Leach Documentation

Release 2.1.0-dev

Pierre Minnieur

February 04, 2015

Contents

CHAPTER 1

Installation

1.1 Phar file *(recommended)*

Download Leach Phar file:

```
wget http://leach.io/leach.phar
php leach.phar
```

1.2 Composer

Add the following entry to your `composer.json`:

```
{ "require": { "leach/leach": "dev-master" }}
```

Checkout detailed package information on Packagist.

1.3 PEAR package *(coming soon)*

Coming soon.

1.4 Clone from GitHub

Clone Leach git repository:

```
git clone git://github.com/pminnieur/Leach.git leach
```

Download Composer Phar file and install dependencies:

```
wget http://getcomposer.org/composer.phar
php composer.phar install --install-suggests
```

Run `leach` executable from `bin` directory:

```
php bin/leach
```

CHAPTER 2

Configuration

2.1 Command

2.1.1 start

```
php leach.phar start <container> [options]
```

send-spec

- **Type**: `string`
- **Default**: `tcp://127.0.0.1:9998`
- **Note**: equals Mongrel2 `recv_spec`

Example

```
<?php
$container = new Container(array(
    'send_spec' => 'tcp://127.0.0.1:9998',
));
```

```
php leach.phar start <container> --send-spec=tcp://127.0.0.1:9998
```

send-id

- **Type**: `string`
- **Default**: `296fef89-153f-4464-8f53-952b3a750b1b`

Example

```
php leach.phar start <container> --send-id=296fef89-153f-4464-8f53-952b3a750b1b
```

recv-spec

- **Type**: `string`
- **Default**: `tcp://127.0.0.1:9999`
- **Note**: equals Mongrel2 `send_spec`

Example

```
php leach.phar start <container> --recv-spec=tcp://127.0.0.1:9999
```

2.2 Container

2.2.1 Server

expose_leach

Adds a `X-Leach-Version` header to each `Response`.

- **Type**: `Boolean`
- **Default**: `false`

Example

```
<?php
return new Container(array(
    'expose_leach' => false,
));
```

max_requests

- **Type**: `integer`
- **Default**: `500`

Example

```
<?php
return new Container(array(
    'max_requests' => 500,
));
```

2.2.2 Transport

send_spec

- **Type**: `string`
- **Default**: `null`
- **Note**: equals Mongrel2 `recv_spec`

Example

```
<?php
return new Container(array(
    'send_spec' => 'tcp://127.0.0.1:9998',
));
```

send_id

- **Type**: `string`
- **Default**: `null`

Example

```
<?php
return new Container(array(
    'send_id' => '296fef89-153f-4464-8f53-952b3a750b1b',
));
```

recv_spec

- **Type**: `string`
- **Default**: `null`
- **Note**: equals Mongrel2 `send_spec`

Example

```
<?php
return new Container(array(
    'recv_spec' => 'tcp://127.0.0.1:9999',
));
```

CHAPTER 3

Extending

Extending Leach is easy via intelligent usage of event listeners. You can interact with Leach if you register your own listeners and provide an `EventDispatcherInterface` instance with your `ContainerInterface` instance. It is also possible to provide and make use of your own options.

3.1 Filter response

```php
<?php

require_once __DIR__.'/silex.phar';

use Silex\Application;
use Symfony\Component\EventDispatcher\EventDispatcher;
use Leach\Container\SilexContainer;
use Leach\Events;
use Leach\Event\FilterResponseEvent;

$app = new Application();

$app->get('/hello/{name}', function($name) use($app) {
    return 'Hello '.$app->escape($name);
});

$dispatcher = new EventDispatcher();
$dispatcher->addListener(Events::RESPONSE, function(FilterResponseEvent $event) {
    $event->getResponse()->headers->set('X-MyApp-Version', '1.2.3');
});

return new SilexContainer($app, array(), $dispatcher);
```

