BaconAuthentication Documentation

Release latest

Ben Scholzen ’DASPRiD’

December 28, 2016

Contents

BaconAuthentication is a general purpose authentication module for Zend Framework 2. It comes with a pluggable authentication service which allows to not only create simple username/password authentication, but also to easily integrate third-party authentication (like OpenID or OAuth), as well as two-factor authentication.

CHAPTER 1

Installation

1.1 With composer

1. Add BaconAuthentication to your composer.json. Either use a stable tag for this or dev-master:

```
"require": {
    "bacon/bacon-authentication": "dev-master"
}
```

2. Tell composer to download BaconAuthentication:

```
$ php composer.phar update
```

1.2 By cloning

Clone BaconAuthentication into your vendor folder and install all at least ZendStdlib and any possible dependencies (refer to the composer.json file for additionaly suggested dependencies).

CHAPTER 2

Theory of operation

BaconAuthentication comes with an authentication service interface (`BaconAuthentication\AuthenticationServiceInte` which defines two methods. The first one being `authenticate($request, $response)`, which tries to authenticate the current request. This method is used for both processing current input from the user (e.g. a login form), as well as retreiving the currently authenticated subject. The return value of this method will always be a `result object`.

The other method is `resetCredentials()`, which will simply remove all persisted information and thus make the authenticated subject anonymous again.

Authentication results

3.1 ResultInterface

Every call to the `authenticate()` method of the authentication service will return a result object. The returned result is defined by the *`ResultInterface`*:

interface `BaconAuthentication\Result\`**`ResultInterface`**
Generic result interface.

`BaconAuthentication\Result\ResultInterface::`**`isSuccess`**`()`
Returns whether the authentication was successful.

Returns bool

`BaconAuthentication\Result\ResultInterface::`**`isFailure`**`()`
Returns whether the authentication was a failure.

Returns bool

`BaconAuthentication\Result\ResultInterface::`**`isChallenge`**`()`
Returns whether the authentication generated a challenge.

Returns bool

`BaconAuthentication\Result\ResultInterface::`**`getPayload`**`()`
Returns the payload associated with the result.

For a successful result, the payload should be the identity of the subject. In the case of a failure, it should contain error information enclosed in an *`Error`* object. For a challenge, no payload is required.

Returns mixed|null

3.2 Result

BaconAuthentication provides a generic implementation of the *`ResultInterface`*, which should be sufficient for most use-cases. It defines the following additional methods:

class `BaconAuthentication\Result\`**`Result`**
Generic result implementation.

constant `BaconAuthentication\Result\Result::`**`STATE_SUCCESS`**
success

constant `BaconAuthentication\Result\Result::`**`STATE_FAILURE`**
failure

constant `BaconAuthentication\Result\Result::`**`STATE_CHALLENGE`**
challenge

`BaconAuthentication\Result\Result::`**`__construct`**`(`*$state*[, *$payload*]`)`

Parameters

- **`$state`** (*`string`*) –
- **`$payload`** (*`mixed|null`*) –

3.3 Error

The *`Error`* object which is returned as payload in the case of a failure is defined like this:

class `BaconAuthentication\Result\`**`Error`**

`BaconAuthentication\Result\Error::`**`__construct`**`(`*$scope*, *$message*`)`

Parameters

- **`$scope`** (*`string`*) –
- **`$message`** (*`string`*) –

`BaconAuthentication\Result\Error::`**`getScope`**`()`

Returns string

`BaconAuthentication\Result\Error::`**`getMessage`**`()`

Returns string

CHAPTER 4

Indices and tables

- BaconAuthentication User Guide
- search

PHP Namespace Index