

Modernising the Legacy

Marek Matulka



SensioLabs UK





Marek Matulka

Software Engineer

Sensio**Labs** UK



@super_marek

Legacy

What is legacy code?

- someone else's code
- your old code
- an old framework you don't know
- or too coupled to the framework
- built over years

What is legacy code?

Old code where...

- most of it is procedural
- no namespaces used
- no code convention
- no separation of concerns

What is legacy code?

Old code where...

- PHP mixed with HTML, CSS
- overused Global variables
- collection of include files
- *.inc files in public folder

What is legacy code?

Old code where...

- no concept of environments
- no error reporting
- no logging
- no version control system used

What is legacy code?

Any code...

- **without tests**

It's too hard

- understand the logic
- add new features
- debug
- fix bugs
- adopt new technologies

**How do we get
out of *legacy*?**



Airline postcards



bmi Boeing 757

£9.95

[view details](#)



Qantas Boeing 747

£7.95

[view details](#)



British Airways Boeing 747

£7.95

[view details](#)



JAL Boeing 777

£7.95

[view details](#)



British Airways Boeing 777

£7.95

[view details](#)



JAL Boeing 777

£7.95

[view details](#)



Royal Jordanian Airbus A310

£5.95

[view details](#)



Virgin Atlantic Airbus A340

£5.95

[view details](#)



EVA AIR Boeing 747

£7.95

[view details](#)



Air China Airbus A330



Singapore Airlines Airbus A380



Virgin Atlantic A340



Airline postcards

Basket

Product	Price	Quantity	Gross	
JAL Boeing 777	£7.95	2	£15.9	remove product
Virgin Atlantic Airbus A340	£5.95	1	£5.95	remove product
Total:		3	£21.85	

[Return to store](#) or [continue to checkout](#).

© 2005 Airline postcards

Airline postcards

Place an order

Products

You are about to order the following products:

Product	Price	Quantity	Gross
JAL Boeing 777	£7.95	2	£15.9
British Airways Boeing 777	£7.95	1	£7.95
Virgin Atlantic Airbus A340	£5.95	1	£5.95
Total:		4	£29.8

Postage

We will contact you regarding postage when we process your order.

Shipping address

Name:

Address:

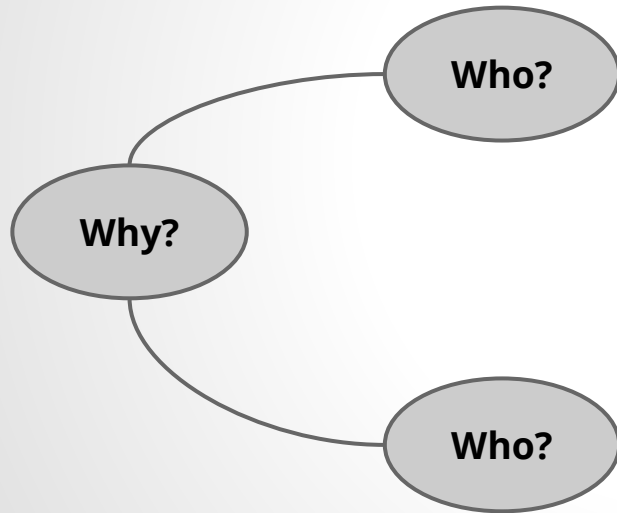
Start with the goal

Business goal

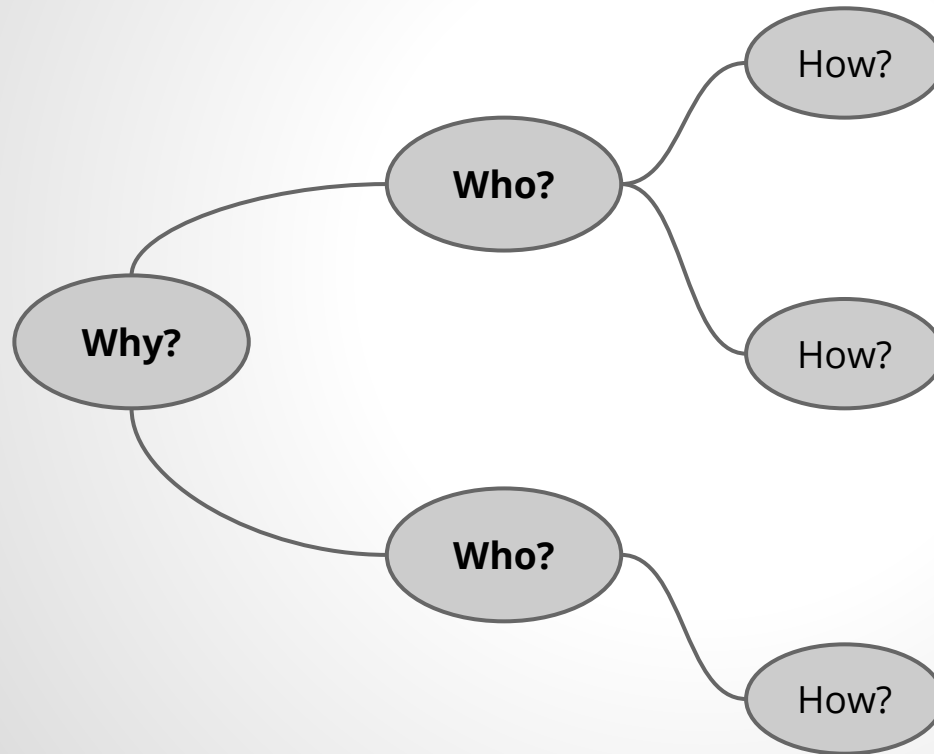


Why?

