ZF - Session 8

1. ACL primera aproximación

Vamos a implementar un control de acceso en el método onBootstrap del fichero Module.php

```
public function onBootstrap(MvcEvent $e)
{
    $eventManager        = $e->getApplication()->getEventManager();
    $moduleRouteListener = new ModuleRouteListener();
    $moduleRouteListener->attach($eventManager);

    $eventManager->attach(MvcEvent::EVENT_ROUTE, array($this, 'routeHandler'), -
100);
}

public function routeHandler(MvcEvent $event)
{
    $match = $event->getRouteMatch();

    if (!$match) { // we need a route
        return;
    }

    $controller     = $match->getParam('controller');
    $action         = $match->getParam('action');
    $roles          = $match->getParam('roles');

    $sm = $event->getApplication()->getServiceManager();
    $authenticationService = $sm->get('User\Service\Authentication');

    $role = ($identity = $authenticationService->getIdentity()) ? $identity->role : 'guest';

    if (!empty($roles) && !in_array($role, $roles)) {
        $response = $event->getResponse();
        $response->setStatusCode(401); // Auth required
        $match->setParam('controller', 'User\Controller\Users');
        $match->setParam('action', 'forbidden');
    }
}
```

Creamos un handler para el evento EVENT_ROUTE de manera que en función de la ruta de la petición y del role del usuario, restringiremos su acceso.

Para ello hemos añadido una key 'roles' en las rutas. Indicando qué roles tienen permiso en esa ruta. Si no se crea la key, se trata de una ruta pública.

Por ejemplo, solo pueden acceder a borrar un usuario aquellos perfiles con role admin:

```
'user\users\delete' => array(
    'type' => 'Segment',
    'options' => array(
        'route'    => '/admin/users/delete/id/[:id]/',
        'constraints' => array(
            'id' => '[0-9]+',
        ),
        'defaults' => array(
            'controller' => 'User\Controller\Users',
            'action'     => 'delete',
            'roles'      => ['admin'],
        ),
    ),
),
```

Los intentos no permitidos serán redirigidos al action 'forbiddenAction' del controller 'User\Controller\UsersController':

```
public function forbiddenAction()
{
    return [];
}
```

Con la vista forbidden.phtml:

```
<h1>Forbidden</h1>

<p class="lead">You must authenticate in order to access <code><?php echo $this->serverUrl(true); ?></code> in this server</p>

<div><a class="btn btn-default" href="<?php echo $this->url('user\login\login'); ?>">Login</a></div>
```

2. ACL

En el punto anterior hemos hecho una implementación de una lista de control de acceso utilizando la key 'roles' en las rutas manejando el evento EVENT_ROUTE. Pero no disponemos de un objeto ACL para poder realizar comprobaciones de control de acceso en otras partes del MVC.

Vamos a crear un servicio ACL que por tanto tendremos disponible vía ServiceLocator. Para también poder utilizarlo en las vistas, lo asignaremos al layout en el método onBootstrap del Module.php.

El servicio ACL nos permitirá gestionar roles, recursos y los permisos de ésos roles sobre cada recurso.

Los roles de nuestra app los fijaremos como una variable de configuración aunque se podrían cargar de una tabla de la base de datos.

Los recursos serán las rutas de nuestra aplicación.

Para saber qué roles tienen permisos de acceso sobre una ruta, seguiremos utilizando la key 'roles' en cada ruta. De manera que si esa key no está presente, dicho recurso se considera público. En caso contrario, solo será accesible por los roles indicados.

Seguiremos realizando el control de acceso en el evento EVENT_ROUTE pero en esta ocasión ya usaremos nuestro servicio ACL.

3. Creando el servicio ACL

Utilizaremos un Factory para crear el servicio ACL:

```php
<?php
/**
 * xenFramework (http://xenframework.com/)
 *
 * This file is part of the xenframework package.
 *
 * (c) Ismael Trascastro <itrascastro@xenframework.com>
 *
 * @link        http://github.com/xenframework for the canonical source repository
 * @copyright   Copyright (c) xenFramework. (http://xenframework.com)
 * @license     MIT License - http://en.wikipedia.org/wiki/MIT_License
 *
 * For the full copyright and license information, please view the LICENSE
 * file that was distributed with this source code.
 */

namespace User\Service\Factory;


use Zend\Permissions\Acl\Acl;
use Zend\ServiceManager\FactoryInterface;
use Zend\ServiceManager\ServiceLocatorInterface;

class AclServiceFactory implements FactoryInterface
{

    /**
     * Create service
     *
     * @param ServiceLocatorInterface $serviceLocator
     * @return mixed
     */
    public function createService(ServiceLocatorInterface $serviceLocator)
```

```
    {
        $acl = new Acl();

        $config = $serviceLocator->get('config');
        $roles = $config['application']['roles'];

        foreach ($roles as $role) {
            $acl->addRole($role);
        }

        $routes = $config['router']['routes'];

        foreach ($routes as $route => $value) {
            $acl->addResource($route);
            $routeRoles = array_key_exists('roles', $value['options']['defaults']) ?
$value['options']['defaults']['roles'] : $roles;
            $acl->allow($routeRoles, $route);
        }

        return $acl;
    }
}
```

Lo que hacemos es crear una instancia de la clase Zend\Permissions\Acl\Acl y luego seteamos los roles a partir de la variable de configuración.

