Test{able|ing} Symfony Applications

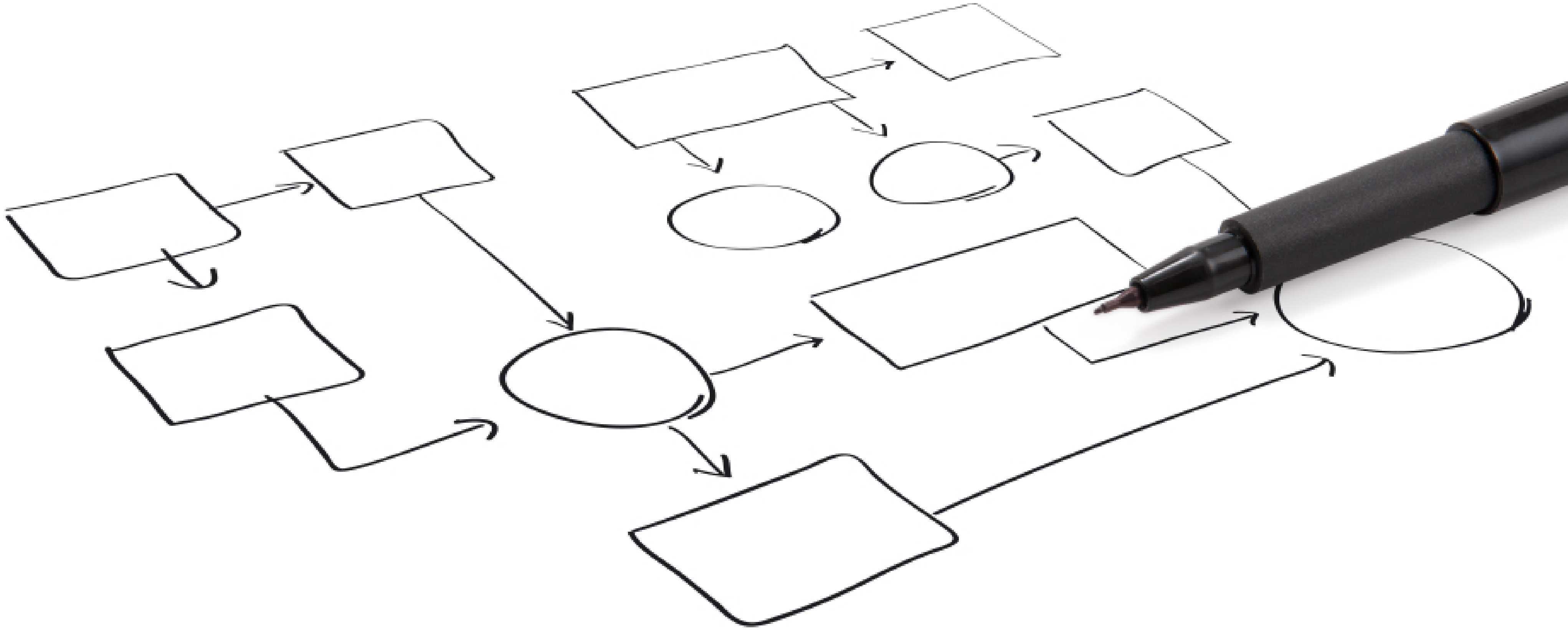
Sebastian Bergmann | September 6th 2013

Sebastian Bergmann

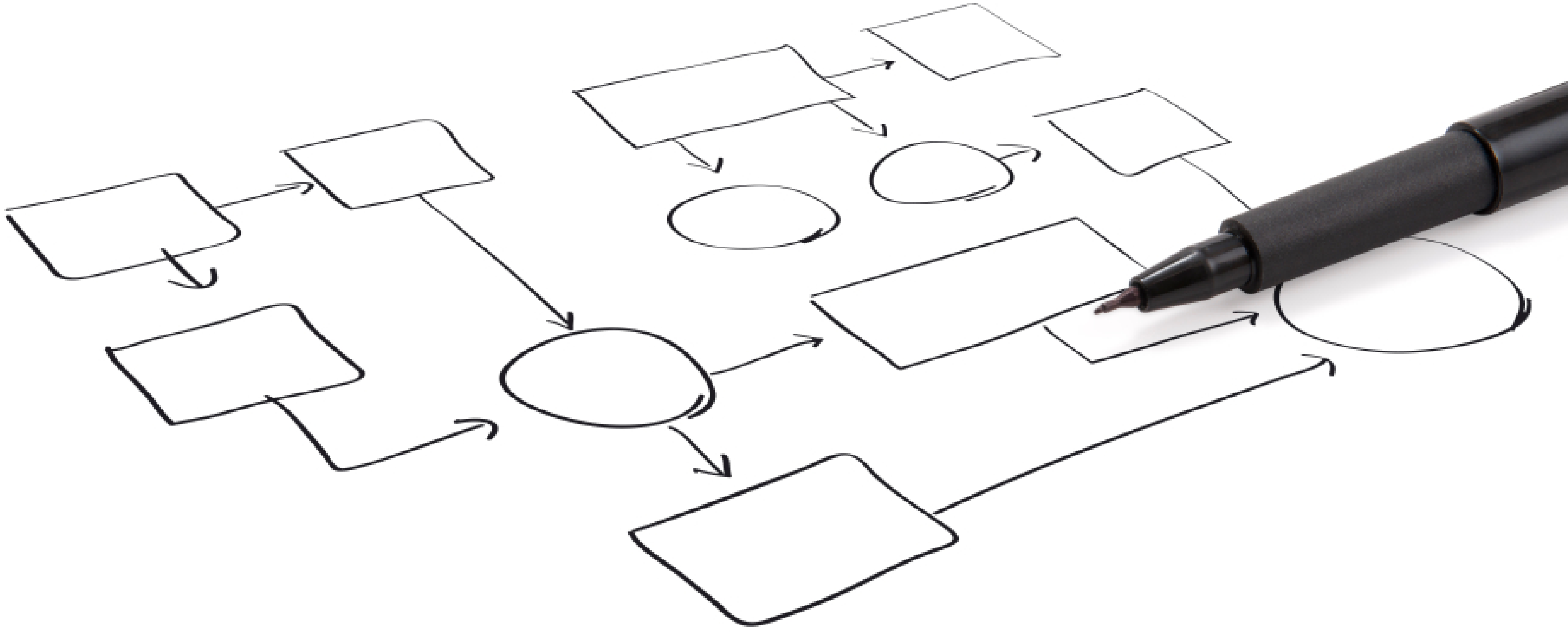
- Has instrumentally contributed to transforming PHP into a reliable platform for large-scale, critical projects.
- Enterprises and PHP developers around the world benefit from the tools that he has developed and the experience he shares.

