PHP UNCONFERENCE HAMBURG

Lets talk about . . .

Stack

`and the idea behind PSR-7`

About me

Hi, i'm

Julius Beckmann

and i work at

SILPION

How does PHP work?

```php
<?php

// The whole request
// var_dump(
//     $_GET, $_POST, $_SERVER,
//     $_FILES, $_COOKIES, $_SESSION
// );

// Creating a response
header("HTTP/1.1 404 Not Found");
echo "Not found";
exit;
```

PHP SAPI - *Server Application Programming Interface*

Gives us:

- Superglobals
  - `$_GET`
  - `$_POST`
  - `$_SERVER`
  - ...
- Primitives
  - `header()`
  - `echo()`
  - `exit()`
  - ...

PHP-SAPI Problems

- Global state
- Hard to inspect / debug
- Hard to reproduce / test

Lets do it better!

HTTP

A protocol is a interface.

The HttpKernelInterface

```php
<?php

namespace Symfony\Component\HttpKernel;

interface HttpKernelInterface
{
    const MASTER_REQUEST = 1;
    const SUB_REQUEST = 2;

    /**
     * Handles a Request to convert it to a Response.
     *
     * @return Response
     */
    public function handle(
        Request $request,
        $type = self::MASTER_REQUEST,
        $catch = true
    );
}
```

Why use HttpKernelInterface?

`Request`, `AppKernel` and `Response` are **values**.

- Inspectable
- Reuseable
- Composeable

PHP SAPI is **global state**.

- *Nothing of the above*

Its nothing new . . .

- 1997: Java Servlet
- 2003: Python WSGI *(Web Server Gateway Interface)*
- 2007: Ruby Rack
- 2009: Perl PSGI/Plack
- 2011: Symfony HttpKernelInterface
- ????: *PSR-7: HTTP message interfaces*

$\rightarrow$ Idea is more than *17 Years* around, lets use it!

Leverage HttpKernelInterface using StackPHP

Home Middlewares Toolbox Conventions Media

Stack

Composing HttpKernelInterface middlewares since 2013!

stackphp @stackphp #stackphp

Brought to you by

@igorwhiletrue

@beausimensen

@hochchristoph

Needed composer packages:

```
{
    "require": {
        "symfony/http-kernel": "~2.0",
        "symfony/http-foundation": "~2.0"
    }
}
```

Example HttpKernelInterface usage

```php
<?php

$kernel = new AppKernel('dev', true);

$response = $kernel->handle(Request::createFromGlobals());

$response->send();
```

Request $\rightarrow$

AppKernel

$\rightarrow$ *Response*

`AppKernel extends HttpKernelInterface`

How would you . . . .

restrict Access by IP?

With IpRestrict by Alsar

```php
<?php

$kernel = new AppKernel('dev', true);

$kernel = new Alsar\Stack\IpRestrict(
    $kernel,
    array('127.0.0.1', '192.168.0.1')
);

$response = $kernel->handle(Request::createFromGlobals());

$response->send();
```

https://github.com/alsar/stack-ip-restrict

Principle of Stack

$Request \rightarrow$

DecoratingApp(App)

$\rightarrow Response$

StackPHP is . . .

StackPHP is **NOT** a Framework

StackPHP is **NOT** a Library

StackPHP is a Toolset and Conventions

StackPHP Conventions

1. Implement the **HttpKernelInterface**
2. Take the decorated app as the **first constructor argument**
3. **Decorate** the handle call, **delegate** to the decorated app

More can be found here: *stackphp.com/specs*.

Lets check out the toolbox!

- Builder
- Run
- Session
- URL Map
- Lazy Kernel

Toolbox: Builder

Stacking Kernels can be confusing, the `Builder` can help here.

```php
<?php

$stack = (new Stack\Builder())
    ->push('Stack\Session')
    ->push('SymfonyHttpCache', new Store('../cache'));

$app = $stack->resolve($app);
```

Result: *Session(SymfonyHttpCache($app, '../cache'))*

Toolbox: Run

Stack\run() is just a shortcut:

```php
<?php

namespace Stack;

use Symfony\Component\HttpKernel\HttpKernelInterface;
use Symfony\Component\HttpKernel\TerminableInterface;
use Symfony\Component\HttpFoundation\Request;

function run(HttpKernelInterface $app, Request $request = null)
{
    $request = $request ?: Request::createFromGlobals();

    $response = $app->handle($request);
    $response->send();
    if ($app instanceof TerminableInterface) {
        $app->terminate($request, $response);
    }
}
```

Toolbox: Session

Injecting a Session to the `Request`

```php
<?php

$app = new Silex\Application();
$app->get('/', function (Request $request) {
    $session = $request->getSession();
    // use session ...
});


$stack = (new Stack\Builder())->push('Stack\Session');

Stack\run($stack->resolve($app));
```

Toolbox: UrlMap

Mapping URL-Prefixes to Apps.

```php
<?php

$map = array(
    '/api' => new ApiKernel(),
    '/blog' => new BlogKernel(),
    '/' => new WebsiteKernel(),
);

$app = (new Stack\Builder())->push('Stack\UrlMap', $map)

Stack\run($stack->resolve($app));
```