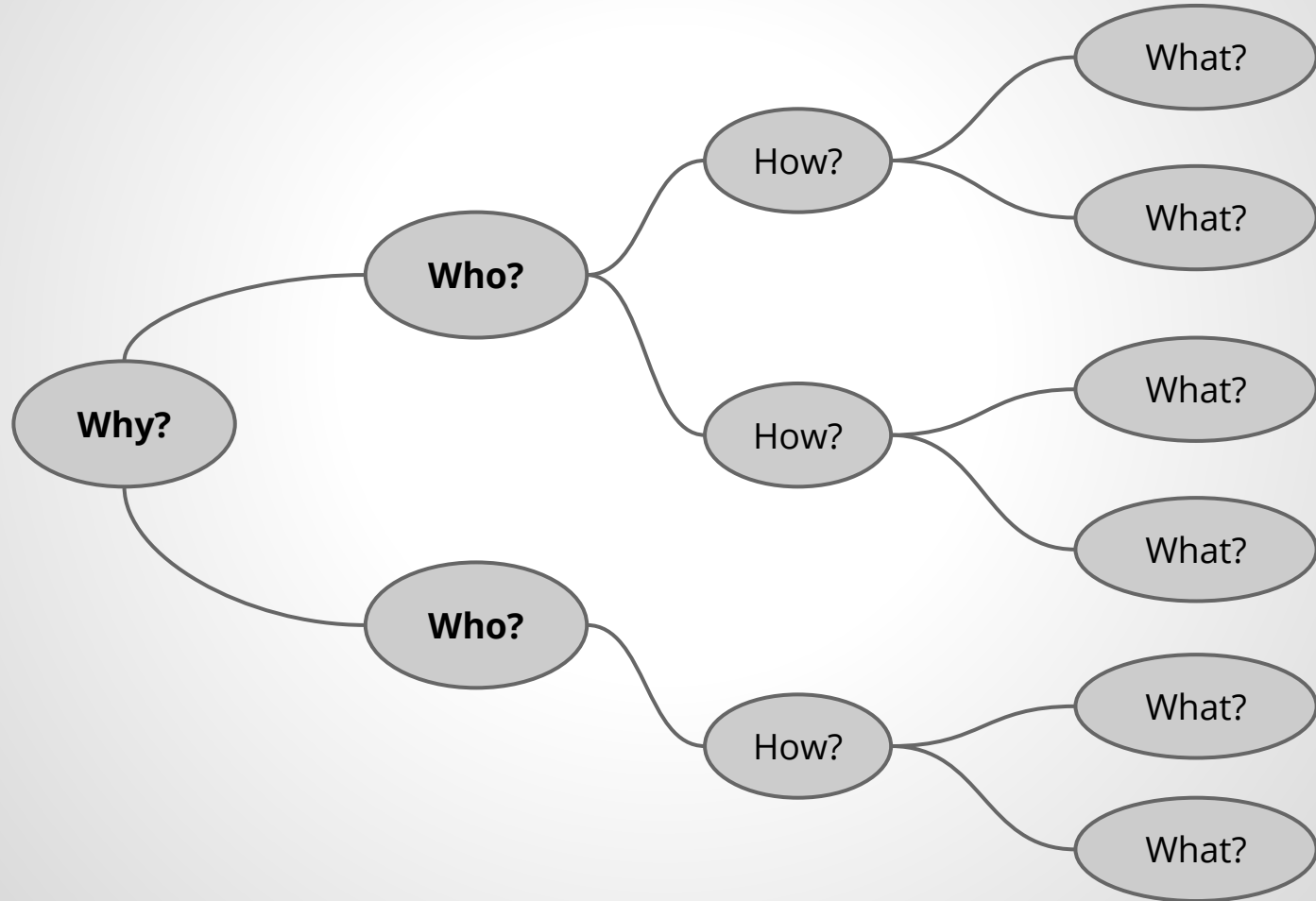
Actors



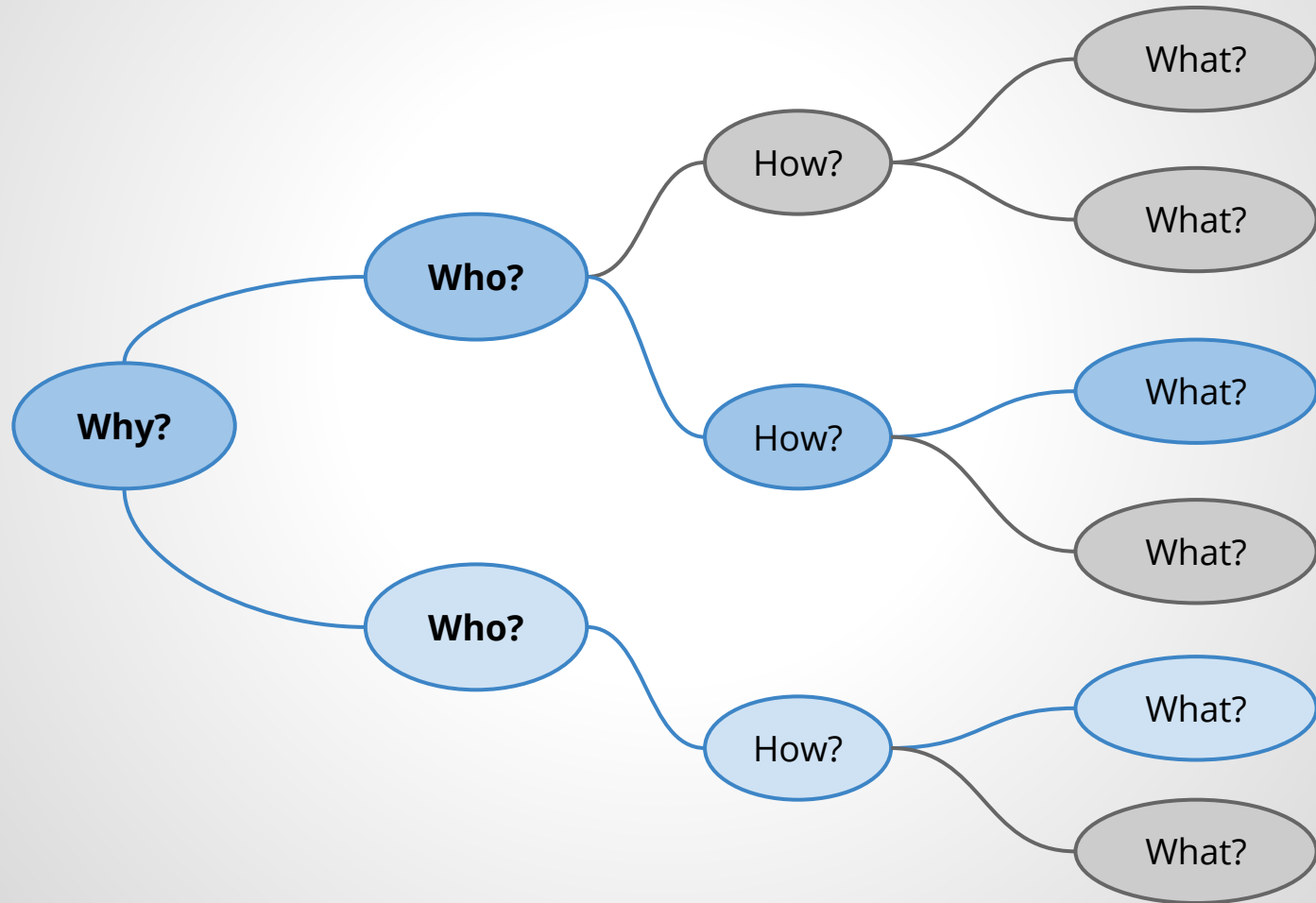
Impacts



Deliverables



Prioritise



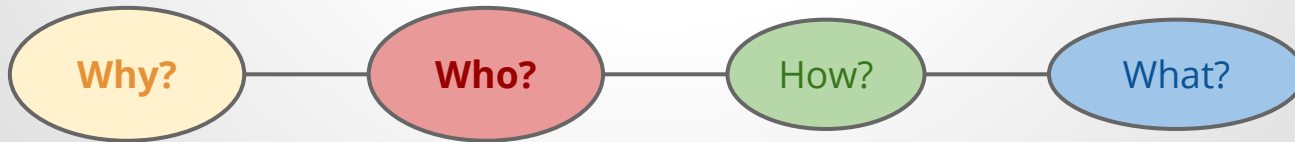
Example

Goal: **Reduce drop-outs by 10%**

In order to **see the total cost of order**

As a **shopper**

I want to **see postage cost**



Write scenarios

Feature: Shopper can see postage cost

In order to see the total cost of order

As a shopper

I want to see a postage cost

Scenario: Shopper can see a postage cost

Given I have added a "Clean Code" book priced £34.50 to the basket

And the postage cost for one book is £3.50

When I check the basket

Then I should see the postage cost of £3.50

And the total of the basket should be £38.00

Implement scenarios



Given-When-Then

Feature: Shopper can see postage cost

In order to see the total cost of order

As a shopper

I want to see a postage cost

Scenario: Shopper can see a postage cost

Given I have added a "Clean Code" book priced £34.50 to the basket

And the postage cost for one book is £3.50

When I check the basket

Then I should see the postage cost of £3.50

And the total of the basket should be £38.00

What to test?

(everything?)



