

Building Websites with Zend Expressive 3

Rob Allen, Nineteen Feet

January 2018 ~ @akrabort

A microframework with full
stack components



μFramework core

- Router
- Container
- Template renderer
- Error handler
- Configuration

Ecosystem

- Filtering and validation
- API rendering: HAL & Problem-API
- Database abstraction
- Session handling
- Logging
- Mail
- Pagination
- Caching

Agnostic

Router:

FastRoute, Aura.Router or Zend Router

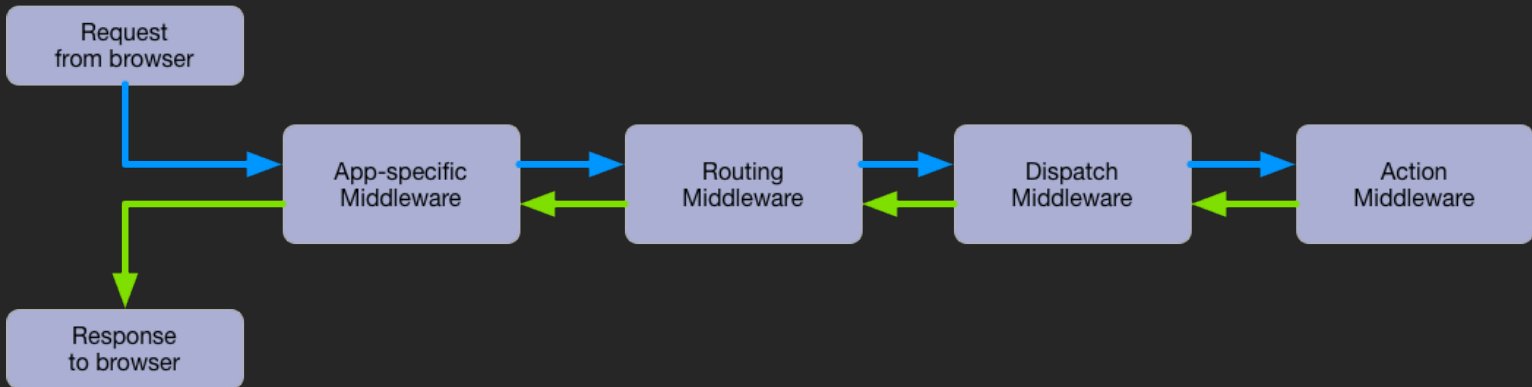
DI Container:

Aura.Di, Auryn, Pimple, Symfony DI Container or Zend-ServiceManager

Template:

Plates, Twig or Zend View

Middleware pipeline



PSR-15 MiddlewareInterface

```
namespace Interop\Http\ServerMiddleware;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface;
}
```

PSR-15 MiddlewareInterface

```
namespace Interop\Http\ServerMiddleware;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface;
}
```


PSR-15 MiddlewareInterface

```
namespace Interop\Http\ServerMiddleware;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface;
}
```

PSR-15 MiddlewareInterface

```
namespace Interop\Http\ServerMiddleware;

use Psr\Http\Message\ResponseInterface;
use Psr\Http\Message\ServerRequestInterface;

interface MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface;
}
```

Structure of middleware

```
class TimerMiddleware implements MiddlewareInterface
{
    public function process($request, $handler)
    {
        $start = microtime(true);

        $response = $handler->handle($request);

        $taken = microtime(true) - $start;
        $response->getBody()->write("<!-- Time: $taken -->");

        return $response;
    }
}
```

Structure of middleware

```
class TimerMiddleware implements MiddlewareInterface
{
    public function process($request, $handler)
    {
        $start = microtime(true);

        $response = $handler->handle($request);

        $taken = microtime(true) - $start;
        $response->getBody()->write("<!-- Time: $taken -->");

        return $response;
    }
}
```

Structure of middleware

```
class TimerMiddleware implements MiddlewareInterface
{
    public function process($request, $handler)
    {
        $start = microtime(true);

        $response = $handler->handle($request);

        $taken = microtime(true) - $start;
        $response->getBody()->write("<!-- Time: $taken -->");

        return $response;
    }
}
```

Structure of middleware

```
class TimerMiddleware implements MiddlewareInterface
{
    public function process($request, $handler)
    {
        $start = microtime(true);

        $response = $handler->handle($request);

        $taken = microtime(true) - $start;
        $response->getBody()->write("<!-- Time: $taken -->");

        return $response;
    }
}
```

