

Débutez avec Zend_Auth

Traduction du tutoriel de Rob Allen :  **Getting started with Zend_Auth.**

par Rob Allen

Date de publication : 27/05/2007


Dernière mise à jour :

Ce tutoriel a pour but de nous donner les bases du composant Zend_Auth fourni avec le Zend Framework. Il est nécessaire d'avoir réalisé le précédent tutoriel "Débutez avec le Zend Framework", disponible en français sur **[cette page](#)**.

- I - Introduction
 - I-A - Important
 - I-B - Avant de commencer
 - I-C - Authentification
- II - Mise en place
 - II-A - La table 'users'
 - II-B - Le contrôleur Auth
- III - Méthodes
 - III-A - Se connecter
 - III-B - Se déconnecter
 - III-C - Protéger les actions
- IV - Conclusion
 - IV-A - Epilogue
 - IV-B - Liens

I - Introduction

I-A - Important

 *Attention : ce tutoriel a été testé avec les versions 0.9, 0.9.1 et 0.9.2 du Zend Framework. Il y a de grandes chances qu'il fonctionne avec des versions plus récentes, mais aucune avec des versions inférieurs à la 0.9.*

I-B - Avant de commencer

L'implémentation de l'authentification faite dans ce tutoriel utilise les sessions PHP. Vérifiez avant que le répertoire donné dans votre fichier php.ini sur la ligne session.save_path possède les droits d'écriture pour votre serveur web.

I-C - Authentification

Le but de ce tutoriel est de permettre aux visiteurs de se connecter à votre application web. Nous allons modifier l'application listant les albums, créée avec le précédent tutoriel (**Débutez avec le Zend Framework**), pour forcer l'identification des visiteurs afin de rendre accessible votre application.

Voici les différentes étapes en résumé :

- Créer une table 'users' dans la base de données (et insérer un utilisateur).
- Créer un formulaire login.
- Créer un contrôleur contenant les actions pour se connecter et se déconnecter.
- Modifier le pied de page afin de permettre la fermeture de session par l'utilisateur.
- Vérifier que l'utilisateur est connecté avant d'autoriser l'accès à votre application.

II - Mise en place

II-A - La table 'users'

La première chose dont nous avons besoin est une table 'users'. Ce n'est pas si compliqué, voici le schéma de la table users :

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

La requête SQL pour créer cette table est la suivante :

```
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)  
);
```

On a aussi besoin d'un utilisateur qui puisse se connecter :

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

Exécutez ces requêtes dans votre client MySQL, comme par exemple phpMyAdmin ou la console MySQL directement (évidemment, vous pouvez changer les valeurs de username et password).

zf-tutorial/index.php

```
...  
Zend_Loader::loadClass('Zend_Db_Table');  
Zend_Loader::loadClass('Zend_Debug');  
Zend_Loader::loadClass('Zend_Auth');  
// load configuration  
...  
and  
...  
// setup database  
$dbAdapter = Zend_Db::factory($config->db->adapter,  
$config->db->config->toArray());  
Zend_Db_Table::setDefaultAdapter($db);  
Zend_Registry::set('dbAdapter', $db);  
// setup controller  
$frontController = Zend_Controller_Front::getInstance();  
...  
...
```

Tout ce que nous avons à faire ici est de s'assurer que nous avons chargé la classe Zend_Auth et intégré l'adaptateur dbAdapter dans le registre. Nous le stockons dans le registre puisque nous en aurons besoin plus tard pour l'authentification.

II-B - Le contrôleur Auth

Nous avons besoin d'un contrôleur pour mettre en place les actions login et logout. En toute logique, appelons-le AuthController.

Nous commencerons avec un IndexController simple :

```
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('/');
    }
}
```

Nous créons d'abord la méthode `init()` pour que la vue soit initialisée et que `baseUrl` y soit assignée. Nous créons aussi la méthode `indexAction()` comme le nécessite le `Zend_Controller_Action`. Nous n'avons pas besoin de la méthode `indexAction()` puisque nous utiliserons les méthodes `loginAction()` et `logoutAction()`, donc nous redirigerons tous ceux qui tentent d'accéder par la barre d'adresse à `auth/index` sans s'être identifiés.

III - Méthodes

III-A - Se connecter

Pour nous connecter à notre application, nous avons besoin d'un formulaire. L'action login() est très similaire aux autres formulaires du contrôleur IndexController, précédemment vu dans le tutoriel "**Débutez avec le Zend Framework**". Le gabarit(=un **template**) du formulaire sera dans le fichier views/scripts/auth/login.phtml et le code dans AuthController::loginAction(). Le formulaire est très simple, nécessitant juste deux champs texte : username et 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'); ?>
```

Le gabarit affiche le contenu des fichiers header.phtml et footer.phtml en haut et en bas de la fenêtre respectivement. Remarquez que nous affichons un message uniquement si `$this->message` n'est pas vide. Cela indique à l'utilisateur si l'identification a échoué. Le reste du gabarit est le formulaire pour se connecter.

Maintenant que nous avons un formulaire, nous devons créer l'action du contrôleur qui l'exécutera. Ajoutons alors au fichier 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();
    }
}
```

Pour commencer, nous renseignons uniquement les valeurs title et message pour l'affichage du formulaire. Vous pouvez voir le résultat sur <http://localhost/zf-tutorial/auth/login>.