End-to-end

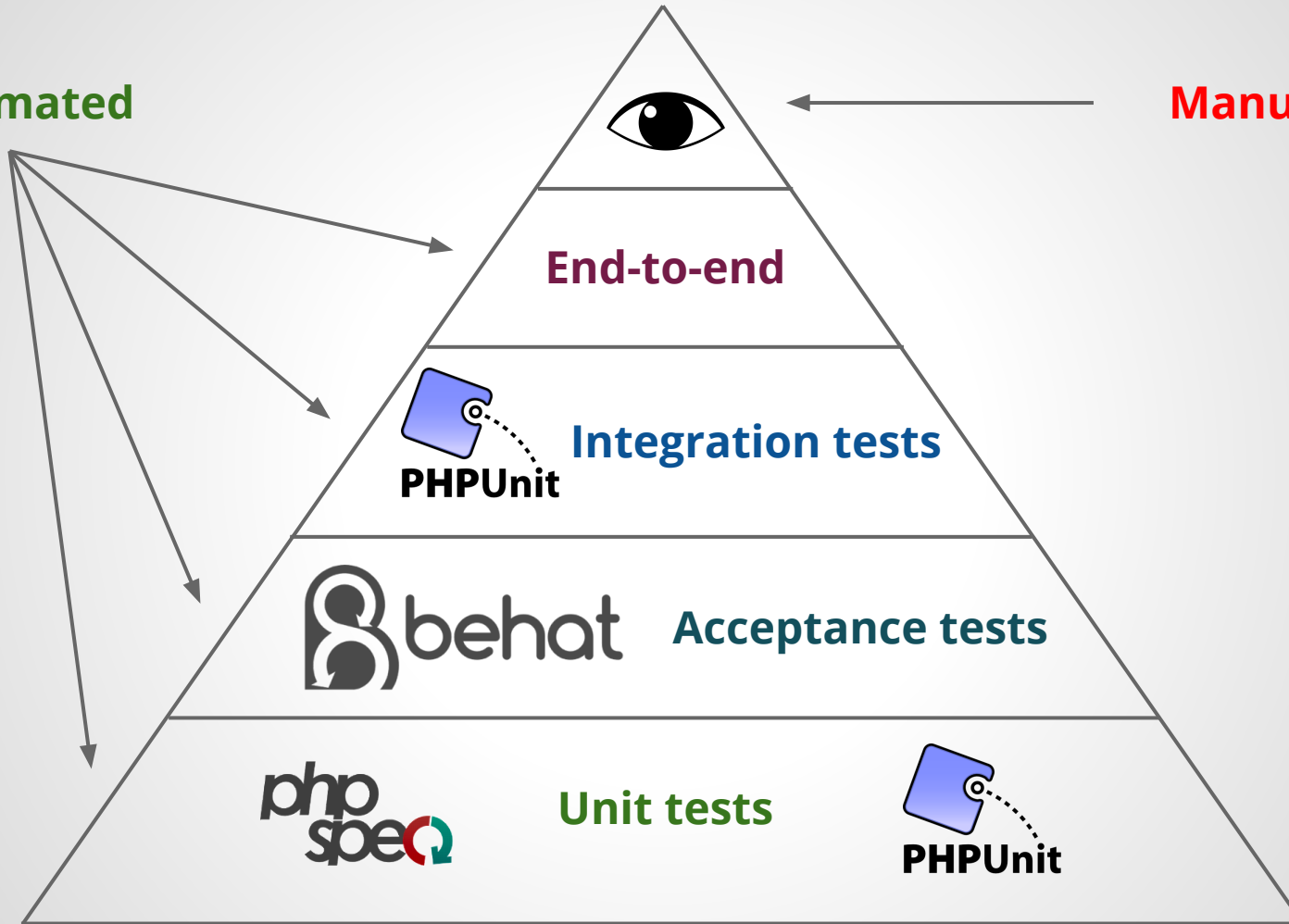
Integration tests

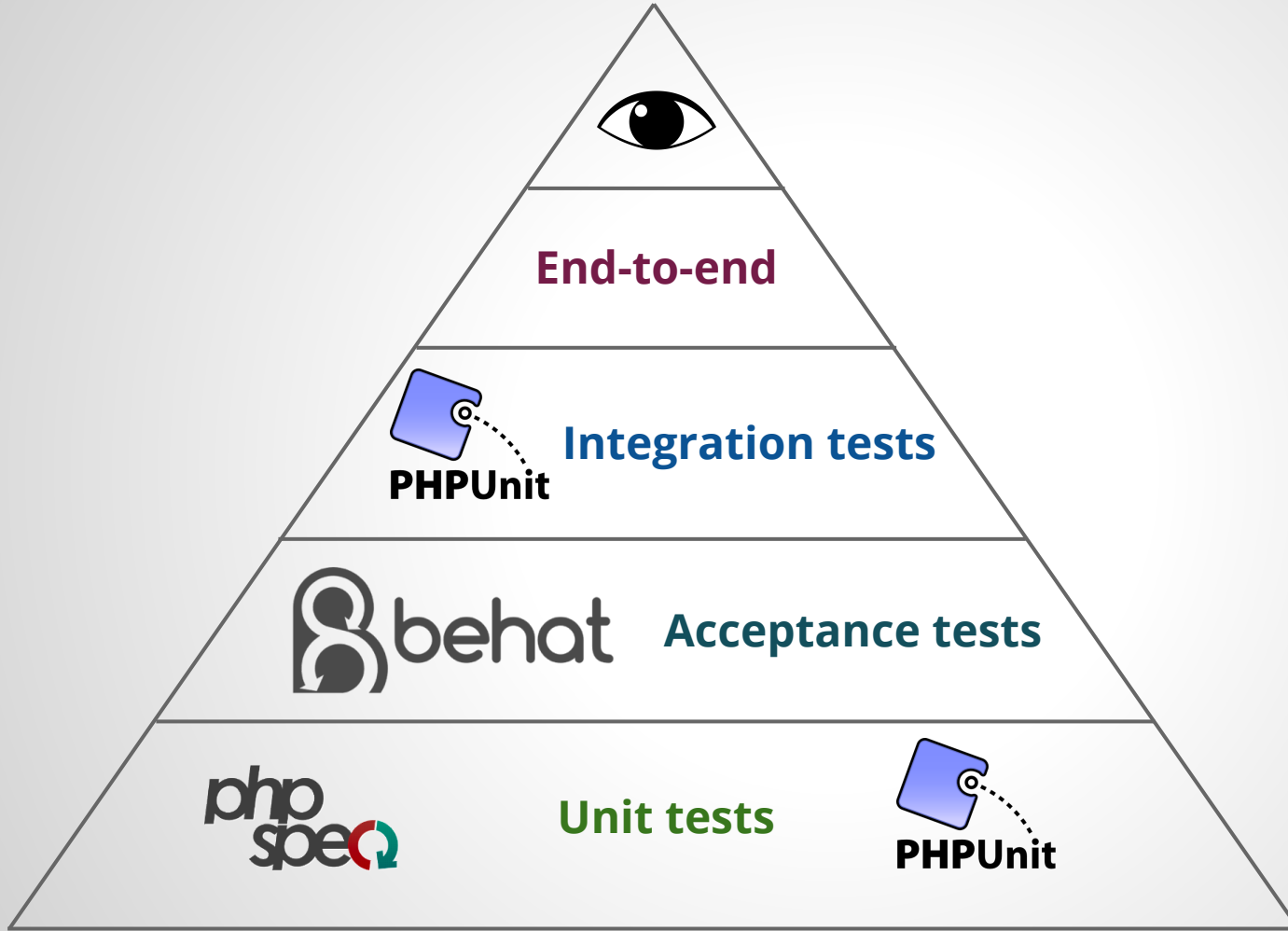