Structure of middleware

```
class TimerMiddleware implements MiddlewareInterface
{
    public function process($request, $handler)
    {
        $start = microtime(true);

        $response = $handler->handle($request);

        $taken = microtime(true) - $start;
        $response->getBody()->write("<!-- Time: $taken -->");

        return $response;
    }
}
```

Getting started

Skeleton

```
$ composer create-project zendframework/zend-expressive-skeleton new-app
```

```
rob@swiftsure ~ $ composer create-project "zendframework/zend-expressive-skeleton:3.0.x-dev" new-app
Installing zendframework/zend-expressive-skeleton (3.0.x-dev c7b5c746c15f0220380fc0d61c596ac9ef)
- Installing zendframework/zend-expressive-skeleton (dev-release-3.0.0 release-3.0.0): Cloning repository
Created project in new-app
> ExpressiveInstaller\OptionalPackages::install
Setting up optional packages
Setup data and cache dir
Removing installer development dependencies

What type of installation would you like?
[1] Minimal (no default middleware, templates, or assets; configuration only)
[2] Flat (flat source code structure; default selection)
[3] Modular (modular source code structure; recommended)
Make your selection (2): 2
- Copying src/App/ConfigProvider.php

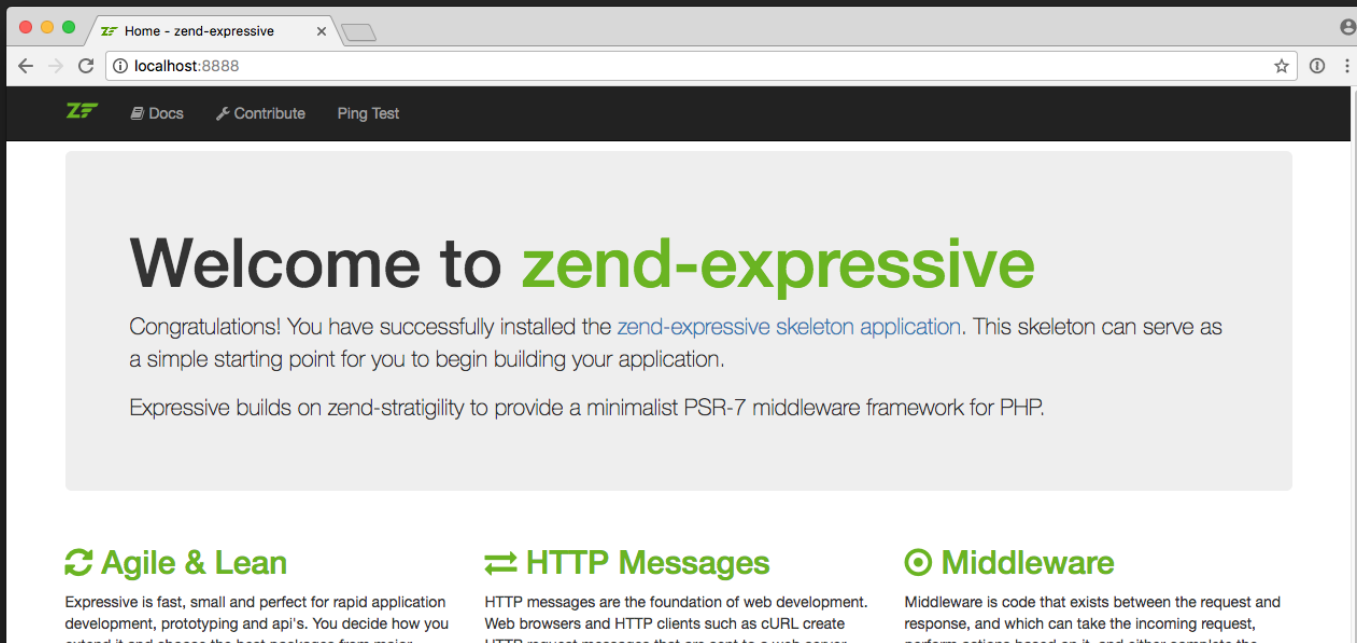
Which container do you want to use for dependency injection?
[1] Aura.Di
[2] Pimple
[3] Zend ServiceManager
[4] Aurn
[5] Symfony DI Container
Make your selection or type a composer package name and version (Zend ServiceManager): 3
- Adding package zendframework/zend-service-manager (^3.3)
- Copying config/container.php
```

```
2. php
Which template engine do you want to use?
[1] Plates
[2] Twig
[3] Zend View installs Zend ServiceManager
[n] None of the above
Make your selection or type a composer package name and version (n): 2
- Adding package zendframework/zend-expressive-twig-renderer (^2.0.0-dev)
- Copying config/autoload/templates.global.php
- Copying templates/error/404.html.twig
- Copying templates/error/error.html.twig
- Copying templates/layout/default.html.twig
- Copying templates/app/home-page.html.twig

Which error handler do you want to use during development?
[1] Whoops
[n] None of the above
Make your selection or type a composer package name and version (Whoops): 1
- Adding package filp/whoops (^2.1.12)
- Copying config/autoload/development.local.php.dist
Remove installer
Removing composer.lock from .gitignore
Removing Expressive installer classes, configuration, tests and docs
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 55 installs, 0 updates, 0 removals
- Installing zendframework/zend-component-installer (1.1.0): Loading from cache
```



