ZF - Session 7

1. Autenticación

Necesitamos ahora identificar perfiles de acceso para así decidir cuando mostrar el contenido en función de los privilegios del usuario logueado. Implementaremos un sistema de autenticación usando el componente

'Zend\Authentication\AuthenticationService'.

La clase Zend\Authentication\AuthenticationService tiene dos propiedades:

- storage
- adapter

1.1. Adapter

La propiedad adapter es un objeto de la clase

'Zend\Authentication\Adapter\DbTable\CredentialTreatmentAdapter'

Mediante este objeto podemos indicar qué tabla de nuestra base de datos será usada para identificar usuarios. Debemos indicar qué campo de la tabla será usado como identidad, qué columna será la credencial (password) y qué método se usará para generar la credencial.

```
$authAdapter = new CredentialTreatmentAdapter($adapter);
$authAdapter
    ->setTableName('User')
    ->setIdentityColumn('email')
    ->setCredentialColumn('password')
;
```

1.2. Storage

En la propiedad 'storage' indicaremos el tipo de almacenamiento que usaremos para guardar las identidades. Podemos usar por ejemplo sesiones:

```
<?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;


use Zend\Authentication\Storage\Session;

class AuthenticationStorageService extends Session
{
    public function setRememberMe($rememberMe = 0, $time = 2592000)
    {
        if ($rememberMe == 1) {
            $this->session->getManager()->rememberMe($time);
        }
    }

    public function forgetMe()
    {
        $this->session->getManager()->forgetMe();
    }
}
```

1.3. AuthenticationService

Con lo que la clase Factory para crear la instancia de AuthenticationService quedaría:

```
<?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;
```

```php
use Zend\Authentication\Adapter\DbTable\CredentialTreatmentAdapter;
use Zend\Authentication\AuthenticationService;
use Zend\ServiceManager\FactoryInterface;
use Zend\ServiceManager\ServiceLocatorInterface;

class AuthenticationServiceFactory implements FactoryInterface
{

    /**
     * Create service
     *
     * @param ServiceLocatorInterface $serviceLocator
     * @return mixed
     */
    public function createService(ServiceLocatorInterface $serviceLocator)
    {
        $adapter = $serviceLocator->get('database');

        $authAdapter = new CredentialTreatmentAdapter($adapter);
        $authAdapter
            ->setTableName('User')
            ->setIdentityColumn('email')
            ->setCredentialColumn('password')
        ;

        $storage = $serviceLocator->get('User\Service\AuthenticationStorage');

        return new AuthenticationService($storage, $authAdapter);
    }
}
```

Creamos también un Factory para el storage y así podemos personalizarlo con valores de configuración:

```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.
 */
```

```php
namespace User\Service\Factory;


use User\Service\AuthenticationStorageService;
use Zend\ServiceManager\FactoryInterface;
use Zend\ServiceManager\ServiceLocatorInterface;

class AuthenticationStorageServiceFactory implements FactoryInterface
{

    /**
     * Create service
     *
     * @param ServiceLocatorInterface $serviceLocator
     * @return mixed
     */
    public function createService(ServiceLocatorInterface $serviceLocator)
    {
        $config = $serviceLocator->get('config');
        $appName = $config['application']['name'];

        return new AuthenticationStorageService($appName);
    }
}
```

2. LoginController

Una vez tenemos creado el servicio AuthenticationService ya podemos usarlo en un controller. Nuestro controller se llamará User\Controller\LoginController:

```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.
 */
```

```php
namespace User\Controller;


use User\Form\InputFilter\LoginFormInputFilter;
use User\Form\LoginForm;
use Zend\Authentication\AuthenticationService;
use Zend\Mvc\Controller\AbstractActionController;
use Zend\Authentication\Adapter\DbTable\CredentialTreatmentAdapter;
use Zend\View\Model\ViewModel;
use User\Service\AuthenticationStorageService;

class LoginController extends AbstractActionController
{
    /**
     * @var AuthenticationService
     */
    private $authenticationService;

    /**
     * @var CredentialTreatmentAdapter
     */
    private $adapter;

    /**
     * @var AuthenticationStorageService
     */
    private $storage;

    /**
     * @param AuthenticationService $authenticationService
     */
    function __construct(AuthenticationService $authenticationService)
    {
        $this->authenticationService    = $authenticationService;
        $this->adapter                  = $authenticationService->getAdapter();
        $this->storage                  = $authenticationService->getStorage();
    }

    public function loginAction()
    {
        if ($this->identity()) {
            return $this->redirect()->toRoute('user\users\index');
        }

        $this->layout()->title = 'Login';

        $form = new LoginForm();
        $form->get('submit')->setValue('Login');
        $form->setAttribute('action', $this->url()->fromRoute('user\login\doLogin'));
```

```
        return [
            'form'      => $form,
            'messages'  => $this->flashMessenger()->getMessages(),
        ];

    }

    public function doLoginAction()
    {
        $request = $this->getRequest();

        if ($request->isPost()) {
            $form = new LoginForm();
            $inputFilter = new LoginFormInputFilter();
            $form->setInputFilter($inputFilter->getInputFilter());
            $form->setData($request->getPost());
            $messages = '';

            if ($form->isValid()) {
                $data = $form->getData();

                $this->adapter
                    ->setIdentity($data['email'])
                    ->setCredential($data['password'])
                ;

                $result = $this->authenticationService->authenticate();
                $messages = $result->getMessages();

                if ($result->isValid()) {
                    if ($data['rememberme'] == 1 ) {
                        $this->storage->setRememberMe(1);
                    }

                    $user = $this->adapter->getResultRowObject();
                    $this->storage->write($user);

                    return $this->redirect()->toRoute('user\users\index'); // success
                }
            }

            $form->prepare();

            $this->layout()->title = 'Login - Error - Review your data';

            $view = new ViewModel([
                'form'      => $form,
                'messages'  => $messages,
            ]);
            $view->setTemplate('user/login/login.phtml');
```

```
            return $view;
        }

        // trying to access 'user\login\doLogin' directly
        $this->flashMessenger()->addMessage('You must use this form');

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

    public function logoutAction()
    {
        $this->storage->forgetMe();
        $this->authenticationService->clearIdentity();
        $this->flashMessenger()->clearCurrentMessages();
        $this->flashMessenger()->addMessage('Logged Out');

        return $this->redirect()->toRoute('user\login\login');
    }
}
```