Comment procède-t-on après que le formulaire soit envoyé ? Pour réaliser l'authentification, nous utiliserons le même formulaire que les méthodes ajouter() et supprimer() du contrôleur IndexController et lancer l'identification si le formulaire envoyé utilise la méthode post. Changeons donc la méthode loginAction() que nous venons juste de créer :

```
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();
    }
}
```

Baucoup de choses se sont passées ici, revenons dessus en détail :

```
// 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 {
```

Nous créons un nouveau filtre et nous extrayons les valeurs username et password des données du formulaire. Remarquez que nous utilisons la fonction `getPost()` qui intègre la fonction `isset()` afin de vérifier si le champ renseigné existe, et nous retourne une chaîne vide si le champ n'existe pas dans le tableau POST. Si le champ username est vide, il n'y a donc aucune chance qu'on puisse l'identifier (et `Zend_Auth` nous retourne une exception si nous essayons !), alors nous informons le visiteur que le champ username est vide.

```
// 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` utilise un système d'adaptateurs qui vous autorise à utiliser n'importe quel nombre de méthodes pour réaliser l'authentification. Nous voulons utiliser la table de la base de données et pour cela nous appelons l'adaptateur `Zend_Auth_Adapter_DbTable`. Pour mettre en place un adaptateur, vous devez renseigner les champs pour obtenir une connection correcte à la base de données.

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

Nous devons dire à l'adaptateur exactement les valeurs des champs username et password que nous avons récupéré dans le formulaire.

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

Pour l'authentification elle-même, nous exécutons la méthode `authenticate()` de `Zend_Auth`. Cela assure que les résultats de l'authentification soient automatiquement enregistrés dans la session.

```
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 cas de succès, nous stockons entièrement l'enregistrement de la table (excepté le mot de passe !) dans le singleton (cf. [dico design pattern](#)) `Zend_Auth`. Cela nous permet de collecter le nom de l'utilisateur pour l'afficher par exemple dans le pied de page.

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

Si l'authentification échoue, nous renseignons la valeur message de notre vue afin que l'utilisateur sache ce qui est arrivé.

Le processus d'authentification est maintenant terminé.

III-B - Se déconnecter

Se déconnecter est beaucoup plus simple que se connecter puisque nous avons juste à dire au singleton Zend_Auth de supprimer ses données. Ceci nous ramène à réaliser une nouvelle action logoutAction() dans le contrôleur AuthController, qui se lancera via l'url suivante <http://localhost/zf-tutorial/auth/logout>.

zf-tutorial/application/controllers/AuthController.php

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

La méthode logoutAction() parle d'elle-même, je ne pense pas avoir quelque chose à dire à son propos !

Maintenant, nous devons permettre à l'utilisateur de se déconnecter de l'application via un simple lien. C'est très facile, nous placerons donc ce lien dans le pied de page. Nous dirons aussi à l'utilisateur son nom, afin qu'il sache qu'il est bien connecté. Son nom est stocké dans le champ real_name de la table utilisateur et il est disponible dans l'instance Zend_Auth. La première chose à faire est de l'avoir dans notre vue, donc nous insérerons notre ligne dans la méthode init() du contrôleur 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();
    }
    ...
}
```

C'est très pratique que le Zend_Auth soit un singleton, autrement nous aurions dû le stocker dans le registre !

Nous devons maintenant ajouter du code html dans le fichier 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>
```

Ce code HTML devrait vous paraître un peu familier puisqu'il n'y a rien de nouveau ici. Nous utilisons la méthode escape() afin de s'assurer que le nom réel de l'utilisateur s'affiche correctement, et utilisons aussi la valeur baseUrl pour que l'adresse href soit complète.

C'est tout ce dont nous avons besoin pour se déconnecter.

III-C - Protéger les actions

Nous devons maintenant nous assurer qu'aucune action soit accessible si vous n'êtes pas connecté. Pour y parvenir, nous devons ajouter plusieurs lignes de code dans la méthode `preDispatch()` du `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');
        }
    }
    ...
}
```


`preDispatch()` est appelé avant toutes les actions contenues dans le contrôleur. Nous récupérons l'instance `Zend_Auth` et ensuite sa méthode `hasIdentity()` qui nous informe s'il y a un utilisateur loggué. S'il n'y en a aucun, nous redirigeons le visiteur vers la page `auth/login`.

Et c'est terminé !

IV - Conclusion




IV-A - Epilogue

Nous terminons donc notre bref coup d'oeil vers l'intégration du Zend_Auth dans une application MVC. Bien entendu, il y a des tas de choses que vous pouvez faire avec le Zend_Auth, et il y a beaucoup de chemins pour parvenir à faire fonctionner son code, spécifiquement si vous avez de multiples contrôleurs à protéger. Notez que nous n'avons pas détaillé ici les niveaux d'accès comme le permet le composant Zend_Acl. Zend_Acl pourrait être utilisé en combinaison avec le Zend_Auth pour autoriser différents niveaux d'accès à nos actions ou données, mais c'est le sujet d'un autre tutoriel.

J'espère que vous avez trouvé ce tutoriel intéressant et ludique. Si vous trouvez une quelconque erreur, veuillez m'envoyer un email à  rob@akrobat.com

IV-B - Liens

Ressources Developpez.com :

-  [Débutez avec le Zend Framework](#), par Rob Allen et Guillaume Rossolini.
-  [Présentation du Zend Framework](#), par Julien Pauli.
-  [Forum d'entraide au Zend Framework](#).

Ressources externes :

- Le tutoriel original :  [Getting started with Zend_Auth](#), par Rob Allen.
-  [Un site dédié au Zend Framework](#).