Acceptance tests

Unit tests

Automated

Manual





Slowest



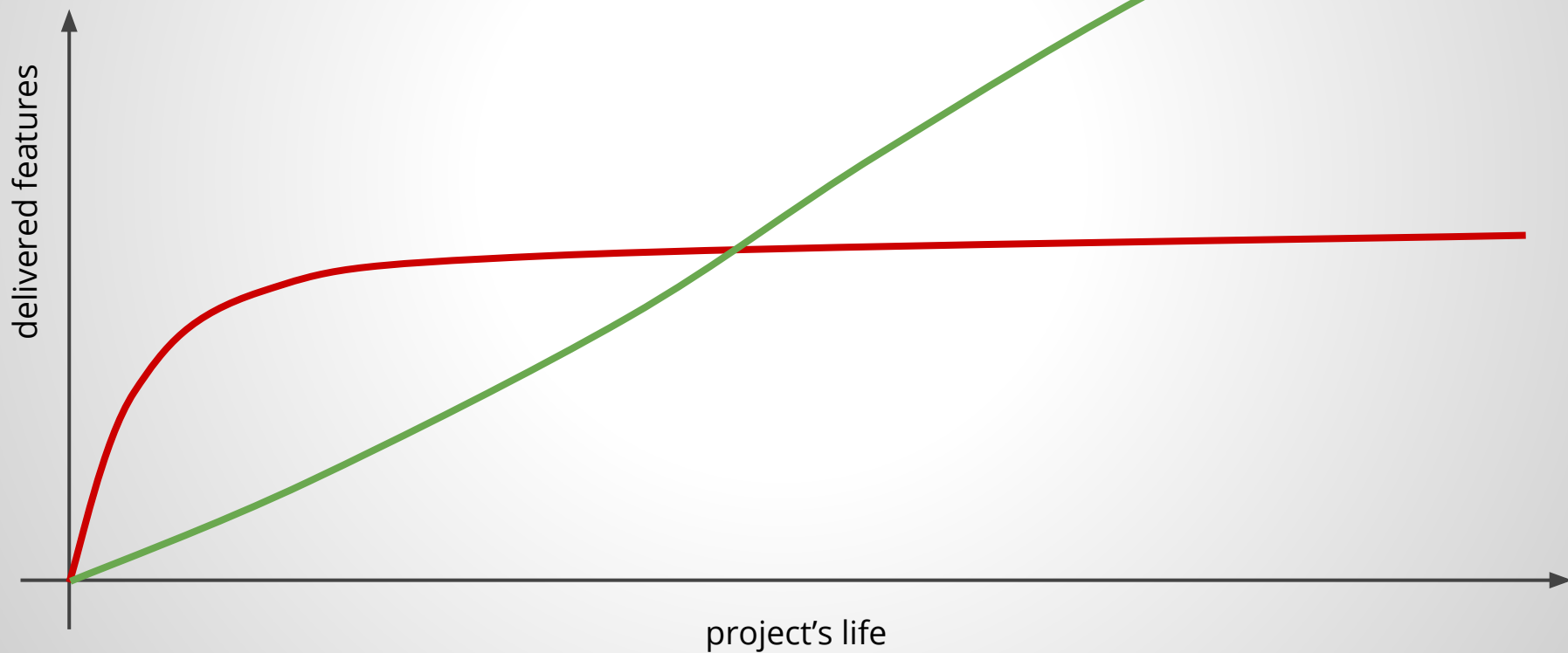
Fastest

Why test?

