Symfony UserGroup Hamburg

Lets talk about . . .

"Do you stack up?"

About me

Julius Beckmann

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

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

Home Middlewares Toolbox Conventions Media

Stack

Composing HttpKernelInterface middlewares since 2013!

stackphp @stackphp #stackphp

Brought to you by

@igorwhiletrue

@beausimensen

@hochchristoph

Modularization in Frameworks

→ Bundles

→ ServiceProviders

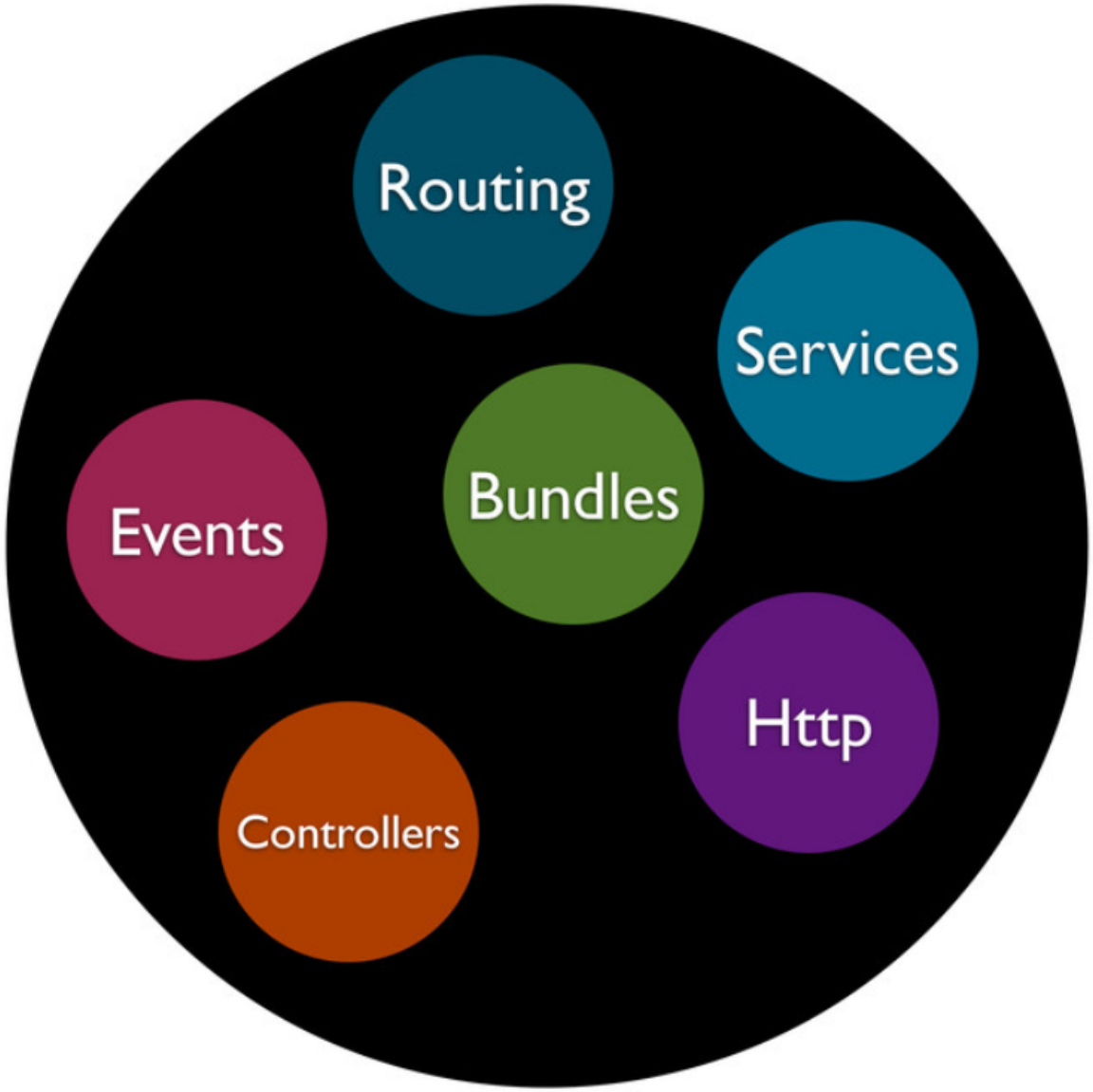
→ Modules

→ Modules

→ ServiceProviders

→ ???