Skeleton



Directory structure

```
.
├── bin/
├── config/
│   ├── autoload/
│   │   ├── dependencies.global.php
│   │   ├── development.local.php
│   │   ├── development.local.php.dist
│   │   ├── local.php.dist
│   │   ├── router.global.php
│   │   ├── templates.global.php
│   │   └── zend-expressive.global.php
│   ├── config.php
│   ├── container.php
│   ├── development.config.php
│   ├── development.config.php.dist
│   ├── pipeline.php
│   └── routes.php
├── data/
├── public/
│   ├── css/
│   ├── js/
│   └── index.php
├── src/
│   └── App/
├── test/
│   └── AppTest/
├── vendor/
├── composer.json
├── composer.lock
├── phpcs.xml.dist
└── phpunit.xml.dist
```

src/App directory

- Each module lives in its own namespace
- Contains all code for application
- `ConfigProvider` class for initialisation
 - Configuration
 - DI registration

src/App directory structure

```
└─ src/
  └─ App/
    └─ src/
      └─ Action/
        ├── HomePageAction.php
        ├── HomePageFactory.php
        └─ PingAction.php
      └─ ConfigProvider.php
    └─ templates/
      ├── app/
        └─ home-page.html.twig
      ├── error/
        ├── 404.html.twig
        └─ error.html.twig
      └─ layout/
        └─ default.html.twig
    └─ test/
      └─ AppTest/
        └─ Action/
```

An action

```
namespace App\Action;

use ...;

class HomePageAction implements MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) {
        return new HtmlResponse('<p>Hello World</p>');
    }
}
```

Let's write a web page!

Bitcoin conversion

*Create a page that displays the current value of
1 Bitcoin in £, \$ & €*

Create a route

config/routes.php:

```
$app->get(  
    '/bitcoin',  
    App\Action\BitcoinPageAction::class,  
    'bitcoin'  
);
```



Routes have a method

config/routes.php:

```
$app->get(  
    '/bitcoin',  
    App\Action\BitcoinPageAction::class,  
    'bitcoin'  
);
```

HTTP Method

```
$app->get()  
$app->post()  
$app->put()  
$app->patch()  
$app->delete()
```

Multiple methods:

```
$app->any()  
$app->route(..., ..., ['GET', 'POST'], ...);
```

Routes have a pattern

config/routes.php:

```
$app->get(  
    '/bitcoin',  
    App\Action\BitcoinPageAction::class,  
    'bitcoin'  
);
```

FastRoute URI pattern

- Literal string match

```
$app->get('/hello', ...);
```

FastRoute URI pattern

- Literal string match

```
$app->get('/hello', ...);
```

- Placeholders are wrapped in { }

```
$app->get('/hello/{name}', ...);
```

FastRoute URI pattern

- Literal string match

```
$app->get('/hello', ...);
```

- Placeholders are wrapped in { }

```
$app->get('/hello/{name}', ...);
```

- Optional segments are wrapped with []

```
$app->get('/news[/]{year}[/]{month}]', ...);
```

FastRoute URI pattern

- Literal string match

```
$app->get('/hello', ...);
```

- Placeholders are wrapped in { }

```
$app->get('/hello/{name}', ...);
```

- Optional segments are wrapped with []

```
$app->get('/news[/year][/{month}]', ...);
```

- Constrain placeholders via Regex

```
$app->get('/news/{year:\d{4}}', ...); // exactly 4 digits
```


Routes have a name

config/routes.php:

```
$app->get(  
    '/bitcoin',  
    App\Action\BitcoinPageAction::class,  
    'bitcoin'  
);
```

