

Como usar Zend_Auth del Zend Framework

Elaborado por Rob Allen, www.akrobat.com

Revision 1.0.4

Copyright © 2007

Traducido por Claudio Cossio, www.ajaxcode.net

El propósito de este tutorial es dar una introducción básica del componente Zend_Auth en conjunto con el Zend Framework. Da continuidad al tutorial previo de "Desarrollo de Aplicaciones con Zend Framework" que se encuentra en <http://akrobat.com/zend-framework-tutorial> para la versión en inglés y para la versión en castellano ir al siguiente enlace http://ajaxcode.net/descargas/Desarrollo_Aplicaciones_ZendFramework.pdf

NOTA: El código ha sido probado con las versiones 0.9 y 0.9.1 del Zend Framework. Puede existir alguna posibilidad de que funcione con versiones posteriores, pero no funcionara en versiones anteriores de la 0.9.

Autenticación

Para el propósito de este tutorial, autenticación es el proceso de conceder el acceso a una persona a nuestra aplicación Web. Vamos a modificar la aplicación del listado de CD's creada en el tutorial "Desarrollo de Aplicaciones con Zend Framework" para que solicite una credencial de acceso para cualquier parte de la aplicación.

A grosso modo, vamos a necesitar las siguientes cosas:

- Crear una tabla de "users" (registrar unos usuarios).
- Crear una forma para acceder a la aplicación.
- Crear un `controller` que almacenara las acciones (`actions`) para el acceder (logging in) y salir del sistema (logging out).
- Alterar el pie de página para facilitar la salida del sistema (log out).
- Asegurarnos que el usuario ha dado los datos correctos antes de acceder a la aplicación.

La tabla de Usuarios.

Lo primero que necesitamos será una tabla de usuarios. No necesita ser compleja, así que la estructura será la siguiente:

<i>Fieldname</i>	<i>Type</i>	<i>Null?</i>	<i>Notes</i>
id	Integer	No	Primary key, Autoincrement
username	Varchar(50)	No	Unique key
password	Varchar(50)	No	
real_name	Varchar(100)	No	

Si utilizamos MySQL, la sentencia SQL para crear la tabla será la siguiente:

```
CREATE TABLE users (  
  id int(11) NOT NULL auto_increment,  
  username varchar(50) NOT NULL,  
  password varchar(50) NOT NULL,  
  real_name varchar(100) NOT NULL,  
  PRIMARY KEY (id),  
  UNIQUE KEY username (username)  
)
```

También necesitamos dar de alta un usuario para acceder a la aplicación:

```
INSERT INTO users (id, username, password, real_name)
VALUES (1, 'rob', 'rob', 'Rob Allen');
```

Utiliza esta sentencia en un cliente MySQL como phpAdmin o mediante la línea de comandos de MySQL (es obvio de que deberán usar un mejor username y password).

Cambios a la Configuración de Arranque (Bootstrap)

Para dar un seguimiento del proceso de autenticación del usuario, vamos a utilizar una sesión. Zend Framework nos proporciona `Zend_Session_Namespace` que nos ofrece una interfase orientada a objetos para sesiones.

Los cambios a `index.php` son:

zf-tutorial/index.php:

```
...
Zend_Loader::loadClass('Zend_Db_Table');
Zend_Loader::loadClass('Zend_Debug');
Zend_Loader::loadClass('Zend_Auth');
```

```
// load configuration
```

También aquí.

```
..
// setup database
$dbAdapter = Zend_Db::factory($config->db->adapter,
    $config->db->config->asArray());
Zend_Db_Table::setDefaultAdapter($dbAdapter);
Zend_Registry::set('dbAdapter', $dbAdapter);

// setup controller
$frontController = Zend_Controller_Front::getInstance();
...
```

Todo lo que tenemos que hacer aquí es asegurarnos de cargar la clase `Zend_Auth` y registrar la base de datos dentro del adaptador `registry`. Lo almacenamos en `registry` por que lo necesitaremos cuando realicemos la autenticación después.