sharing experience

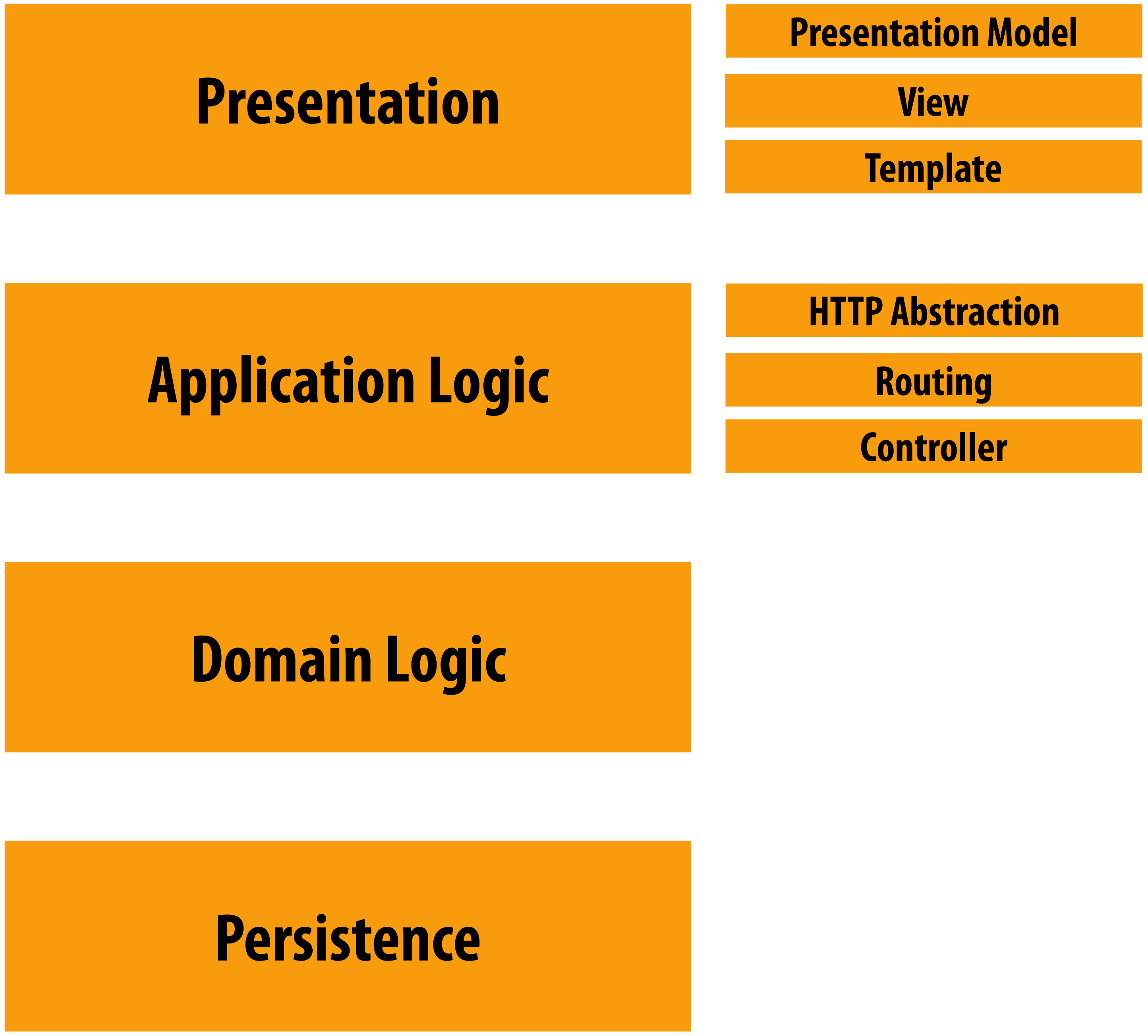
thePHP.cc