Why test?

- code level documentation
- features described through examples
- clean code
- you're not afraid to change your code
- peace of mind

Why test?



**Why use
Symfony
components?**

Symfony Components

The standard foundation on which the best PHP applications are built.

- fully tested
- decoupled
- reusable
- de-facto standard

Symfony Components

Build a micro
framework.

Use components
you need
in your application.

Composer

standardised way to install components



Dependency Manager for PHP

[Getting Started](#)[Download](#)[Documentation](#)[Browse Packages](#)[Issues](#)[GitHub](#)

Authors: [Nils Adermann](#), [Jordi Boggiano](#) and many [community contributions](#)

Sponsored by:



composer.json

```
{
    "require": {
        "php": ">=5.6.0"
    },
    "require-dev": {
        "phpspec/phpspec": "~2.0"
    },
    "config": {
        "bin-dir": "bin"
    },
    "autoload": {"psr-0": {"": "src"}}
}
```

```
$ sudo dnf install composer
```

```
$ curl -sS https://getcomposer.org/installer | php
```

```
$ composer install
```

\$ composer update

```
$ composer update package/name
```

```
$ composer require package/name
```

Dependency Injection

symfony/dependency-injection

Dependency Injection

*In software engineering, **dependency injection** is a software design pattern that implements **inversion of control** for resolving dependencies. Dependency injection means giving an object its instance variables. Really. That's it.*