El controlador Auth

Necesitamos un controlador para almacenar el “login” y el “logout”. Se nos hace razonable llamarlo `AuthController`.

Vamos a empezar de lo básico a partir de `IndexController`:

zf-tutorial/application/controllers/AuthController.php:

```
<?php
class AuthController extends Zend_Controller_Action
{
    function init()
    {
        $this->initView();
        $this->view->baseUrl = $this->_request->getBaseUrl();
    }

    function indexAction()
    {
        $this->_redirect('/');
    }
}
```

Hemos configurado `init()` para que el `view` sea inicializado y le podamos asignar el `baseUrl`. También creamos la función `indexAction()` ya que es requerida por `Zend_Controller_Action`. No vamos a necesitar `indexAction()` ya que haremos uso de `loginAction()` y `logoutAction()`, así que redirigimos a la ruta por defecto si alguien trata de navegar al directorio `auth/index`.

Acceso al Sistema (Login)

Para acceder a nuestra aplicación vamos a necesitar una forma, así que la acción de acceso va a trabajar de igual manera que otras formas dentro de `IndexController`. El template de la forma estará ubicado en `views/scripts/auth/login.phtml` y el código estará en `AuthController::loginAction()`. La forma es sencilla, ya que solo necesitara de dos campos: el nombre del usuario (`username`) y la contraseña (`password`):

zf-tutorial/application/views/scripts/auth/login.phtml:

```
<?php echo $this->render('header.phtml'); ?>
<h1><?php echo $this->escape($this->title); ?></h1>

<?php if(!empty($this->message)) :?>
<div id="message">
<?php echo $this->escape($this->message);?>
</div>
<?php endif; ?>

<form action="<?php echo $this->baseUrl ?>/auth/login" method="post">
<div>
    <label for="username">Username</label>
    <input type="text" name="username" value=""/>
</div>
<div>
    <label for="password">Password</label>
    <input type="password" name="password" value=""/>
</div>
<div id="formbutton">
<input type="submit" name="login" value="Login" />
</div>
</form>

<?php echo $this->render('footer.phtml'); ?>
```

El template interpreta los archivos `header.phtml` y `footer.phtml` en la cabecera y el pie de la página. Hacemos hincapié en que solamente se mostrara el mensaje si `$this->message` no contiene información. Esto es para informarle al usuario que no pudo acceder al sistema. El resto del contenido es el template de la forma.

Ahora que disponemos de una forma, vamos a crear un controlador de acción (`action controller`). Que lo codificamos dentro de `AuthController.php`:

zf-tutorial/application/controllers/AuthController.php:

```
class AuthController extends Zend_Controller_Action
{
    ...
    function loginAction()
    {
        $this->view->message = '';
        $this->view->title = "Log in";
        $this->render();
    }
}
```

Inicialmente todo lo que debemos hacer es inicializar el título y el mensaje, posteriormente se interpretara la forma. Si vamos a la dirección `http://zf-tutorial/auth/login` debe mostrar la forma para acceder al sistema.

¿Como vamos a procesar la forma una vez enviada la información? Para hacer eso, usamos el mismo método que las formas de *add* y *edit* dentro de IndexController y realizamos el proceso si el método de petición es POST. Vamos a modificar la función loginAction() que acabamos de crear:

zf-tutorial/application/controllers/AuthController.php:

```
class AuthController extends Zend_Controller_Action
{
    ...
    function loginAction()
    {
        $this->view->message = '';
        if ($this->request->isPost()) {
            // collect the data from the user
            Zend_Loader::loadClass('Zend_Filter_StripTags');
            $f = new Zend_Filter_StripTags();
            $username = $f->filter($this->request->getPost('username'));
            $password = $f->filter($this->request->getPost('password'));

            if (empty($username)) {
                $this->view->message = 'Please provide a username.';
            } else {
                // setup Zend_Auth adapter for a database table
                Zend_Loader::loadClass('Zend_Auth_Adapter_DbTable');
                $dbAdapter = Zend_Registry::get('dbAdapter');
                $authAdapter = new Zend_Auth_Adapter_DbTable($dbAdapter);
                $authAdapter->setTableName('users');
                $authAdapter->setIdentityColumn('username');
                $authAdapter->setCredentialColumn('password');

                // Set the input credential values to authenticate against
                $authAdapter->setIdentity($username);
                $authAdapter->setCredential($password);

                // do the authentication
                $auth = Zend_Auth::getInstance();
                $result = $auth->authenticate($authAdapter);
                if ($result->isValid()) {
                    // success: store database row to auth's storage
                    // system. (Not the password though!)
                    $data = $authAdapter->getResultRowObject(null,
                        'password');
                    $auth->getStorage()->write($data);
                    $this->redirect('/');
                } else {
                    // failure: clear database row from session
                    $this->view->message = 'Login failed.';
                }
            }
        }
        $this->view->title = "Log in";
        $this->render();
    }
}
```