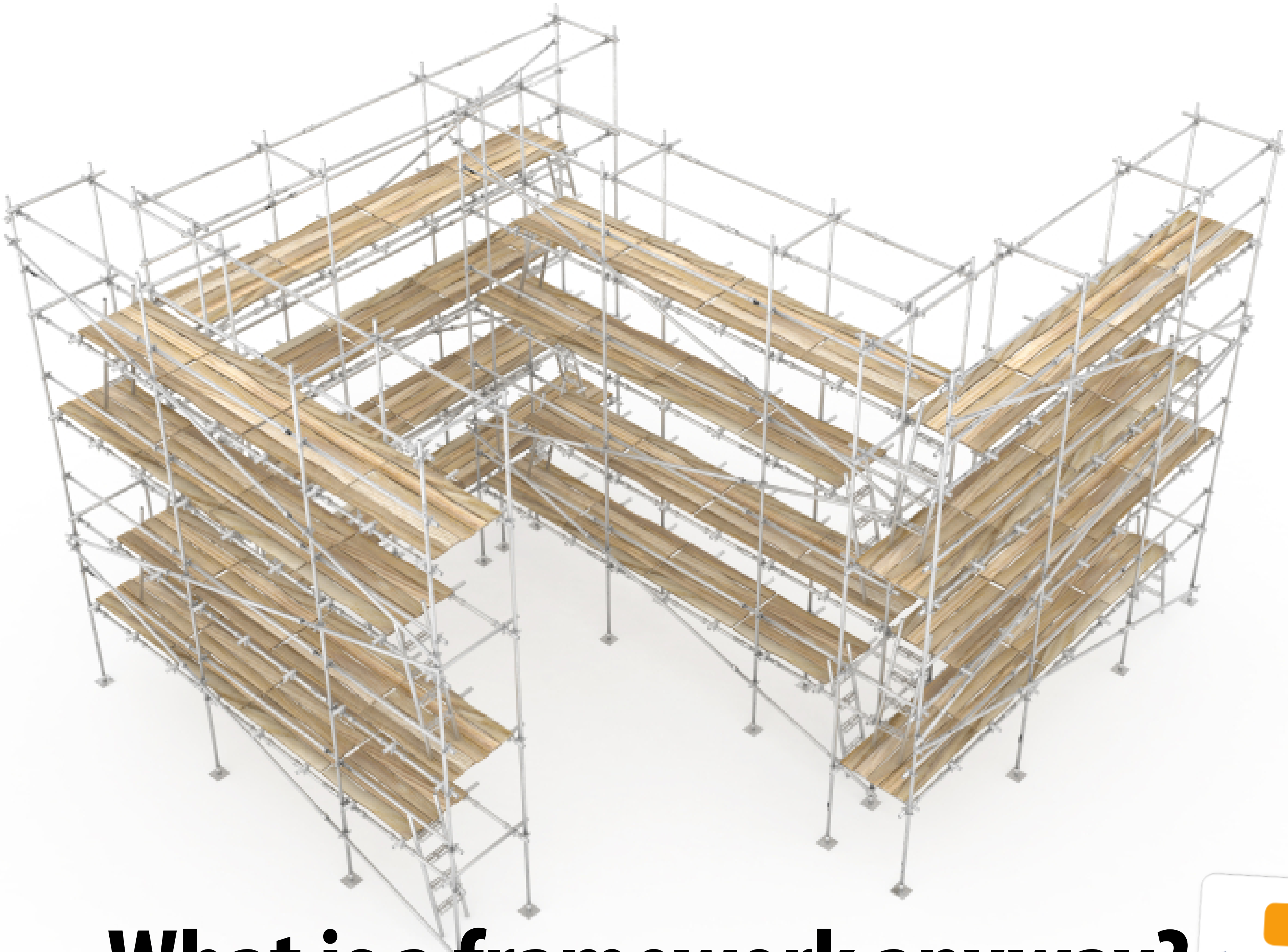


What is a Symfony2 application anyway?

