeIgniter?

A graphical view



Name:	🔍	🗄	⚑	🗄	✗	👤
Current User:	✓	✓	✓	✓	✓	✓
userA	✓	✓	✓	✓	✓	✓
userB	✓	✓	◇	◇	◇	◇
userC	✓	✓	✓	◇	✓	◇

Key

🔍	Lookup
🗄	Read
⚑	Write
🗄	Create
✗	Delete
👤	Admin

Style: ☐

Then!! How do we do authorization in CodeIgniter?

- Using our custom control check

```
function loadPage($pageID)
{
    if($_SESSION['userType'] != "member")
    {
        die("you do not have access to this page");
    }
}
```

- And the check goes on & on, Hard coded in our controller files…..

Hook – a powerful feature of CodeIgniter to do the rescue

- What is this hook?
  - a means to tap into and modify the inner workings of the framework without hacking the core files
  - Have you heard of wordpress or mediawiki hook?
  - Examples:
    - We want to execute a functionality before controllers are loaded

Defining Hooks

- Hooks must be enabled in CodeIgniter Config file
  $config['enable_hooks'] = True;
- Hooks are defined in *application/config/hooks.php* file. Each hook is specified as a part of a global array named $hook

```
$hook['Hook_Point'] = array(
                          'class'    => 'MyClass',
                          'function' => 'Myfunction',
                          'filename' => 'Myclass.php',
                          'filepath' => 'hooks',
                          'params'   => array()
                          );
```

Hook Point

- **pre_system**
  Called very early during system execution. Only the benchmark and hooks class have been loaded at this point. No routing or other processes have happened.
- **pre_controller**
  Called immediately prior to any of your controllers being called. All base classes, routing, and security checks have been done.
- **post_controller_constructor**
  Called immediately after your controller is instantiated, but prior to any method calls happening.
- **post_controller**
  Called immediately after your controller is fully executed.

- **class** The name of the class you wish to invoke. If you prefer to use a procedural function instead of a class, leave this item blank.
- **function** The function name you wish to call.
- **filename** The file name containing your class/function.
- **filepath** The name of the directory containing your script. Note: Your script must be located in a directory INSIDE your application folder, so the file path is relative to that folder. For example, if your script is located in *application/hooks*, you will simply use hooks as your filepath. If your script is located in *application/hooks/utilities* you will use hooks/utilities as your filepath. No trailing slash.
- **params** Any parameters you wish to pass to your script. This item is optional.

Lets start our ACL implementation

```
/* application/config/hooks.php */

$hook['pre_controller'] = array(
                                'class' => 'Accesscheck',
                                'function' => 'index',
                                'filename' => 'accesscheck.php',
                                'filepath' => 'hooks');
```

application/hooks/accesscheck.php

```
class Accesscheck
{
   public  function index($params)
   {
         require_once('permissions.php');
         $baseURL = $GLOBALS['CFG']->config['base_url'];
         $routing =& load_class('Router');
         $class  = $routing->fetch_class();
         $method = $routing->fetch_method();
```

```
if(! empty($doesNotRequireLogin[$class][$method])) {  return true; }
else {
            if(! $_SESSION['userType']) {       //checking authentication
                     header("location: {$baseURL}common/login"); exit;
            }
            else {

if(empty($permissions[$_SESSION['userType']][$class][$method])
                     ||
$permissions[$_SESSION['userType']][$class][$method]!=true) {

                     header("location: {$baseURL}common/unauthorized");
exit;
                     } else {
                            return true;
                     }
                 }
      }
          header("location: {$baseURL}common/unauthorized");
```

Permissions.php

```
<?php
$doesNotRequireLogin = array();
$permissions = array();
$doesNotRequireLogin['common']['index'] = true;
$doesNotRequireLogin['common']['login'] = true;
$doesNotRequireLogin['common']['dologin'] = true;
$doesNotRequireLogin['common']['unauthorized'] = true;
$doesNotRequireLogin['common']['message'] = true;
$doesNotRequireLogin['common']['forgotpassword'] = true;
```

```
$permissions[‘member’][‘blog'][‘post’] = true;
$permissions[‘member’][‘blog'][‘view’] = true;
$permissions[‘member’][‘blog'][‘save’] = true;
$permissions[‘member’][‘blog'][‘rating’] = true;
$permissions[‘guest’][‘blog'][‘view’] = true;
```

What we achieved from this?

- We have eliminated the process of writing the authorization code on each controller functions
- We have a better authorized application
- We have a central access point with permissions and check.
- We have used Array for better performance (you can use XML though)
- This solution is better suited for role based access as well as dynamic role option.

Other available methodologies

- ACL as a Library
  - There are few libraries available from CodeIgniter wiki page and other sources which can be used for ACL purpose.

THANK YOU

M. MIZANUR RAHMAN
Founder & C.T.O
Informatix Technologies
[mizan@informatixbd.com]