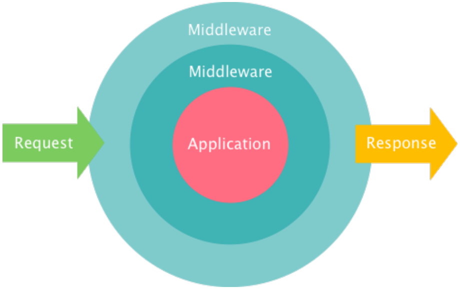
Toolbox: LazyHttpKernel

Lazy proxy for `HttpKernelInterfaces`.

```php
<?php

$map = array(
    '/api' => Stack\lazy(function () {
        return new ApiKernel();
    }),
    '/blog' => Stack\lazy(function () {
        return new BlogKernel();
    }),
    '/' => Stack\lazy(function () {
        return new WebsiteKernel();
    }),
);

// ...
```

Middlewares

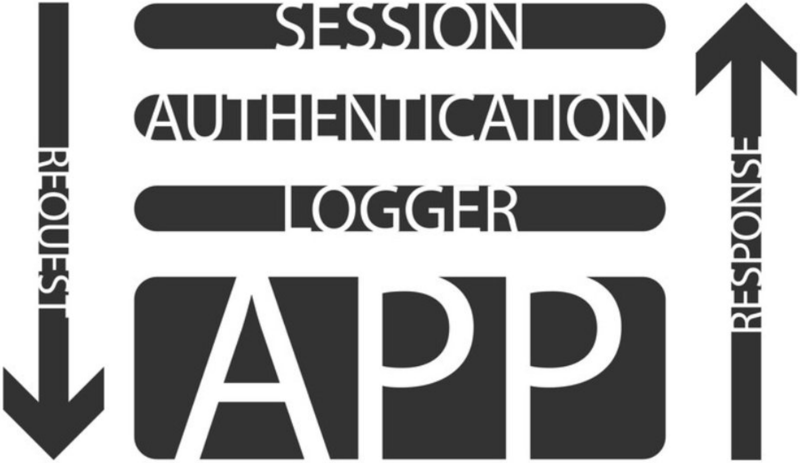
Adding functionality by decorating Application.



Example Middleware

```php
<?php

class MyMiddleware implements HttpKernelInterface
{
    private $app;

    public function __construct(HttpKernelInterface)
    {
        $this->app = $app;
    }

    public function handle(Request $request, $type = self::MASTER_REQUEST, $catch = true)
    {
        // Before request

        $response = $this->app->handle($request, $type, $catch);

        // After response

        return $response;
    }
}
```

What the Stack logo means



List of middlewares

stackphp.com/middlewares/

- HttpCache *(by Symfony)*
- CookieGuard *(by Laravel)*
- GeoIp *(by geocoder)*
- IpRestrict *(by alsar)*
- Backstage *(by atst)*
- OAuth *(by igorw)*
- Basic Authentication *(by dflydev)*
- Hawk *(by dflydev)*
- CORS *(by asm89)*
- Robots *(by dongilbert)*
- Negotiation *(by wildurand)*
- Honeypot *(by CHH)*
- Turbolinks *(by Helthe)*
- Logger *(by h4cc)*

List of Apps using HttpKernelInterface

symfony.com/components/HttpKernel

- Sculpin
- Symfony
- Proem Framework
- phpBB
- Drupal
- Thelia
- Shopware
- Silex
- Pagekit
- Laravel

What about PSR-7 ?

StackPHP is currently using `symfony` packages.

A equal PSR-Standard is proposed, but not yet ready :(

But let me show you the drafts. . .

PSR-7 MessageInterface DRAFT

Base for `Request` and `Response`

```php
<?php

namespace Psr\Http\Message;

interface MessageInterface
{
    public function getProtocolVersion();

    public function getBody();
    public function setBody(StreamInterface $body = null);

    public function getHeaders();
    public function hasHeader($header);
    public function getHeader($header);
    public function getHeaderAsArray($header);
    public function setHeader($header, $value);
    public function setHeaders(array $headers);
    public function addHeader($header, $value);
    public function addHeaders(array $headers);
    public function removeHeader($header);
}
```

PSR-7 RequestInterface DRAFT

```php
<?php

namespace Psr\Http\Message;

interface RequestInterface extends MessageInterface
{
    public function getMethod();
    public function setMethod($method);

    public function getUrl();
    public function setUrl($url);
}
```

PSR-7 ResponseInterface DRAFT

```php
<?php

namespace Psr\Http\Message;

interface ResponseInterface extends MessageInterface
{
    public function getStatusCode();
    public function setStatusCode($code);

    public function getReasonPhrase();
    public function setReasonPhrase($phrase);
}
```

Summary

- Using **Http as a interface** with HttpKernelInterface.
- **Compose** simple Applications.
- Add functionality through **Middlewares**.
- Build framework-agnostic features based on HTTP.

Stuff i did with StackPHP

- Request/Response logger: **silpion/stack-logger**
- REST fileserver application: **h4cc/stack-flysystem**
- StackPHP adapter for Mongrel2: **h4cc/stack-mongrel2**
- ReactPHP StackPHP Handler: **yosymfony/httpserver**
- *More not yet released stuff . . .*

About me

Julius Beckmann

(twitter|github).com/**h4cc**

- PHP
  - Symfony
  - Silex
- BEAM
  - Erlang
  - Elixir
- DevOps
  - Server
  - Deployment
- Pandoc
  - Markdown
  - LaTeX

Questions?