Inside the Frameworks



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
    );
}
```

Example HttpKernelInterface usage

```php
<?php

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

$kernel->handle(Request::createFromGlobals())
       ->send();
```

Base of Stack

Request $\longrightarrow$ **App** $\longrightarrow$ *Response*

`App extends HttpKernelInterface`

How would you ....

restrict Access to
your App 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')
);

$kernel->handle(Request::createFromGlobals())
       ->send();
```

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

With IpRestrict by Alsar

```php
<?php

class IpRestrict implements HttpKernelInterface
{
    private $app;
    private $allowedIps;

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

    public function handle(Request $request, $type = self::MASTER_REQUEST, $catch = true)
    {
        $ip = $request->getClientIp();

        if (!in_array($ip, $this->allowedIps)) {
            return new Response(sprintf('IP %s is not allowed.', $ip), 403);
        }

        return $this->app->handle($request, $type, $catch);
    }
}
```

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

Principle of Stack

Request $\longrightarrow$

AnotherApp(App)

$\longrightarrow$ *Response*

Wait, how did we do this till now?

PHP SAPI - *Server Application Programming Interface*

Gives us:

- Superglobals
  - `$_REQUEST`
  - `$_SERVER`
- Primitives
  - `header()`
  - `echo()`
  - `exit()`

Why use HttpKernelInterface?

`Request`, `App` 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

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

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');

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

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

Toolbox: LazyHttpKernel

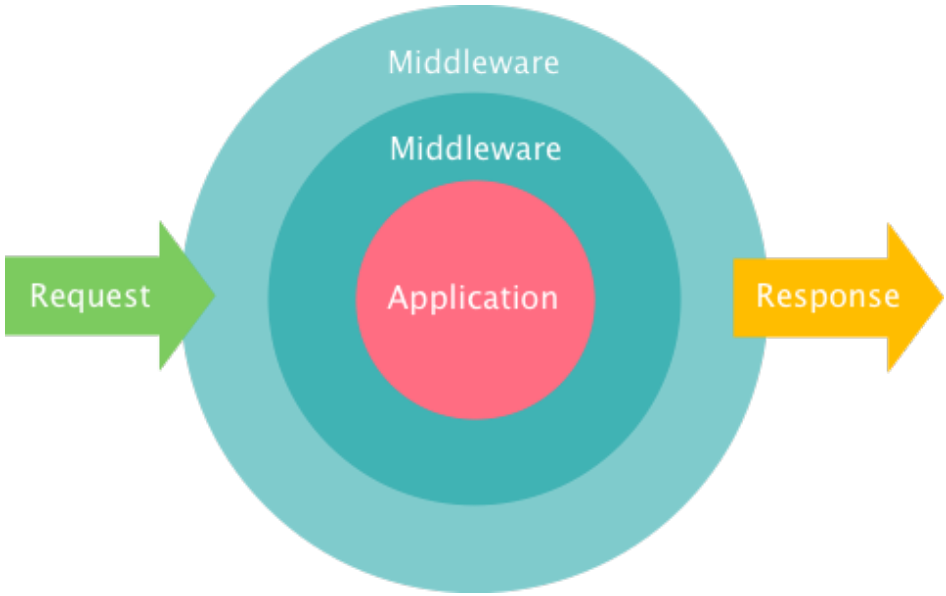
Lazy proxy for `HttpKernelInterfaces`.

```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.
- Decorating Application.

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

Divine the future

One day, using StackPHP we will:

- Combine multiple Application Kernels to one.
- Use Symfony, Silex, Drupal, Ez, BoltCM, ... in combination.
- Have single-purpose-application, decorated to our needs.
- even interact with external services using HttpKernelInterface.

Summary

- Using Http as a interface with HttpKernelInterface.
- Decorating Apps is easy.
- Writing own HttpKernel wrapper is easy.
- Nice Toolbox available.
- Public Middleware available

Stop writing Bundles.

Start writing Middlewares!

Questions?