Zend Framework! Programming
Shopping application

Zend_Auth và ứng dụng

Zend Auth là một lớp cung cấp cho chúng ta nhiều phương thức xử lý giúp chúng ta đơn giản hóa thực hiện quá trình chứng thực tài khoản có trong hệ thống ứng dụng. Zend Auth hỗ trợ cho chúng ta trong quá trình đăng nhập, thoát và lấy các thông tin cần thiết của tài khoản người sử dụng đang đăng nhập. Zend_Auth không là một lớp riêng lẻ mà nó kết hợp giữa các lớp khác nhau như Zend_Db_Table, Zend_Session để hình thành.

Giáo trình: Zend Framework! Programming
Chuyên đề: Shopping application

Biên soạn: Phạm Vũ Khánh
Email: vukhanh2212@gmail.com
Điện thoại: 0908.893326
Website: www.zend.vn
Cập nhật: 07-2010

Để xử dụng tốt Zend Auth đầu tiên chúng ta cần nắm rõ những phương thức cơ bản được xử dụng trong quá trình chứng thực này

1. Khởi tạo database

Zend Auth không sử dụng Model truy cập vào Database để so sánh dữ liệu trong bảng chứa tài khoản đăng nhập mà nó sử dụng qua một lớp riêng đó là Zend_Auth_Adapter_DbTable vì vậy chúng ta cần cung cấp chi Zend Auth một cấu hình kết nối database.

Tạo đăng ký database tại Bootstrap của ứng dụng:

```
protected function _initDb()
{
	$dbOption = $this->getOption('resources');
	$dbOption = $dbOption['db'];

	// Setup database
	$db = Zend_Db::factory($dbOption['adapter'],$dbOption['params']);

	$db->setFetchMode(Zend_Db::FETCH_ASSOC);
	$db->query("SET NAMES 'utf8'");
	$db->query("SET CHARACTER SET 'utf8'");

	Zend_Registry::set('connectDB',$db);

	//Khi thiet lap che do nay model moi co the su dung duoc
	Zend_Db_Table::setDefaultAdapter($db);

	// Return it, so that it can be stored by the bootstrap
	return $db;
}
```

2. Khởi tạo Zend Auth

```
$auth = Zend_Auth::getInstance ();
```

3. Thiết lập kết nối với bảng chứa tài khoản

```
$authAdapter = new Zend_Auth_Adapter_DbTable ( $db );
$authAdapter->setTableName($tableName)
                  ->setIdentityColumn($identityColumn)
                  ->setCredentialColumn($credentialColumn);
```

$tableName : tên bảng chứa tài khoản
$identityColumn: tên cột chứa tài khoản trong bảng $tableName
$credentialColumn: tên cột chứa mật khẩu của tài khoản trong bảng $tableName

4. Đưa dữ liệu từ FORM đăng nhập vào so sánh:

```
$authAdapter->setIdentity ($value);
$authAdapter->setCredential ($credential);
```

$value: dữ liệu của tài khoản gửi từ FORM đăng nhập

$credential: mật khẩu của tài khoản được gửi từ FORM đăng nhập

5. Kiểm tra điều kiện phụ

```
$select = $authAdapter->getDbSelect();
$select->where($cond);
```

$cond: điều kiện phụ để có thể login vào tài khoản. Ví dụ như tài khoản đã được kích hoạt.

6. Lấy kết quả truy vấn trong Database

```
$result = $auth->authenticate ( $authAdapter );
```

7. Kiểm tra kết quả truy vấn

```
$result->isValid ()
```

8. Lấy những dữ liệu cần thiết trong bảng chứa tài khoản khi chứng thực thành công

```
$authAdapter->getResultRowObject($returnColumns, $omitColumns);
```

$returnColumns: Mảng chứa tên các cột cần lấy trong bảng tài khoản
$omitColumns: Mảng chứa tên các cột không cần lấy trong bảng tài khoản

9. Lưu dữ liệu đã lấy ra từ bảng chứa tài khoản vào session của Zend Auth

```
$auth->getStorage ()->write ( $data );
```

$data: Dữ liệu lấy ra từ bảng chứa tài khoản (mục 8)

10. Kiểm tra xem tài khoản đã được chứng thực hay chưa

```
$auth  = Zend_Auth::getInstance();
$auth->hasIdentity());
```

11. Lấy thông tin của tài khoản trong session sau khi đã chứng thực thành công

```
$auth = Zend_Auth::getInstance();
$infoUser = $auth->getIdentity();
```

12. Xóa chứng thực (logout)

```
$auth = Zend_Auth::getInstance();
$auth->clearIdentity();
```

13. Sử dụng hàm preDispatch() để thiết lập chế độ bảo vệ trang

```
function  preDispatch(){
       //Khối lệnh chặn user nếu chưa login vào tài khoản
}
```