https://en.wikipedia.org/wiki/Dependency_injection

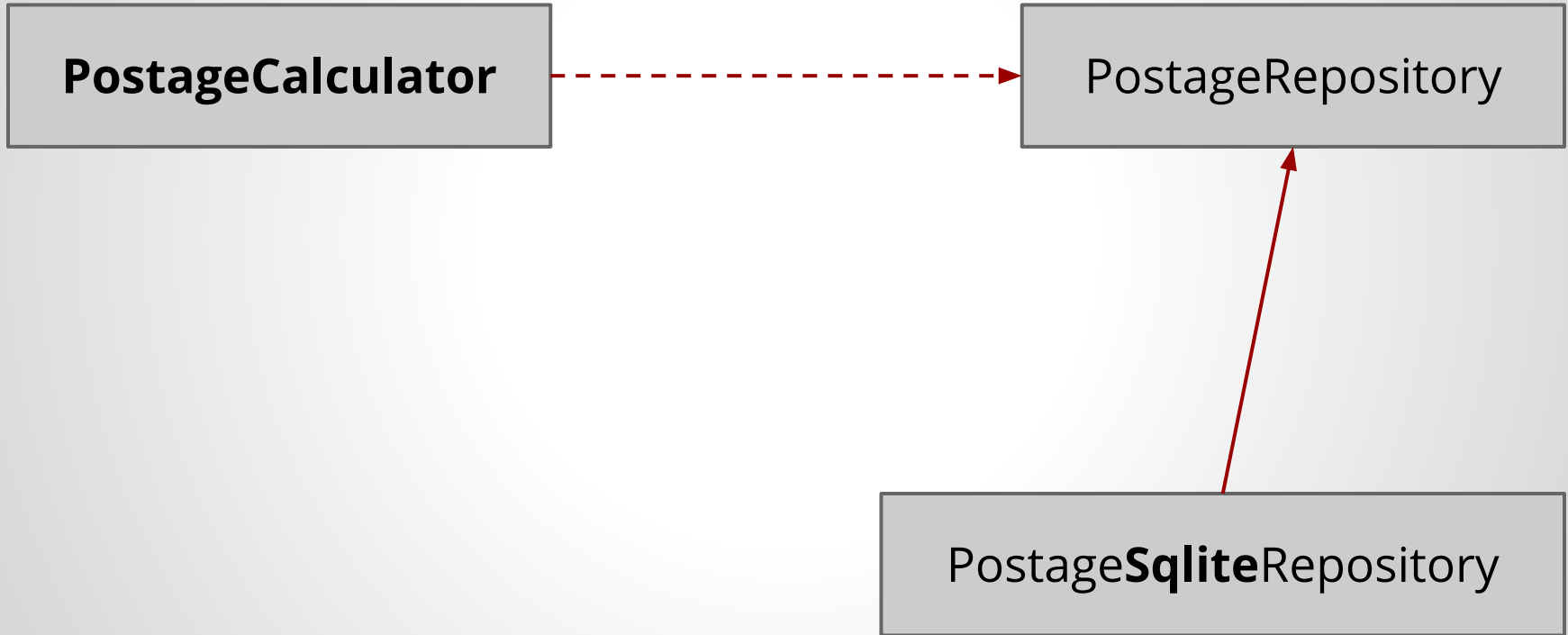
Dependency Injection



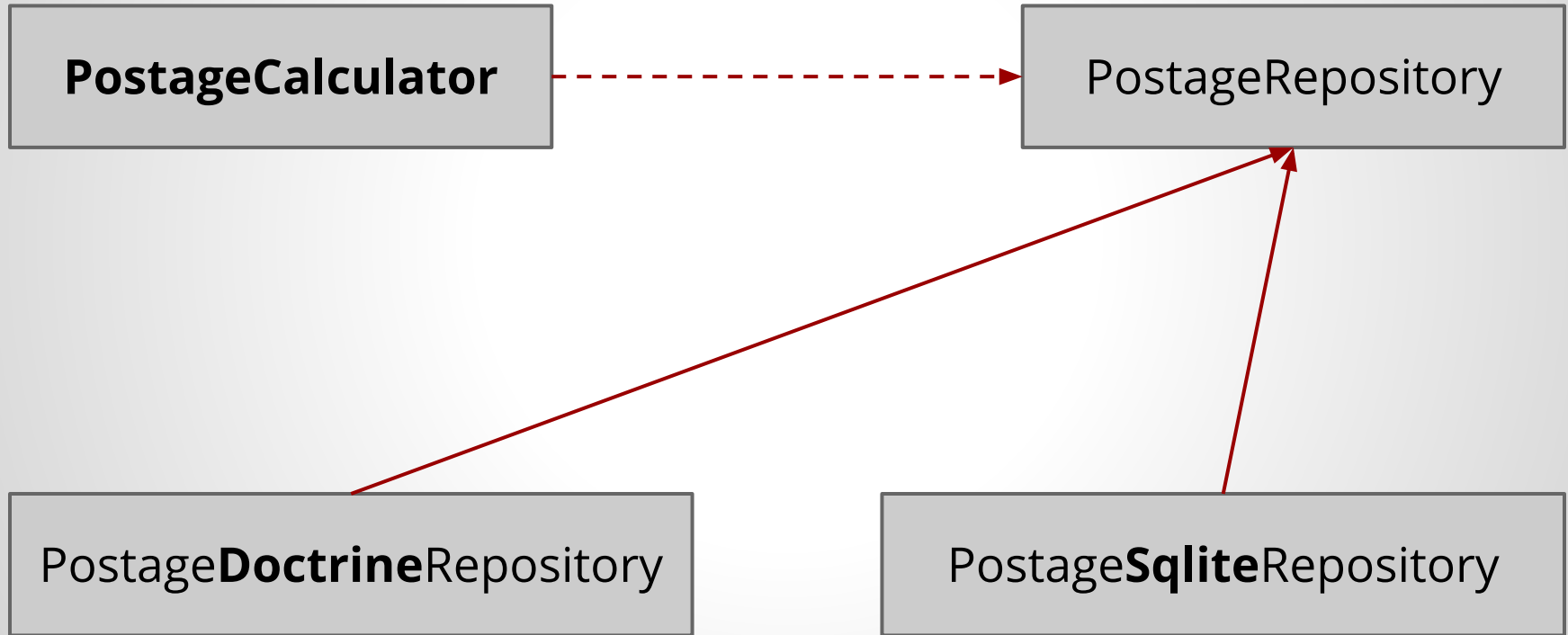
Dependency Inversion



Dependency Inversion



Dependency Inversion



Install dependency injection

```
$ composer require symfony/dependency-injection
```

```
$ composer require symfony/config
```

Create container.php

```
<?php
// app/container.php

use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\DependencyInjection\Loader\YamlFileLoader;
use Symfony\Component\Config\FileLocator;

require __DIR__ . '/../vendor/autoload.php';

$container = new ContainerBuilder();

$loader = new YamlFileLoader($container,
    new FileLocator(__DIR__ . '/app/config'));
$loader->load('services.yml');

$container->compile();
```

Create container.php

```
<?php
// app/container.php

use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\DependencyInjection\Loader\YamlFileLoader;
use Symfony\Component\Config\FileLocator;

require __DIR__ . '/../vendor/autoload.php';

$container = new ContainerBuilder();

$loader = new YamlFileLoader($container,
    new FileLocator(__DIR__ . '/app/config'));
$loader->load('services.yml');

$container->compile();
```

Create container.php

```
<?php
// app/container.php

use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\DependencyInjection\Loader\YamlFileLoader;
use Symfony\Component\Config\FileLocator;

require __DIR__ . '/../vendor/autoload.php';

$container = new ContainerBuilder();

$loader = new YamlFileLoader($container,
    new FileLocator(__DIR__ . '/app/config'));
$loader->load('services.yml');

$container->compile();
```

Create container.php

```
<?php
// app/container.php

use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\DependencyInjection\Loader\YamlFileLoader;
use Symfony\Component\Config\FileLocator;

require __DIR__ . '/../vendor/autoload.php';

$container = new ContainerBuilder();

$loader = new YamlFileLoader($container,
    new FileLocator(__DIR__ . '/app/config'));
$loader->load('services.yml');

$container->compile();
```

Create container.php

```
<?php
// app/container.php

use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\DependencyInjection\Loader\YamlFileLoader;
use Symfony\Component\Config\FileLocator;

require __DIR__ . '/../vendor/autoload.php';

$container = new ContainerBuilder();

$loader = new YamlFileLoader($container,
    new FileLocator(__DIR__ . '/app/config'));
$loader->load('services.yml');

$container->compile();
```

Create container.php

```
<?php
// app/container.php

use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\DependencyInjection\Loader\YamlFileLoader;
use Symfony\Component\Config\FileLocator;

require __DIR__ . '/../vendor/autoload.php';

$container = new ContainerBuilder();

$loader = new YamlFileLoader($container,
    new FileLocator(__DIR__ . '/app/config'));
$loader->load('services.yml');

$container->compile();
```

Create services.yml

```
# app/config/services.yml
```

```
services:
```

```
    acme.postage.repository:
```

```
        class: Acme\Infrastructure\Sqlite\PostageSqliteRepository
```

```
    acme.postage.calculator:
```

```
        class: Acme\Basket\PostageCalculator
```

```
        arguments:
```

```
            - @acme.postage.repository
```

Create services.yml

```
# app/config/services.yml
```

```
services:
```

```
    acme.postage.repository:  
        class: Acme\Infrastructure\Sqlite\PostageSqliteRepository
```

```
    acme.postage.calculator:  
        class: Acme\Basket\PostageCalculator  
        arguments:  
            - @acme.postage.repository
```

Create services.yml

```
# app/config/services.yml
```

```
services:
```

```
    acme.postage.repository:  
        class: Acme\Infrastructure\Sqlite\PostageSqliteRepository
```

```
    acme.postage.calculator:  
        class: Acme\Basket\PostageCalculator  
        arguments:  
            - @acme.postage.repository
```

Use container

```
require '../_inc/db.php';  
require '../app/container.php';
```

```
// ...
```

```
$postageCalculator = $container->get('acme.postage.calculator');
```

```
if ($postage = $postageCalculator->calculateForBasket($_SESSION['basket'])) {  
    $postage = $postage->toView();  
    echo sprintf("<p>Estimated postage for %d item(s): £%1.02f.</p>",  
        $postage['quantity'],  
        $postage['price']  
    );  
}
```

```
// ...
```