Url helper

- Builds URLs from route names
- Can be helpful to use `.` to group related route names

Use the router:

```
$router->generateUri('user.profile', ['name' => 'Rob']);
```

or in the template:

```
{{ path('user.profile', {'name': 'Rob'}) }}
```



Routes have an action

config/routes.php:

```
$app->get(  
    '/bitcoin',  
    App\Action\BitcoinPageAction::class,  
    'bitcoin'  
);
```

Actions

- Receive a PSR-7 Request
- Manage business logic operations
- Must return a PSR-7 Response
- Implemented as PSR-15 Middleware

Bitcoin action

```
class BitcoinPageAction implements MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface {

        $data['prices'] = $this->btcService->getCurrentPrices();

        return new HtmlResponse(
            $this->template->render('app::bitcoin-page', $data)
        );
    }
}
```

Bitcoin action

```
class BitcoinPageAction implements MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface {

        $data['prices'] = $this->btcService->getCurrentPrices();

        return new HtmlResponse(
            $this->template->render('app::bitcoin-page', $data)
        );
    }
}
```

Bitcoin action

```
class BitcoinPageAction implements MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface {

        $data['prices'] = $this->btcService->getCurrentPrices();

        return new HtmlResponse(
            $this->template->render('app::bitcoin-page', $data)
        );
    }
}
```

Bitcoin action

```
class BitcoinPageAction implements MiddlewareInterface
{
    public function process(
        ServerRequestInterface $request,
        RequestHandlerInterface $handler
    ) : ResponseInterface {

        $data['prices'] = $this->btcService->getCurrentPrices();

        return new HtmlResponse(
            $this->template->render('app::bitcoin-page', $data)
        );
    }
}
```


Injecting dependencies

```
class BitcoinPageAction implements MiddlewareInterface
{
    protected $template;
    protected $btcService;

    public function __construct(
        TemplateRendererInterface $template,
        BitcoinService $btcService
    ) {
        $this->template = $template;
        $this->btcService = $btcService;
    }

    // ...
}
```

Expressive configuration

A mashed-up associative array created from:

- Invoked `ConfigProvider` classes from libs and modules
- Config files in `config/autoload/`

Common top level keys:

- `dependencies`
- `twig`
- `validators`
- `cache`
- `templates`
- `filters`
- `db`

ConfigProvider

```
namespace App;
```

```
class ConfigProvider  
{  
    public function __invoke() : array  
    {  
        return [  
            'dependencies' => $this->getDependencies(),  
            'templates'    => $this->getTemplates(),  
        ];  
    }  
}
```

ConfigProvider

```
namespace App;
```

```
class ConfigProvider
```

```
{
```

```
    public function __invoke() : array
```

```
    {
```

```
        return [
```

```
            'dependencies' => $this->getDependencies(),
```

```
            'templates'    => $this->getTemplates(),
```

```
        ];
```

```
    }
```

Dependency configuration

```
public function getDependencies() : array
{
    return [
        'factories' => [
            Action\BitcoinPageAction::class =>
                Action\BitcoinPageFactory::class,
        ],
    ];
}
```

Dependency configuration

```
public function getDependencies() : array
{
    return [
        'factories' => [
            Action\BitcoinPageAction::class =>
                Action\BitcoinPageFactory::class,
        ],
    ];
}
```

Dependency configuration

```
public function getDependencies() : array
{
    return [
        'factories' => [
            Action\BitcoinPageAction::class =>
                Action\BitcoinPageFactory::class,
        ],
    ];
}
```

Action factory

Return an instance of the action with its injected dependencies

```
class BitcoinPageFactory
{
    public function __invoke($container)
    {
        return new BitcoinPageAction(
            $container->get(TemplateRendererInterface::class),
            $container->get(BitcoinService::class)
        );
    }
}
```


Action factory

Return an instance of the action with its injected dependencies

```
class BitcoinPageFactory
{
    public function __invoke($container)
    {
        return new BitcoinPageAction(
            $container->get(TemplateNameInterface::class),
            $container->get(BitcoinService::class)
        );
    }
}
```

Action factory

Return an instance of the action with its injected dependencies

```
class BitcoinPageFactory
{
    public function __invoke($container)
    {
        return new BitcoinPageAction(
            $container->get(TemplateRendererInterface::class),
            $container->get(BitcoinService::class)
        );
    }
}
```

Templates

Expressive's view layer

Templating

- Render method to convert page:

```
$html = $this->template->render('app::bitcoin-page', $data);
```

