**Zend Auth**

Reference :- <https://samsonasik.wordpress.com/2012/10/23/zend-framework-2-create-login-authentication-using-authenticationservice-with-rememberme/>

**Register Auth Service into ServiceManager in Module class**

//module/SanAuth/Module.php

namespace SanAuth;

use Zend\ModuleManager\Feature\AutoloaderProviderInterface;

use Zend\Authentication\Storage;

use Zend\Authentication\AuthenticationService;

use Zend\Authentication\Adapter\DbTable as DbTableAuthAdapter;

class Module implements AutoloaderProviderInterface

{

public function getAutoloaderConfig(){/\*common code\*/}

public function getConfig(){ /\*common code\*/}

public function getServiceConfig()

{

return array(

'factories'=>array(

'SanAuth\Model\MyAuthStorage' => function($sm){

return new \SanAuth\Model\MyAuthStorage('zf\_tutorial');

},

'AuthService' => function($sm) {

$dbAdapter = $sm->get('Zend\Db\Adapter\Adapter');

$dbTableAuthAdapter = new DbTableAuthAdapter($dbAdapter, 'users','user\_name','pass\_word', 'MD5(?)');

$authService = new AuthenticationService();

$authService->setAdapter($dbTableAuthAdapter);

$authService->setStorage($sm->get('SanAuth\Model\MyAuthStorage'));

return $authService;

},

),

);

}

}

**Create the Auth Controller**

//module/SanAuth/src/SanAuth/Controller/AuthController.php

namespace SanAuth\Controller;

use Zend\Mvc\Controller\AbstractActionController;

use Zend\Form\Annotation\AnnotationBuilder;

use Zend\View\Model\ViewModel;

use SanAuth\Model\User;

class AuthController extends AbstractActionController

{

protected $form;

protected $storage;

protected $authservice;

public function getAuthService()

{

if (! $this->authservice) {

$this->authservice = $this->getServiceLocator()

->get('AuthService');

}

return $this->authservice;

}

public function getSessionStorage()

{

if (! $this->storage) {

$this->storage = $this->getServiceLocator()

->get('SanAuth\Model\MyAuthStorage');

}

return $this->storage;

}

public function getForm()

{

if (! $this->form) {

$user = new User();

$builder = new AnnotationBuilder();

$this->form = $builder->createForm($user);

}

return $this->form;

}

public function loginAction()

{

//if already login, redirect to success page

if ($this->getAuthService()->hasIdentity()){

return $this->redirect()->toRoute('success');

}

$form = $this->getForm();

return array(

'form' => $form,

'messages' => $this->flashmessenger()->getMessages()

);

}

public function authenticateAction()

{

$form = $this->getForm();

$redirect = 'login';

$request = $this->getRequest();

if ($request->isPost()){

$form->setData($request->getPost());

if ($form->isValid()){

//check authentication...

$this->getAuthService()->getAdapter()

->setIdentity($request->getPost('username'))

->setCredential($request->getPost('password'));

$result = $this->getAuthService()->authenticate();

foreach($result->getMessages() as $message)

{

//save message temporary into flashmessenger

$this->flashmessenger()->addMessage($message);

}

if ($result->isValid()) {

$redirect = 'success';

//check if it has rememberMe :

if ($request->getPost('rememberme') == 1 ) {

$this->getSessionStorage()

->setRememberMe(1);

//set storage again

$this->getAuthService()->setStorage($this->getSessionStorage());

}

$this->getAuthService()->getStorage()->write($request->getPost('username'));

}

}

}

return $this->redirect()->toRoute($redirect);

}

public function logoutAction()

{

$this->getSessionStorage()->forgetMe();

$this->getAuthService()->clearIdentity();

$this->flashmessenger()->addMessage("You've been logged out");

return $this->redirect()->toRoute('login');

}

}

**Create Success Controller**

//module/SanAuth/src/SanAuth/Controller/SuccessController.php

namespace SanAuth\Controller;

use Zend\Mvc\Controller\AbstractActionController;

use Zend\View\Model\ViewModel;

class SuccessController extends AbstractActionController

{

    public function indexAction()

    {

        if (! $this->getServiceLocator()

                 ->get('AuthService')->hasIdentity()){

            return $this->redirect()->toRoute('login');