Lo siguiente es recorrer todas las rutas guardadas en el fichero de configuración e ir creando permisos según los roles de cada ruta. Para ello utilizamos el método 'allow' al que le indicamos qué roles tienen permiso sobre qué ruta.

Una vez creado el Factory, registramos el servicio en el module.config.php:

```
'service_manager' => array(
    'aliases' => array(
        'Zend\Authentication\AuthenticationService' => 'User\Service\Authentication', //
needed for identity plugin
    ),
    'factories' => array(
        'User\Model\UsersModel'              => 'User\Model\Factory\UsersModelFactory',
        'User\Service\AuthenticationStorage'    =>
'User\Service\Factory\AuthenticationStorageServiceFactory',
        'User\Service\Authentication'          =>
'User\Service\Factory\AuthenticationServiceFactory',
        'User\Form\User'                    => 'User\Form\Factory\UserFormFactory',
        'User\Form\Login'                    => 'User\Form\Factory\LoginFormFactory',
        'User\Service\Acl'                  => 'User\Service\Factory\AclServiceFactory',
    ),
),
```

A partir de ahora ya podemos recuperarlo mediante el ServiceLocator.

4. Control de acceso en EVENT_ROUTE

Veamos ahora cómo queda nuestro handler para el evento EVENT_ROUTE qué habíamos puesto en el Module.php:

```php
public function routeHandler(MvcEvent $event)
{
    $match = $event->getRouteMatch();

    if (!$match) { // we need a route
        return;
    }

    $sm = $event->getApplication()->getServiceManager();
    $authenticationService = $sm->get('User\Service\Authentication');

    /**
     * @var Acl $acl
     */
    $acl = $sm->get('User\Service\Acl');

    $role = ($identity = $authenticationService->getIdentity()) ? $identity->role : 'guest';

    if (!$acl->isAllowed($role, $match->getMatchedRouteName())) {
        $response = $event->getResponse();
        $response->setStatusCode(401); // Auth required
        $match->setParam('controller', 'User\Controller\Users');
        $match->setParam('action', 'forbidden');
    }

    $event->getViewModel()->setVariable('acl', $acl);
}
```

El código es muy similar al que teníamos, solamente que ahora utilizamos el servicio 'User\Service\Acl'.

Recuperamos el role del usuario haciendo uso del Authentication Service, que en caso de no tener identidad (no está logueado) será 'guest'.

Y utilizamos el método isAllowed de ACL para comprobar si tiene acceso a la ruta de la petición. En caso de no tener permiso se le redirige cambiando los parámetros de la ruta a un action llamado 'forbidden'.

Por último ponemos el Servicio ACL a disposición de las vistas asignándolo a una variable de layout.

5. Uso de ACL en las vistas

```
<div class="lead"><a href="<?php echo $this->url('user\users\view', ['id' => $this->identity()->id]) ?>">@<?php echo $this->identity()->username; ?></a> Welcome to your Control Panel</div>

<div>
    <table class="table table-bordered">
        <tr>
            <th>Id</th>
            <th>Username</th>
            <th>Email</th>
            <th>Password</th>
            <th>Role</th>
            <th>Date</th>
        </tr>
        <?php foreach($this->users as $user): ?>
            <tr>
                <td><a href="<?php echo $this->url('user\users\view', ['id' => $user->getId()]) ?>"><?php echo $this->escapeHtml($user->getId()); ?></a></td>
                <td><?php echo $this->escapeHtml($user->getUsername()); ?></td>
                <td><?php echo $this->escapeHtml($user->getEmail()); ?></td>
                <td><?php echo $this->escapeHtml($user->getPassword()); ?></td>
                <td><?php echo $this->escapeHtml($user->getRole()); ?></td>
                <td><?php echo $this->escapeHtml($user->getDate()); ?></td>
                <?php if ($this->layout()->acl->isAllowed($this->identity()->role, 'user\users\update')): ?>
                <td><a class="btn btn-success" href="<?php echo $this->url('user\users\update', ['id' => $user->getId()]) ?>">Update</a></td>
                <?php endif; ?>
                <?php if ($this->layout()->acl->isAllowed($this->identity()->role, 'user\users\delete')): ?>
                <td><a class="btn btn-danger" href="<?php echo $this->url('user\users\delete', ['id' => $user->getId()]) ?>">Delete</a></td>
                <?php endif; ?>
            </tr>
        <?php endforeach; ?>
    </table>
</div>

<div>
    <p><a href="<?php echo $this->url('user\users\create') ?>">Create a new User</a> | <a href="<?php echo $this->url('user\login\logout') ?>">Logout</a> </p>
</div>
```

Donde vemos que solo mostramos los botones Modificar y Eliminar en caso de que el usuario tenga permisos para esas acciones.

En la vista debemos recuperar la variable de layout haciendo uso del View Helper layout().

```
$this->layout()->acl
```

6. Uso de ACL en un controller

Si quisiéramos usar el Servicio ACL en un controller, debemos recuperarlo a través del ServiceLocator o bien inyectarlo en el Factory de dicho controller.

```
$acl = $this->serviceLocator->get('User\Service\Acl');
```

Aquí tienes la documentación oficial del componente para futuras referencias:

http://framework.zend.com/manual/current/en/modules/zend.permissions.acl.intro.html