- Templates are namespaced: namespace::template
- Extension is resolved by the adapter:

```
app::bitcoin-page => app/bitcoin-page.html.twig
```

Twig templates

"Twig is a modern template engine for PHP"

- Manual: <https://twig.symfony.com>
- Script extension: *.twig*
- Variables: `{{ }}`
- Control statements: `{% %}`
- Comments: `{# #}`

Action template

```
{% extends '@layout/default.html.twig' %}

{% block title %}Bitcoin converter{% endblock %}

{% block content %}
  <h1>Bitcoin converter</h1>

  <p>One BTC:</p>
  {% for price in prices %}
    {{ price.symbol }}
    {{ price.rate_float|number_format(2, '.', ',') }}<br>
  {% endfor %}
{% endblock %}
```

Print variables

```
{% extends '@layout/default.html.twig' %}

{% block title %}Bitcoin converter{% endblock %}

{% block content %}
    <h1>Bitcoin converter</h1>

    <p>One BTC:</p>
    {% for price in prices %}
        {{ price.symbol }}
        {{ price.rate_float|number_format(2, '.', ',') }}<br>
    {% endfor %}
{% endblock %}
```

Control statements

```
{% extends '@layout/default.html.twig' %}

{% block title %}Bitcoin converter{% endblock %}

{% block content %}
    <h1>Bitcoin converter</h1>

    <p>One BTC:</p>
    {% for price in prices %}
        {{ price.symbol }}
        {{ price.rate_float|number_format(2, '.', ',') }}<br>
    {% endfor %}
{% endblock %}
```


Template inheritance

```
{% extends '@layout/default.html.twig' %}
```

```
{% block title %}Bitcoin converter{% endblock %}
```

```
{% block content %}
```

```
  <h1>Bitcoin converter</h1>
```

```
  <p>One BTC:</p>
```

```
  {% for price in prices %}
```

```
    {{ price.symbol }}
```

```
    {{ price.rate_float|number_format(2, '.', ',') }}<br>
```

```
  {% endfor %}
```

```
{% endblock %}
```

Template inheritance

- For cohesive look and feel
 - includes default CSS, JS & structural HTML
- Build a base skeleton
- Define *blocks* for children to override
- Each child chooses which template to inherit from

Base skeleton

src/App/templates/layout/default.html.twig:

```
<!DOCTYPE html>
<html>
  <head>
    {% block head %}
      <link rel="stylesheet" href="style.css" />
      <title>{% block title %}{% endblock %} - Akrobat</title>
    {% endblock %}
  </head>
  <body>
    {% block content %}{% endblock %}
    <footer>{% block footer %}&copy; 2017{% endblock %}</footer>
  </body>
</html>
```

Base skeleton

```
src/App/templates/layout/default.html.twig:
```

```
<!DOCTYPE html>
<html>
  <head>
    {% block head %}
      <link rel="stylesheet" href="style.css" />
      <title>{% block title %}{% endblock %} - Akrobat</title>
    {% endblock %}
  </head>
  <body>
    {% block content %}{% endblock %}
    <footer>{% block footer %}&copy; 2017{% endblock %}</footer>
  </body>
</html>
```