        }

        return new ViewModel();

    }

}

**Access Control List (*ACL*) in Zend 2**

Reference : - <https://framework.zend.com/manual/2.2/en/modules/zend.permissions.acl.intro.html>

**Defining Access Controls**

**use** Zend\Permissions\Acl\Acl;

**use** Zend\Permissions\Acl\Role\GenericRole **as** Role;

$acl = **new** Acl();

$roleGuest = **new** Role('guest');

$acl->addRole($roleGuest);

$acl->addRole(**new** Role('staff'), $roleGuest);

$acl->addRole(**new** Role('editor'), 'staff');

$acl->addRole(**new** Role('administrator'));

*// Guest may only view content*

$acl->allow($roleGuest, **null**, 'view');

/\*

Alternatively, the above could be written:

$acl->allow('guest', null, 'view');

//\*/

*// Staff inherits view privilege from guest, but also needs additional*

*// privileges*

$acl->allow('staff', **null**, **array**('edit', 'submit', 'revise'));

*// Editor inherits view, edit, submit, and revise privileges from*

*// staff, but also needs additional privileges*

$acl->allow('editor', **null**, **array**('publish', 'archive', 'delete'));

*// Administrator inherits nothing, but is allowed all privileges*

$acl->allow('administrator');

## Querying an ACL[¶](https://framework.zend.com/manual/2.2/en/modules/zend.permissions.acl.intro.html#querying-an-acl)

We now have a flexible ACL that can be used to determine whether requesters have permission to perform functions throughout the web application. Performing queries is quite simple using the isAllowed()method:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | **echo** $acl->isAllowed('guest', **null**, 'view') ?  "allowed" : "denied";  *// allowed*  **echo** $acl->isAllowed('staff', **null**, 'publish') ?  "allowed" : "denied";  *// denied*  **echo** $acl->isAllowed('staff', **null**, 'revise') ?  "allowed" : "denied";  *// allowed*  **echo** $acl->isAllowed('editor', **null**, 'view') ?  "allowed" : "denied";  *// allowed because of inheritance from guest*  **echo** $acl->isAllowed('editor', **null**, 'update') ?  "allowed" : "denied";  *// denied because no allow rule for 'update'*  **echo** $acl->isAllowed('administrator', **null**, 'view') ?  "allowed" : "denied";  *// allowed because administrator is allowed all privileges*  **echo** $acl->isAllowed('administrator') ?  "allowed" : "denied";  *// allowed because administrator is allowed all privileges*  **echo** $acl->isAllowed('administrator', **null**, 'update') ?  "allowed" : "denied";  *// allowed because administrator is allowed all privileges* |

## Removing Access Controls[¶](https://framework.zend.com/manual/2.2/en/modules/zend.permissions.acl.refining.html#removing-access-controls)

To remove one or more access rules from the ACL, simply use the available removeAllow() or removeDeny() methods. As with allow() and deny(), you may provide a NULL value to indicate application to all roles, resources, and/or privileges:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | *// Remove the denial of revising latest news to staff (and marketing,*  *// by inheritance)*  $acl->removeDeny('staff', 'latest', 'revise');  **echo** $acl->isAllowed('marketing', 'latest', 'revise') ?  "allowed" : "denied";  *// allowed*  *// Remove the allowance of publishing and archiving newsletters to*  *// marketing*  $acl->removeAllow('marketing',  'newsletter',  **array**('publish', 'archive'));  **echo** $acl->isAllowed('marketing', 'newsletter', 'publish') ?  "allowed" : "denied";  *// denied*  **echo** $acl->isAllowed('marketing', 'newsletter', 'archive') ?  "allowed" : "denied";  *// denied* |

Privileges may be modified incrementally as indicated above, but a NULL value for the privileges overrides such incremental changes:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | *// Allow marketing all permissions upon the latest news*  $acl->allow('marketing', 'latest');  **echo** $acl->isAllowed('marketing', 'latest', 'publish') ?  "allowed" : "denied";  *// allowed*  **echo** $acl->isAllowed('marketing', 'latest', 'archive') ?  "allowed" : "denied";  *// allowed*  **echo** $acl->isAllowed('marketing', 'latest', 'anything') ?  "allowed" : "denied";  *// allowed* |