Use container

```
require '../_inc/db.php';
require '../app/container.php';

// ...

$postageCalculator = $container->get('acme.postage.calculator');

if ($postage = $postageCalculator->calculateForBasket($_SESSION['basket'])) {
    $postage = $postage->toView();
    echo sprintf("<p>Estimated postage for %d item(s): £%1.02f.</p>",
        $postage['quantity'],
        $postage['price']
    );
}

// ...
```

Use container

```
require '../_inc/db.php';  
require '../app/container.php';
```

```
// ...
```

```
$postageCalculator = $container->get('acme.postage.calculator');
```

```
if ($postage = $postageCalculator->calculateForBasket($_SESSION['basket'])) {  
    $postage = $postage->toView();  
    echo sprintf("<p>Estimated postage for %d item(s): £%1.02f.</p>",  
        $postage['quantity'],  
        $postage['price']  
    );  
}
```

```
// ...
```

Airline postcards

Basket

Product	Price	Quantity	Gross	
Virgin Atlantic Airbus A340	£5.95	1	£5.95	remove
JAL Boeing 777	£7.95	2	£15.90	remove
British Airways Boeing 777	£7.95	1	£7.95	remove
Postage & packing:			£3.95	
Total:			£33.75	

[Return to store](#) or [continue to checkout](#).

© 2005 Airline postcards

Application Kernel

container.php

```
<?php
// app/container.php

use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\DependencyInjection\Loader\YamlFileLoader;
use Symfony\Component\Config\FileLocator;

require __DIR__ . '/../vendor/autoload.php';

$container = new ContainerBuilder();

$loader = new YamlFileLoader($container,
    new FileLocator(__DIR__ . '/app/config'));
$loader->load('services.yml');

$container->compile();
```

container.php

```
<?php
// app/container.php

use Symfony\Component\DependencyInjection\ContainerBuilder;
use Symfony\Component\DependencyInjection\Loader\YamlFileLoader;
use Symfony\Component\Config\FileLocator;

require __DIR__ . '/../vendor/autoload.php';

$container = new ContainerBuilder();

$loader = new YamlFileLoader($container,
    new FileLocator(__DIR__ . '/app/config'));
$loader->load('services.yml');

$container->compile();
```

```
<?php # src\Acme\Application\Kernel.php

namespace Acme\Application;

class Kernel
{
    /** @var ContainerBuilder */
    private $container;

    public function boot()
    {
        $this->container = new ContainerBuilder();

        $loader = new YamlFileLoader($container,
            new FileLocator(__DIR__ . '/app/config'));
        $loader->load('services.yml');

        $this->container->compile();
    }

    public function getContainer()
    {
        return $this->container;
    }
}
```

container.php

```
<?php
// app/container.php

require __DIR__ . '/../vendor/autoload.php';

$kernel = (new \Acme\Application\Kernel())->boot();

$container = $kernel->getContainer();
```

Singleton Kernel

```
<?php # src\Acme\Application\Kernel.php

namespace Acme\Application;

class Kernel
{
    public static $instance;

    // ...

    public static getInstance()
    {
        if (null === static::$instance) {
            static::$instance = (new static())->boot();
        }

        return static::$instance;
    }

    protected function __construct() {}

    private function __clone() {}
    private function __wakeup() {}
}
```

```
<?php # src\Acme\Application\Kernel.php
```

```
namespace Acme\Application;
```

```
final class Kernel
```

```
{
```

```
    public static $instance;
```

```
    // ...
```

```
    public static getInstance()
```

```
    {
```

```
        if (null === static::$instance) {  
            static::$instance = (new static())->boot();  
        }
```

```
        return static::$instance;
```

```
    }
```

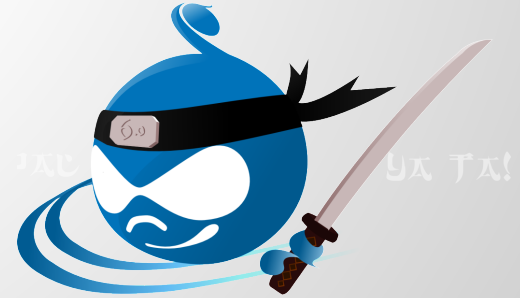
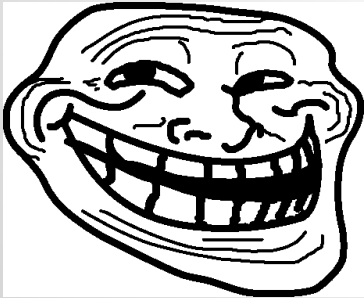
```
    protected function __construct() {}
```

```
    private function __clone() {}
```

```
    private function __wakeup() {}
```

```
}
```

Dependency Injection with Drupal 7?



container.php

```
<?php
```

```
require_once DRUPAL_ROOT . '/../vendor/autoload.php';
```

```
$kernel = (new \Acme\Application\Kernel())->boot();
```

```
variable_set('di_container', $kernel->getContainer());
```

container.php

```
<?php
```

```
require_once DRUPAL_ROOT . '/../vendor/autoload.php';
```

```
$kernel = (new \Acme\Application\Kernel())->boot();
```

```
variable_set('di_container', $kernel->getContainer());
```

public/sites/default/settings.php

```
<?php
```

```
// ...
```

```
include_once 'config/container.php';  
include 'database.php';
```

```
// ...
```

depot_locator.module

```
<?php
```

```
function get_depot_json($postcode)
{
    $depotEntity = variable_get('di_container')
        ->get('acme.depot_locator')->locateDepot($postcode);

    $depot = $depotEntity->toArray();

    set_depot_cookies($depot);

    return ['depot' => $depot];
}
```

depot_locator.module

```
<?php
```

```
function get_depot_json($postcode)
{
    $depotEntity = variable_get('di_container')
        ->get('acme.depot_locator')->locateDepot($postcode);

    $depot = $depotEntity->toArray();

    set_depot_cookies($depot);

    return ['depot' => $depot];
}
```