Dentro del código hay mucho que comentar, así que empecemos:

```
// collect the data from the user
Zend_Loader::loadClass('Zend_Filter_StripTags');
$f = new Zend_Filter_StripTags();
$username = $f->filter($this->request->getPost('username'));
$password = $f->filter($this->request->getPost('password'));

if (empty($username)) {
    $this->view->message = 'Please provide a username.';
} else {
```

Como es usual se aplica un filtro para extraer los campos de "username" y "password" de los datos enviados por POST. Hacemos nota, que de la función `getPost()` de la petición ya que gestionara la revisión `isset()` y nos devolverá una cadena (string) vacía si no encuentra el campo dentro del arreglo (array) de POST. Si el "username" esta vacío, no tiene caso realizar la autenticación, inclusive `Zend_Auth` respondería con una excepción. Por dicha razón se verifica que el "username" este vacío e informar al usuario.

```
// setup Zend_Auth adapter for a database table
Zend_Loader::loadClass('Zend_Auth_Adapter_DbTable');
$dbAdapter = Zend_Registry::get('dbAdapter');
$authAdapter = new Zend_Auth_Adapter_DbTable($dbAdapter);
$authAdapter->setTableName('users');
$authAdapter->setIdentityColumn('username');
$authAdapter->setCredentialColumn('password');
```

`Zend_Auth` hace uso de un adaptador de sistema que le permite usar cualquier tipo de mecanismos de autenticación. Como vamos a utilizar una tabla de la base de datos, lo haremos con `Zend_Auth_Adapter_DbTable`. Para inicializar el adaptador, le asignas los campos a utilizar y le envías una conexión válida para la base de datos.

```
// Set the input credential values to authenticate against
$authAdapter->setIdentity($username);
$authAdapter->setCredential($password);
```

Debemos de indicarle al adaptador exactamente que "username" y "password" han sido capturados en la forma por el usuario.

```
// do the authentication
$auth = Zend_Auth::getInstance();
$result = $auth->authenticate($authAdapter);
```

Para realizar la autenticación, llamamos a la función `authenticate()` de `Zend_Auth`. Esto nos asegura que el resultado de la autenticación está almacenado automáticamente en la sesión.

```
if ($result->isValid()) {
    // success : store database row to auth's storage
    // system. (not the password though!)
    $data = $authAdapter->getResultRowObject(null,
        'password');
    $auth->getStorage()->write($data);
    $this->_redirect('/');
}
```

En el caso exitoso de la autenticación, almacenamos el registro completo (excepto la contraseña) de la base de datos dentro de `Zend_Auth`. Para asegurarnos de que se puede mostrar el nombre del usuario en el pie de la página Web (footer.phtml).

```
} else {
    // failure: clear database row from session
    $this->view->message = 'Login failed.';
}
}
```

En el caso de un fallo en la autenticación, se muestra un mensaje en la página para informar al usuario del acontecimiento.

El proceso de acceder a la aplicación se ha completado.