Base skeleton

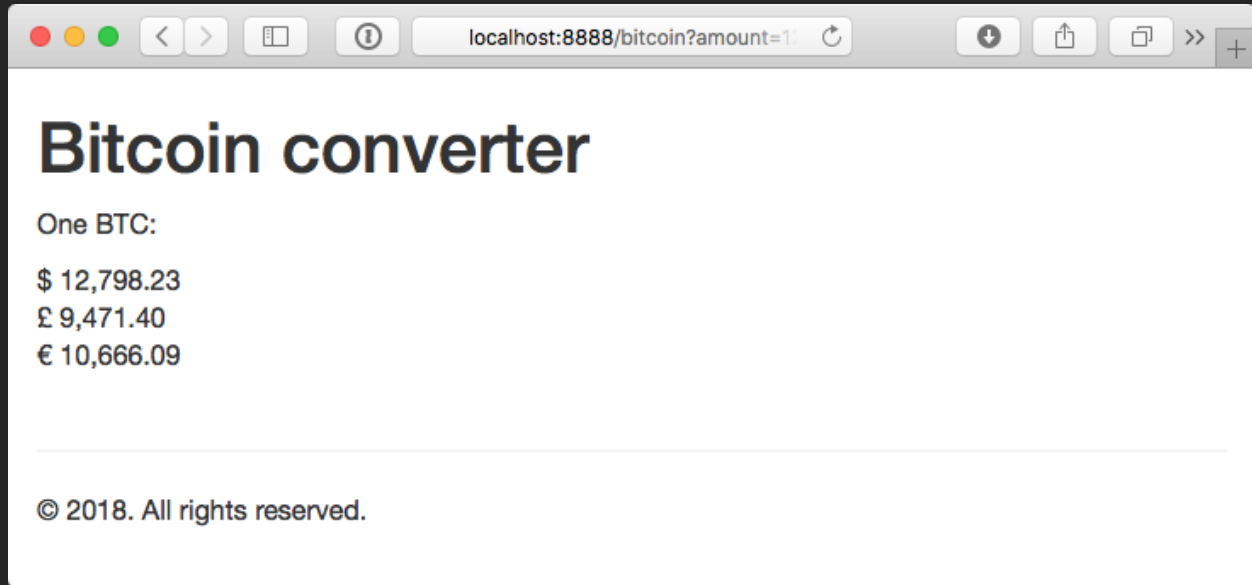
```
src/App/templates/layout/default.html.twig:
```

```
<!DOCTYPE html>
<html>
  <head>
    {% block head %}
      <link rel="stylesheet" href="style.css" />
      <title>{% block title %}{% endblock %} - Akrobat</title>
    {% endblock %}
  </head>
  <body>
    {% block content %}{% endblock %}
    <footer>{% block footer %}&copy; 2017{% endblock %}</footer>
  </body>
</html>
```

Base skeleton

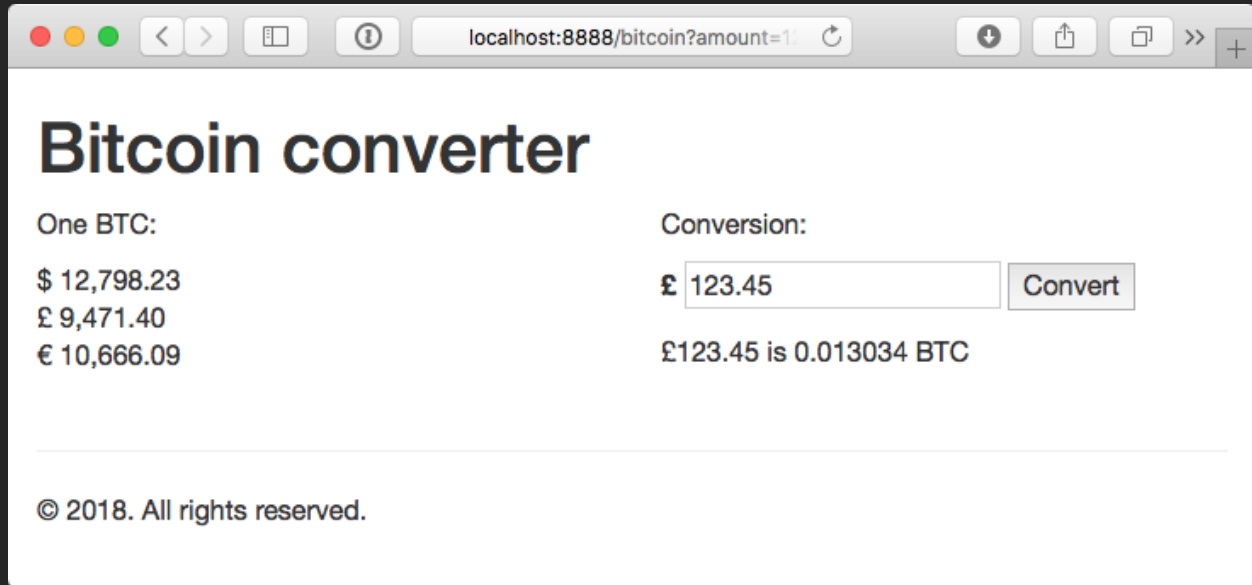
```
src/App/templates/layout/default.html.twig:
```

```
<!DOCTYPE html>
<html>
  <head>
    {% block head %}
      <link rel="stylesheet" href="style.css" />
      <title>{% block title %}{% endblock %} - Akrobat</title>
    {% endblock %}
  </head>
  <body>
    {% block content %}{% endblock %}
    <footer>{% block footer %}&copy; 2017{% endblock %}</footer>
  </body>
</html>
```



