comodojo/cookies documentation

Release 2.0.0

Marco Giovinazzi

Jul 01, 2018

Contents:

Minimalist and extensible library to manage cookies.

CHAPTER 1

Introduction

This library aims to provide an OO approach to defining cookies and to simplify cookies' management.
It includes also a cookie manager class that can handle multiple cookies at the same time.

1.1 Installation

First install composer, then:

```
composer require comodojo/cookies
```

1.1.1 Requirements

To work properly, comodojo/cookies requires PHP >=5.4.0.

1.2 General Usage

1.2.1 Creating a new cookie

A cookie can be defined creating a new instance of *Comodojo\Cookies\Cookie*, or one of the other *Cookie types*.
Once defined, additional methods can be used to set/get cookie properties, but cookie is just a server-side object.
Nothing is passed to the client until the *Cookie::save()* method is invoked.

```
1  <?php
2
3  use \Comodojo\Cookies\Cookie;
4
5  // create a new cookie instance
6  $cookie = new Cookie('my-cookie');
7
8  // set cookie's properties
9  $cookie->setValue( "Lorem ipsum dolor" )
```

(continues on next page)

(continued from previous page)

```
    ->setExpire( time()+3600 )
    ->setPath( "/myapp" )
    ->setDomain( "example.com" )
    ->setSecure()
    ->setHttponly();

// persist the cookie
$result = $cookie->save();
```

Alternatively, the static constructor *Cookie::create* is available to quikly create a cookie.

```
<?php

use \Comodojo\Cookies\Cookie;

// define a new cookie
$cookie = Cookie::create('my_cookie', array(
    'value' => "Lorem ipsum dolor"
    'expire' => time()+3600
    'path' => "/myapp"
    'domain' => "example.com"
    'secure' => true
    'httponly' => true
    )
);

// persist the cookie
$result = $cookie->save();
```

1.2.2 Loading a cookie

An existent cookie can be easily loaded using the *Cookie::load()* method.

```
<?php

use \Comodojo\Cookies\Cookie;

// create a new cookie instance
$cookie = new Cookie('my-cookie');

// load cookie
$cookie->load();

// read the cookie value
$value = $cookie->getValue();
```

The static constructor *Cookie::retrieve* can be used to speed up operation.

```
<?php

use \Comodojo\Cookies\Cookie;

// create a new cookie instance
$cookie = Cookie::retrieve('my-cookie');

// read the cookie value
$value = $cookie->getValue();
```

1.2.3 Cookie size

By default maximum allowed lenght for a cookie is 4000 bytes, to allow it to work in all major browsers.

Note: General informations about cookie limits is available in the rfc6265. See also this site for more informations.

Maximum cookie size can be overridden when creating a cookie.

```
<?php

use \Comodojo\Cookies\Cookie;

// create a new cookie instance with m-size of approximately 3k
$cookie = new Cookie('my-cookie', 3000);
```

Note: The real available size of cookie can <4KB due to serialization and (in case) encryption. In case of cookie > 4KB, a *\Comodojo\Exception\CookieException* is raised.

1.2.4 Content serialization

By default a cookie will serialize/unserialize it's content prior to be setted/loaded.

This behaviour can be overridden in *Cookie::setValue* and *Cookie::getValue* methods.

```
<?php

use \Comodojo\Cookies\Cookie;

// create a new cookie instance
$cookie = new Cookie('my-cookie');

// set a plain value (no serialization)
$cookie->setValue("Lorem ipsum dolor", false);
```

1.3 Cookie types

1.3.1 Plain cookie

Plain cookies are just an OO implementation of standard cookies. As an option, cookie content can be serialized on the client side.

```
<?php

use \Comodojo\Cookies\Cookie;

// create a new cookie instance with m-size of approximately 3k
$cookie = new Cookie('my-cookie');
```

1.3.2 Encrypted cookie

The class *\Comodojo\Cookies\EncryptedCookie* provides an extension of plain cookie in which cookie content is encrypted using a 256bit AES key, that should be provided to class contructor.

To setup a EncryptedCookie:

```php
<?php

use \Comodojo\Cookies\EncryptedCookie;

// create a new cookie instance with m-size of approximately 3k
$cookie = new EncryptedCookie('my-cookie', 'my-super-secret-key');
```

1.3.3 Secure cookie

The class *\Comodojo\Cookies\SecureCookie* provides an extension of Encrypted Cookie to ensure a minimum protection from cookie spoofing.

The crypto key is calculated using both user defined secret and *IP informations* from *$_SERVER* superglobal (if available).

This can be useful in internal networks or where clients does not often change IP address.

To setup a SecureCookie:

```php
<?php

use \Comodojo\Cookies\SecureCookie;

// create a new cookie instance with m-size of approximately 3k
$cookie = new SecureCookie('my-cookie', 'my-super-secret-key');
```

1.3.4 Extending

A custom cookie can be easily created implementing the *\Comodojo\Cookies\CookieInterface* interface.

Additionally, the abstract class *\Comodojo\Cookies\AbstractCookie* can be used to reuse common cookie methods.

1.4 Cookie Manager

Cookie manager is a class that can handle multiple cookies at the same time.

Each managed cookie object must implement the *\Comodojo\Cookies\CookieInterface*.

1.4.1 Saving multiple cookies

```php
<?php

use \Comodojo\Cookies\Cookie;
use \Comodojo\Cookies\CookieManager;

// create two different cookies
$first_cookie = new Cookie('first_cookie');
$second_cookie = new Cookie('second_cookie');

// init the manager
$manager = new CookieManager();

// add cookies to the manager
$manager
    ->add($first_cookie)
```

(continues on next page)

(continued from previous page)

```
    ->add($second_cookie);

// save them all
$result = $manager->save();
```

1.4.2 Loading multiple cookies

```
<?php

use \Comodojo\Cookies\Cookie;
use \Comodojo\Cookies\CookieManager;

// create two different cookies
$first_cookie = new Cookie('first_cookie');
$second_cookie = new Cookie('second_cookie');

// init the manager
$manager = new CookieManager();

// add cookies to the manager
$manager
    ->add($first_cookie)
    ->add($second_cookie);

// load cookies and retrieve contents
$result = $manager->load()->getValues();
```