Salir del Sistema (Logout)

Para salirse del sistema es más sencillo que acceder a él, ya que solamente se requiere de borrar la información almacenada en `Zend_Auth`. Esto se lleva a cabo en la acción (`action`) `logoutAction()` dentro del controlador (`controller`) `AuthController` para que podamos dirigirnos a la dirección `http://zftutorial/auth/logout` para que el usuario salga del sistema:

zf-tutorial/application/controllers/AuthController.php:

```
class AuthController extends Zend_Controller_Action
{
    ...
    function logoutAction()
    {
        Zend_Auth::getInstance()->clearIdentity();
        $this->_redirect('/');
    }
}
```

La función `logoutAction()` es tan sencilla, que no necesita mucha explicación.

Necesitamos ofrecerle al usuario un enlace para que pueda salir de la aplicación cuando desee. Esto será más sencillo si lo colocamos en el pie de la página. Vamos a informarle al usuario de su nombre, para que tenga la confianza que esta dentro de la aplicación. Su nombre esta almacenado dentro del campo `real_name` de la tabla “users”, a la cual ya tenemos acceso por medio de la instancia `Zend_Auth`. Lo primero que debemos hacer es que `view` reciba la instancia, lo cual se realiza por medio de la función `init()` dentro del `IndexController()`:

zf-tutorial/application/controllers/IndexController.php:

```
class IndexController extends Zend_Controller_Action
{
    function init()
    {
        $this->initView();
        Zend_Loader::loadClass('Album');
        $this->view->baseUrl = $this->_request->getBaseUrl();
        $this->view->user = Zend_Auth::getInstance()->getIdentity();
    }
    ...
}
```

Es muy conveniente que el `Zend_Auth` es un componente individual, sino fuera así tendríamos que almacenarlo en el `registry`.

Ahora tenemos que agregar HTML al template `footer.phtml`:

zf-tutorial/application/views/footer.phtml:

```
<?php if($this->user) : ?>
<p id="logged-in">Logged in as <?php
    echo $this->escape($this->user->real_name); ?>.
<a href="<?php echo $this->baseUrl ?>/auth/logout">Logout</a></p>
<?php endif; ?>
</div>
</body>
</html>
```

El código HTML ya debería parecerle familiar. Usamos `escape()` para asegurar que el campo del nombre del usuario se despliegue correctamente y usamos `baseUrl` para establecer el sitio correcto del lugar donde se mostrara.

Esto es todo lo que se requiere para salirse del sistema.

Protección de las Acciones (Actions)

Lo que falta por hacer es asegurarnos de que ninguna de las acciones (`actions`) sea accesible si no se ha registrado el usuario. Para hacer eso, necesitamos agregar más código a la función de `preDispatch()` dentro del `IndexController`.

zf-tutorial/application/controllers/IndexController.php:

```
class IndexController extends Zend_Controller_Action
{
    ...
    function preDispatch()
    {
        $auth = Zend_Auth::getInstance();
        if (!$auth->hasIdentity()) {
            $this->_redirect('auth/login');
        }
    }
    ...
}
```

La función `preDispatch()` es llamada antes de cualquier acción (`action`) dentro del `controller`. Recibimos la instancia `Zend_Auth` y después su función `hasIdentity()` nos señalara si el usuario puede acceder al sistema. Si no cuenta con el acceso lo dirigimos al `action auth/login`.

¡De esta manera ya terminamos nuestra aplicación!

Conclusión

Así concluimos nuestra breve introducción de como integrar `Zend_Auth` dentro de una aplicación MVC. Esta muy claro que hay mas cosas que se pueden hacer con `Zend_Auth`, especialmente si se tienen varios controladores que se quieren proteger. Hacemos hincapié en que no hemos visto a detalle el método de autorización aquí, hace falta ver lo que nos puede ofrecer el componente `Zend_Acl`. `Zend_Acl` puede ser utilizado en conjunto con `Zend_Auth` para proveer diferentes niveles de acceso a los `actions` y a la información, pero este es un tema para otro tutorial.

Espero que hayan encontrado el tutorial interesante e informativo. Si encuentran algún error, por favor hacerlo de mi conocimiento escribiendo a rob@akrabat.com.

Nota: El idioma del autor es ingles, si tienen alguna duda o comentario que quieran hacer de la traducción o del tutorial en castellano por favor manden un correo a ccossio@ajaxcode.net.