Como siempre la dependencia será inyectada en el constructor a través de un Factory. Extraemos las propiedades del objeto AuthenticationService para poder trabajar más comodamente.

Utilizamos el action 'loginAction' para mostrar el formulario de login. Preguntamos primero si ya está logueado haciendo uso del plugin identity(). Para poder hacer uso de este helper tanto en controllers como en vistas es necesario que creemos un servicio con el nombre

'Zend\Authentication\AuthenticationService'

nosotros lo haremos en la sección aliases puesto que hemos creado nuestro propio servicio:

```
'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',
    ),
),
```

De manera que nosotros haremos referencia al servicio como 'User\Service\Authentication' pero ZF utilizará la key de aliases.

En el método doLoginAction lo que hacemos es:

- compropar que la petición es por post
- validar el formulario
- validar las credenciales introducidas
- en caso de ser válidas, almacenamos la identidad en el storage

En el método logout borramos la identidad almacenada.

3. LoginForm

El formulario de login:

```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\Form;


use Zend\Form\Form;

class LoginForm extends Form
{
    function __construct($name = null)
    {
        parent::__construct();

        $this->setName('User');
        $this->setAttribute('method', 'post');

        $this->add(array(
            'name' => 'email',
```

```
            'type' => 'Email',
            'attributes' => array(
                'required' => 'required',
            ),
        ));

        $this->add(array(
            'name' => 'password',
            'type' => 'Password',
            'attributes' => array(
                'required' => 'required',
            ),
        ));

        $this->add(array(
            'name' => 'rememberme',
            'type' => 'Checkbox',
        ));

        $this->add(array(
            'name' => 'submit',
            'type' => 'Submit',
            'attributes' => array(
                'id' => 'submitbutton',
            ),
        ));
    }
}
```

3.1. LoginFormInputFilter

El InputFilter utilizado:

```
<?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.
 */
```

```php
namespace User\Form\InputFilter;


use Zend\InputFilter\InputFilter;
use Zend\InputFilter\InputFilterAwareInterface;
use Zend\InputFilter\InputFilterInterface;

class LoginFormInputFilter implements InputFilterAwareInterface
{

    /**
     * @var InputFilterInterface
     */
    private $inputFilter;

    /**
     * @param InputFilterInterface $inputFilter
     */
    function __construct()
    {
        $inputFilter = new InputFilter();

        $inputFilter->add(array(
            'name' => 'email',
            'required' => true,
            'filters' => array(
                array('name' => 'StringTrim'), // clean blank spaces
                array('name' => 'StripTags'), // clean malicious code
                array('name' => 'StringToLower'),
            ),
            'validators' => array(
                array(
                    'name' => 'EmailAddress',
                    'options' => array(
                        'messages' => array(
                            'emailAddressInvalidFormat' => 'You entered an invalid email
address',
                        ),
                    ),
                ),
                array(
                    'name' => 'NotEmpty',
                    'options' => array(
                        'messages' => array(
                            'isEmpty' => 'Email address is required',
                        ),
                    ),
                ),
            ),
        ));
```

```
        $inputFilter->add(array(
            'name' => 'password',
            'required' => true,
            'filters' => array(
                array('name' => 'Alnum'),
            ),
        ));

        $this->inputFilter = $inputFilter;
    }


    /**
     * Set input filter
     *
     * @param  InputFilterInterface $inputFilter
     * @return InputFilterAwareInterface
     */
    public function setInputFilter(InputFilterInterface $inputFilter)
    {
        throw new \Exception('Not used');
    }

    /**
     * Retrieve input filter
     *
     * @return InputFilterInterface
     */
    public function getInputFilter()
    {
        return $this->inputFilter;
    }
}
```

4. La vista

La vista login.phtml:

```
<div>
    <?php foreach ($messages as $m): ?>
    <p><?php echo $m ?></p>
    <?php endforeach; ?>
</div>

<?php echo $this->form()->openTag($form); ?>
    <table>
        <tr>
```

```
        <td colspan="2"><?php echo $this->formElementErrors($form->get('email'));
?></td>
      </tr>
      <tr>
        <td><strong>Email:</strong></td>
        <td><?php echo $this->formElement($form->get('email')) ?></td>
      </tr>
      <tr>
        <td colspan="2"><?php echo $this->formElementErrors($form-
>get('password')); ?></td>
      </tr>
      <tr>
        <td><strong>Password:</strong></td>
        <td><?php echo $this->formElement($form->get('password')) ?></td>
      </tr>
      <tr>
        <td><strong>Remember Me?:</strong></td>
        <td><?php echo $this->formElement($form->get('rememberme')) ?></td>
      </tr>
      <tr align="right">
        <td colspan="2"><?php echo $this->formElement($form->get('submit'))
?></td>
      </tr>
  </table>
<?php echo $this->form()->closeTag(); ?>
```