Add console commands

symfony/console

Install console component

```
$ composer require symfony/console
```

```
class LocateDepotCommand extends Command
{
    /** @var DepotLocator */
    private $depotLocator;

    public function __construct(DepotLocator $depotLocator)
    {
        $this->depotLocator = $depotLocator;

        parent::__construct();
    }

    protected function configure()
    {
        $this->setName('depot:locate')
            ->addArgument('postcode', InputArgument::REQUIRED);
    }

    protected function execute(InputInterface $input, OutputInterface $output)
    {
        $this->depotLocator->locateDepot($input->getArgument('postcode'));

        $output->writeln('Nearest depot is located at: ' . $depot->getAddress());
    }
}
```

app/console

```
#!/usr/bin/php
<?php

require_once (__DIR__ . '/../vendor/autoload.php');
require_once (__DIR__ . '/app/container.php');

use Symfony\Component\Console\Application;

$application = new Application();
$application->addCommands([
    $container->get('acme.command.depot_locate'),
]);

$application->run();
```

```
$ app/console depot:locate wc1a1dg
```

Twig

twig/twig

What is twig?

a SENSIO LABS * product



The flexible, fast, and secure template engine for PHP

ABOUT

DOCUMENTATION

DEVELOPMENT

CONTRIBUTORS



Twig is a modern template engine for PHP

- **Fast:** Twig *compiles* templates down to plain optimized PHP code. The overhead compared to regular PHP code was reduced to the very minimum.
- **Secure:** Twig has a *sandbox* mode to evaluate untrusted template code. This allows Twig to be used as a template language for applications where users may modify the template design.
- **Flexible:** Twig is powered by a flexible *lexer* and *parser*. This allows the developer to define its own custom tags and filters, and create its own DSL.

LEARN MORE →

INSTALL NOW ↓

Stable: 1.21.0

Why yet another template engine?

Airline postcards

Basket

Product	Price	Quantity	Gross	
Virgin Atlantic Airbus A340	£5.95	1	£5.95	remove
JAL Boeing 777	£7.95	2	£15.90	remove
British Airways Boeing 777	£7.95	1	£7.95	remove
Postage & packing:			£3.95	
Total:			£33.75	

[Return to store](#) or [continue to checkout](#).

© 2005 Airline postcards

Airline postcards

Place an order

Products

You are about to order the following products:

Product	Price	Quantity	Gross
Virgin Atlantic Airbus A340	£5.95	1	£5.95
JAL Boeing 777	£7.95	3	£23.85
British Airways Boeing 777	£7.95	1	£7.95
Postage & packing:			£4.95
Total:			£42.70

Shipping address

Name:

Address:

City:

Install twig

```
$ composer require twig/twig
```

How to use twig?

```
<?php
```

```
$loader = new Twig_Loader_Filesystem('/path/to/templates');
```

```
$twig = new Twig_Environment($loader);
```

```
echo $twig->render('index.html', ['variable' => 'value']);
```

Enabling twig in DI container

```
# app/config/services.yml
```

```
parameters:
```

```
    twig.paths:
```

```
        - app/Resources/views
```

```
services:
```

```
    twig.template_loader:
```

```
        class: Twig_Loader_Filesystem
```

```
        arguments:
```

```
            - %twig.paths%
```

```
    twig.renderer:
```

```
        class: Twig_Environment
```

```
        arguments:
```

```
            - @twig.template_loader
```

Enabling twig in DI container

```
# app/config/services.yml
```

```
parameters:
    twig.paths:
        - app/Resources/views
```

```
services:
    twig.template_loader:
        class: Twig_Loader_Filesystem
        arguments:
            - %twig.paths%

    twig.renderer:
        class: Twig_Environment
        arguments:
            - @twig.template_loader
```

Enabling twig in DI container

```
# app/config/services.yml
```

```
parameters:
```

```
    twig.paths:
```

```
        - app/Resources/views
```

```
services:
```

```
    twig.template_loader:
```

```
        class: Twig_Loader_Filesystem
```

```
        arguments:
```

```
            - %twig.paths%
```

```
    twig.renderer:
```

```
        class: Twig_Environment
```

```
        arguments:
```

```
            - @twig.template_loader
```

Enabling twig in DI container

```
# app/config/services.yml
```

```
parameters:
```

```
    twig.paths:
```

```
        - app/Resources/views
```

```
services:
```

```
    twig.template_loader:
```

```
        class: Twig_Loader_Filesystem
```

```
        arguments:
```

```
            - %twig.paths%
```

```
    twig.renderer:
```

```
        class: Twig_Environment
```

```
        arguments:
```

```
            - @twig.template_loader
```

Use twig

```
$basketViewData = $container->get('acme.basket.view')  
->toView($_SESSION['basket']);
```

```
$basketView = $container->get('twig.renderer')  
->render(  
    'basket.twig.html',  
    $basketViewData  
);
```

```
echo $basketView;
```

Decouple from global state

symfony/http-foundation

Http application's flow



Install HttpFoundation component

```
$ composer require symfony/http-foundation
```

De-coupling from global state

```
# /public/index.php
<?php

require __DIR__ . '/../app/container.php';

use Symfony\Component\HttpFoundation\Request;

$request = Request::createFromGlobals();

$response = $container->get('acme.basket_view.controller')
    ->viewAction($request);

$response->send();
```

De-coupling from global state

```
# /public/index.php  
<?php
```

```
require __DIR__ . '/../app/container.php';
```

```
use Symfony\Component\HttpFoundation\Request;
```

```
$request = Request::createFromGlobals();
```

```
$response = $container->get('acme.basket_view.controller')  
    ->viewAction($request);
```

```
$response->send();
```

De-coupling from global state

```
# /public/index.php  
<?php
```

```
require __DIR__ . '/../app/container.php';
```

```
use Symfony\Component\HttpFoundation\Request;
```

```
$request = Request::createFromGlobals();
```

```
$response = $container->get('acme.basket_view.controller')  
    ->viewAction($request);
```

```
$response->send();
```

De-coupling from global state

```
# /public/index.php
<?php

require __DIR__ . '/../app/container.php';

use Symfony\Component\HttpFoundation\Request;

$request = Request::createFromGlobals();

$response = $container->get('acme.basket_view.controller')
    ->viewAction($request);

$response->send();
```

Going forward

Try it yourself!

<https://github.com/mareg/legacy-summercamp-2015>

- four exercises:
 - add dependency injection
 - add twig
 - decouple from global state
 - controller as service

Questions?

Slides: speakerdeck.com/super_marek

Feedback: joind.in/talk/view/15383

Connect: [@super_marek](https://twitter.com/super_marek)