CHAPTER 4

Examples

4.1 Mongrel2

Configure and start Mongrel2:

```
cd examples && mkdir -p logs run
m2sh load -config leach.conf -db leach.db
m2sh start -host localhost -db leach.db
```

Example Mongrel2 configuration (as in `examples/leach.conf`):

```
leach = Handler(send_spec = 'tcp://127.0.0.1:9999',
                send_ident = 'b6c95667-4ede-4cf0-b2de-a54d826576c9',
                recv_spec = 'tcp://127.0.0.1:9998',
                recv_ident = '')

localhost = Host(name="localhost", routes={
    '/': leach
})

main = Server(
    uuid = "2dfc4c3b-1a6d-4965-a924-66ff081c3c29",
    access_log = "/logs/access.log",
    error_log = "/logs/error.log",
    chroot = "./",
    default_host = "localhost",
    name = "leach",
    pid_file = "/run/mongrel2.pid",
    port = 6767,
    hosts = [localhost]
)

servers = [main]
```

4.2 Silex

Download Silex Phar file:

```
wget http://silex.sensiolabs.org/get/silex.phar examples/silex/silex.phar
```

Start Leach:

```
php leach.phar start examples/silex/container.php --send-id=1e44c719-9d26-4992-8dd8-00142f650ea7
```

Example Silex container (as in `examples/silex/container.php`):

```
<?php

require_once __DIR__.'/silex.phar';

use Silex\Application;
use Leach\Container\SilexContainer;

$app = new Application();

$app->get('/hello/{name}', function($name) use($app) {
    return 'Hello '.$app->escape($name);
});

return new SilexContainer($app);
```

4.3 Symfony

Install "Symfony Standard Edition" distribution:

```
git clone git://github.com/symfony/symfony-standard.git examples/symfony/symfony-standard
cd examples/symfony/symfony-standard
php bin/vendors install
```

Start Leach:

```
php leach.phar start examples/symfony/container.php --send-id=0aa1d405-e5b5-4a0c-a222-3fc4e30e0e6d
```

Example Symfony container (as in `examples/symfony/container.php`):

```
<?php

require_once __DIR__.'/symfony-standard/app/bootstrap.php.cache';
require_once __DIR__.'/symfony-standard/app/AppKernel.php';
// require_once __DIR__.'/symfony-standard/app/AppCache.php';

use Leach\Container\SymfonyContainer;

$kernel = new AppKernel('prod', false);
// $kernel = new AppCache($kernel);
$kernel->loadClassCache();

return new SymfonyContainer($kernel);
```

CHAPTER 5

Events

Each event has a getter method to access the payload. To access a `$container` payload, simply use a camelCased method like `getContainer()`.

5.1 leach.setup

Setup a container. The Symfony container uses `leach.setup` events to `boot` kernels.

- **Event:** `Leach\Event\SetUpEvent`
- **Payload**
- `$container`: `Leach\Container\ContainerInterface`

5.2 leach.request

Filter a request.

- **Event:** `Leach\Event\FilterRequestEvent`
- **Payload**
- `$container`: `Leach\Container\ContainerInterface`
- `$request`: `Symfony\Component\HttpFoundation\Request`

5.3 leach.response

Filter a response. The server uses `leach.response` events to add a `X-Leach-Version` header to each response.

- **Event:** `Leach\Event\FilterResponseEvent`
- **Payload**
- `$container`: `Leach\Container\ContainerInterface`
- `$request`: `Symfony\Component\HttpFoundation\Request`
- `$response`: `Symfony\Component\HttpFoundation\Response`

5.4 leach.teardown

Tear down a container. The Symfony container uses `leach.teardown` events to `terminate` and `shutdown` kernels.

- **Event:** `Leach\Event\TearDownEvent`
- **Payload**
- `$container`: `Leach\Container\ContainerInterface`
- `$request`: `Symfony\Component\HttpFoundation\Request`
- `$response`: `Symfony\Component\HttpFoundation\Response`