Adding components

Add a form



The screenshot shows a web browser window with the address bar containing `localhost:8888/bitcoin?amount=1`. The page title is "Bitcoin converter". On the left, under "One BTC:", the values are listed: \$ 12,798.23, £ 9,471.40, and € 10,666.09. On the right, under "Conversion:", there is a text input field containing "123.45" with a "Convert" button to its right. Below the input field, the text "£123.45 is 0.013034 BTC" is displayed. At the bottom of the page, a copyright notice reads "© 2018. All rights reserved."

localhost:8888/bitcoin?amount=1

Bitcoin converter

One BTC:

- \$ 12,798.23
- £ 9,471.40
- € 10,666.09

Conversion:

£

£123.45 is 0.013034 BTC

© 2018. All rights reserved.

Add a form: HTML

```
<form method="GET" action="/bitcoin">  
  <label>&pound;</label>  
  <input name="amount" value="{{ amount }}">  
  <button>Convert</button>  
</form>
```

```
<p>&pound;{{amount}} is {{ bitcoins|number_format(6) }} BTC
```

Add a form: HTML

```
<form method="GET" action="/bitcoin">  
  <label>&pound;</label>  
  <input name="amount" value="{{ amount }}">  
  <button>Convert</button>  
</form>
```

```
<p>&pound;{{amount}} is {{ bitcoins|number_format(6) }} BTC
```

Add a form: HTML

```
<form method="GET" action="/bitcoin">  
  <label>&pound;</label>  
  <input name="amount" value="{{ amount }}">  
  <button>Convert</button>  
</form>
```

```
<p>&pound;{{amount}} is {{ bitcoins|number_format(6) }} BTC
```

Add a form: HTML

```
<form method="GET" action="/bitcoin">  
  <label>&pound;</label>  
  <input name="amount" value="{{ amount }}">  
  <button>Convert</button>  
</form>
```

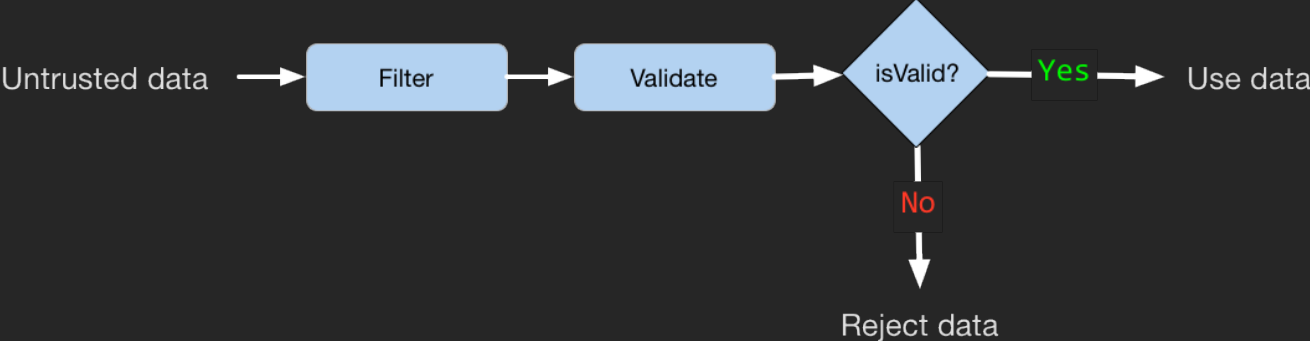
```
<p>&pound;{{amount}} is {{ bitcoins|number_format(6) }} BTC
```

Add a form: HTML

```
<form method="GET" action="/bitcoin">
  <label>&pound;</label>
  <input name="amount" value="{{ amount }}">
  <button>Convert</button>
</form>
```

```
<p>&pound;{{amount}} is {{ bitcoins|number_format(6) }} BTC</p>
```

Zend-InputFilter



Integration via ConfigProvider

```
3. bash
rob@MacBook-Pro ~/bitcoinapp $ composer require zendframework/zend-inputfilter
Using version ^2.8 for zendframework/zend-inputfilter
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 3 installs, 0 updates, 0 removals
 - Installing zendframework/zend-validator (2.10.1): Loading from cache

Please select which config file you wish to inject 'Zend\Validator\ConfigProvider' into:
[0] Do not inject
[1] config/config.php
Make your selection (default is 0):1

Remember this option for other packages of the same type? (y/N)y
Installing Zend\Validator\ConfigProvider from package zendframework/zend-validator
```



Create an input filter

```
$factory = new Zend\InputFilter\Factory();  
$inputFilter = $factory->createInputFilter([  
    'amount' => [  
        'filters' => [  
            ['name' => 'ToInt'],  
        ],  
        'validators' => [  
            [  
                'name' => 'GreaterThan',  
                'options' => ['min' => 0],  
            ],  
        ],  
    ],  
]);
```

Create an input filter

```
$factory = new Zend\InputFilter\Factory();  
$inputFilter = $factory->createInputFilter([  
    'amount' => [  
        'filters' => [  
            ['name' => 'ToInt'],  
        ],  
        'validators' => [  
            [  
                'name' => 'GreaterThan',  
                'options' => ['min' => 0],  
            ],  
        ],  
    ],  
]);
```

Create an input filter

```
$factory = new Zend\InputFilter\Factory();  
$inputFilter = $factory->createInputFilter([  
    'amount' => [  
        'filters' => [  
            ['name' => 'ToInt'],  
        ],  
        'validators' => [  
            [  
                'name' => 'GreaterThan',  
                'options' => ['min' => 0],  
            ],  
        ],  
    ],  
]);
```

Create an input filter

```
$factory = new Zend\InputFilter\Factory();  
$inputFilter = $factory->createInputFilter([  
    'amount' => [  
        'filters' => [  
            ['name' => 'ToInt'],  
        ],  
        'validators' => [  
            [  
                'name' => 'GreaterThan',  
                'options' => ['min' => 0],  
            ],  
        ],  
    ],  
]);
```

Create an input filter

```
$factory = new Zend\InputFilter\Factory();  
$inputFilter = $factory->createInputFilter([  
    'amount' => [  
        'filters' => [  
            ['name' => 'ToInt'],  
        ],  
        'validators' => [  
            [  
                'name' => 'GreaterThan',  
                'options' => ['min' => 0],  
            ],  
        ],  
    ],  
]);
```

Validating request data

1. Retrieve data from Request object
2. Pass to `InputFilter`
3. Call `isValid()`
4. Retrieve sanitized, valid data using `getValues()`
5. Use `getMessages()` to find out what failed

Validating request data

```
public function process($request, $handler)
{
    $requestData = $request->getQueryParams();
    $this->inputFilter->setData($requestData);

    if ($this->inputFilter->isValid()) {
        /* request data is valid */
        $values = $this->inputFilter->getValues();
        $btc = $this->btcService->convert($values['amount']);
    } else {
        /* request data is invalid */
        $errors = $this->inputFilter->getMessages();
    }
}
```

Validating request data

```
public function process($request, $handler)
{
    $requestData = $request->getQueryParams();
    $this->inputFilter->setData($requestData);

    if ($this->inputFilter->isValid()) {
        /* request data is valid */
        $values = $this->inputFilter->getValues();
        $btc = $this->btcService->convert($values['amount']);
    } else {
        /* request data is invalid */
        $errors = $this->inputFilter->getMessages();
    }
}
```


Validating request data

```
public function process($request, $handler)
{
    $requestData = $request->getQueryParams();
    $this->inputFilter->setData($requestData);

    if ($this->inputFilter->isValid()) {
        /* request data is valid */
        $values = $this->inputFilter->getValues();
        $btc = $this->btcService->convert($values['amount']);
    } else {
        /* request data is invalid */
        $errors = $this->inputFilter->getMessages();
    }
}
```

Validating request data

```
public function process($request, $handler)
{
    $requestData = $request->getQueryParams();
    $this->inputFilter->setData($requestData);

    if ($this->inputFilter->isValid()) {
        /* request data is valid */
        $values = $this->inputFilter->getValues();
        $btc = $this->btcService->convert($values['amount']);
    } else {
        /* request data is invalid */
        $errors = $this->inputFilter->getMessages();
    }
}
```

Validating request data

```
public function process($request, $handler)
{
    $requestData = $request->getQueryParams();
    $this->inputFilter->setData($requestData);

    if ($this->inputFilter->isValid()) {
        /* request data is valid */
        $values = $this->inputFilter->getValues();
        $btc = $this->btcService->convert($values['amount']);
    } else {
        /* request data is invalid */
        $errors = $this->inputFilter->getMessages();
    }
}
```

Summary



Resources

- <https://docs.zendframework.com/zend-expressive/>
- <https://github.com/zendframework/>

- <https://akrabat.com/category/zend-expressive/>
- <https://framework.zend.com/blog>

Thank you!



PHPTRAINING.CO.UK