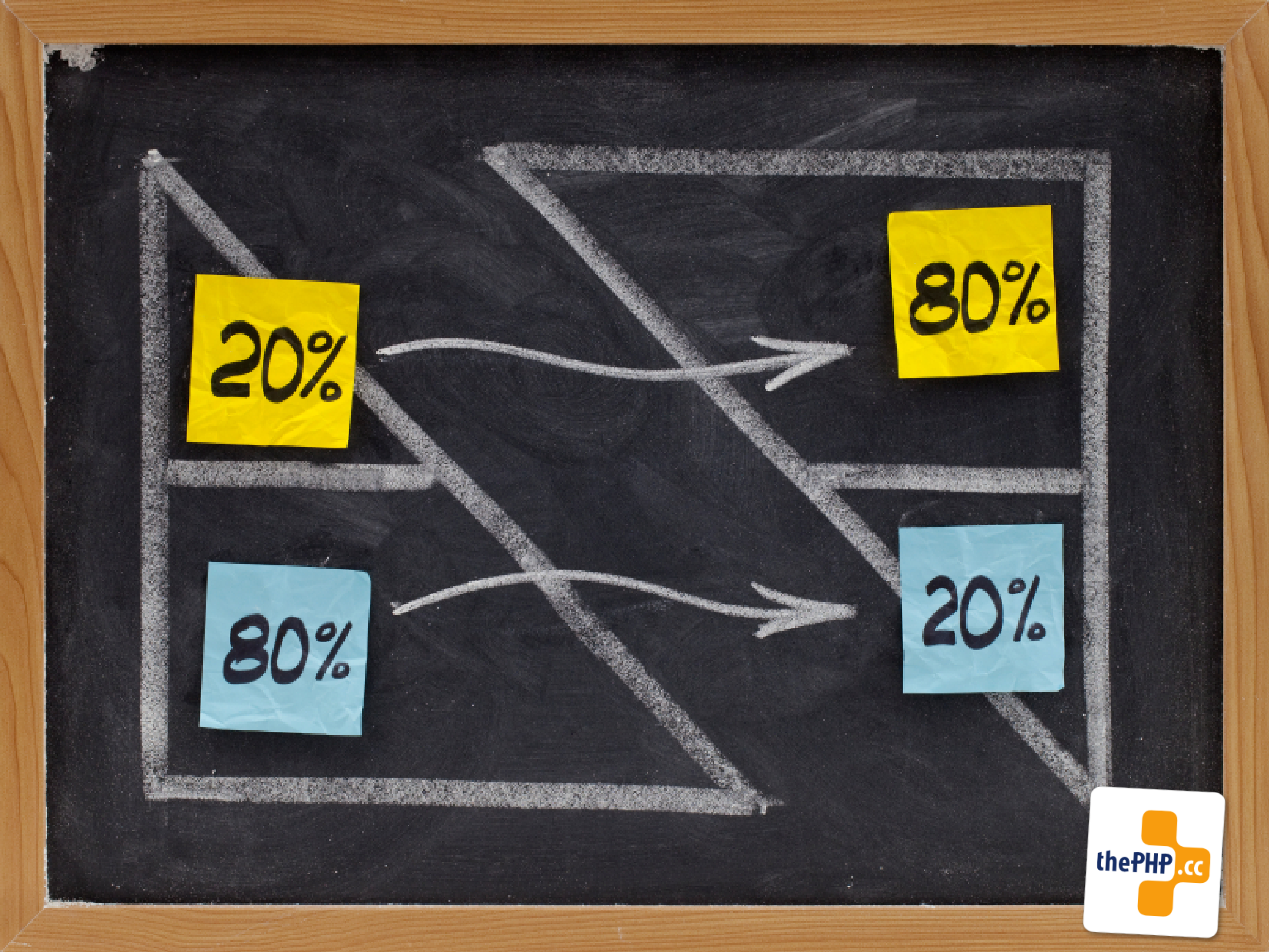


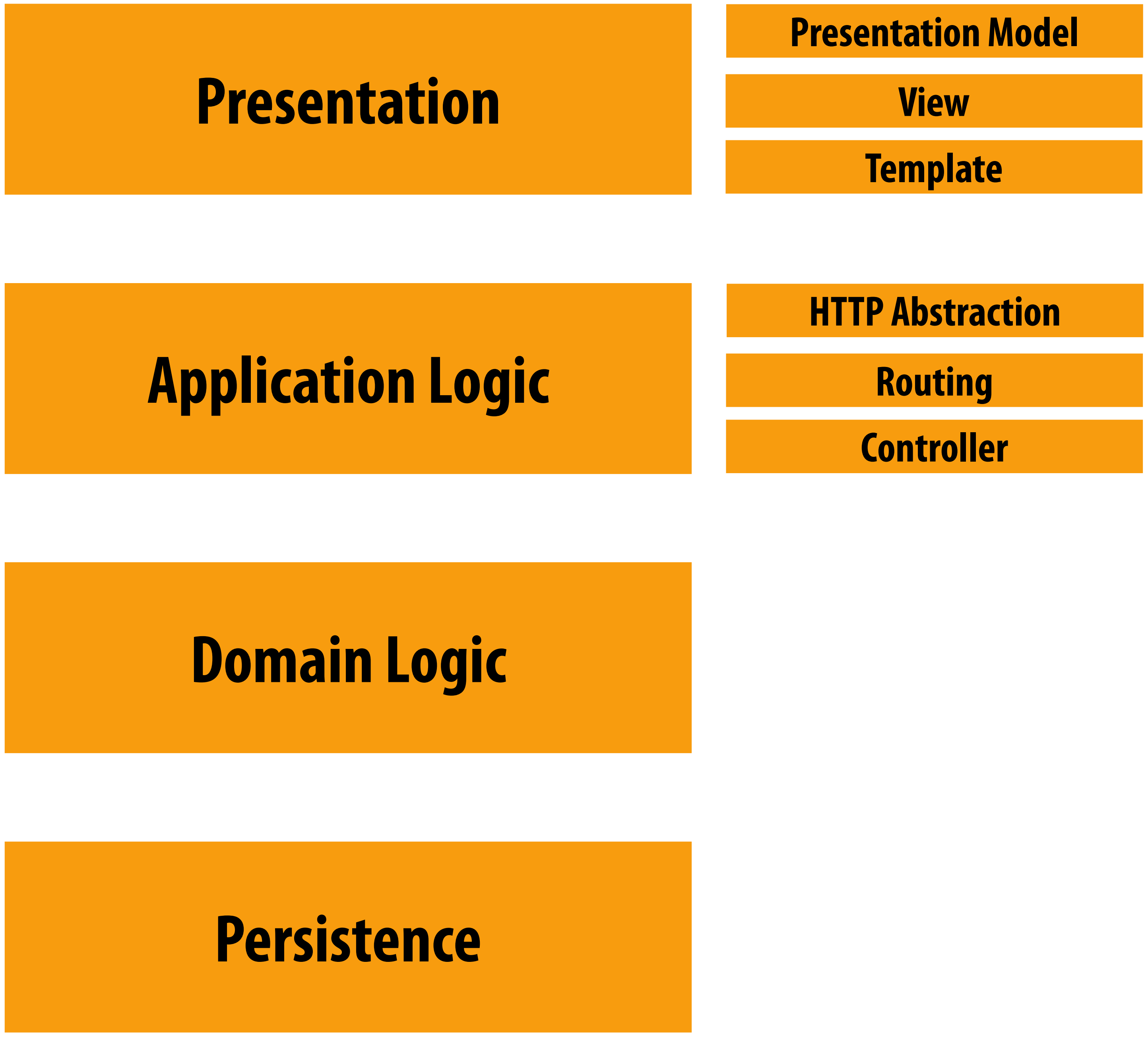
What is a ~~Symfony2~~ web application anyway?

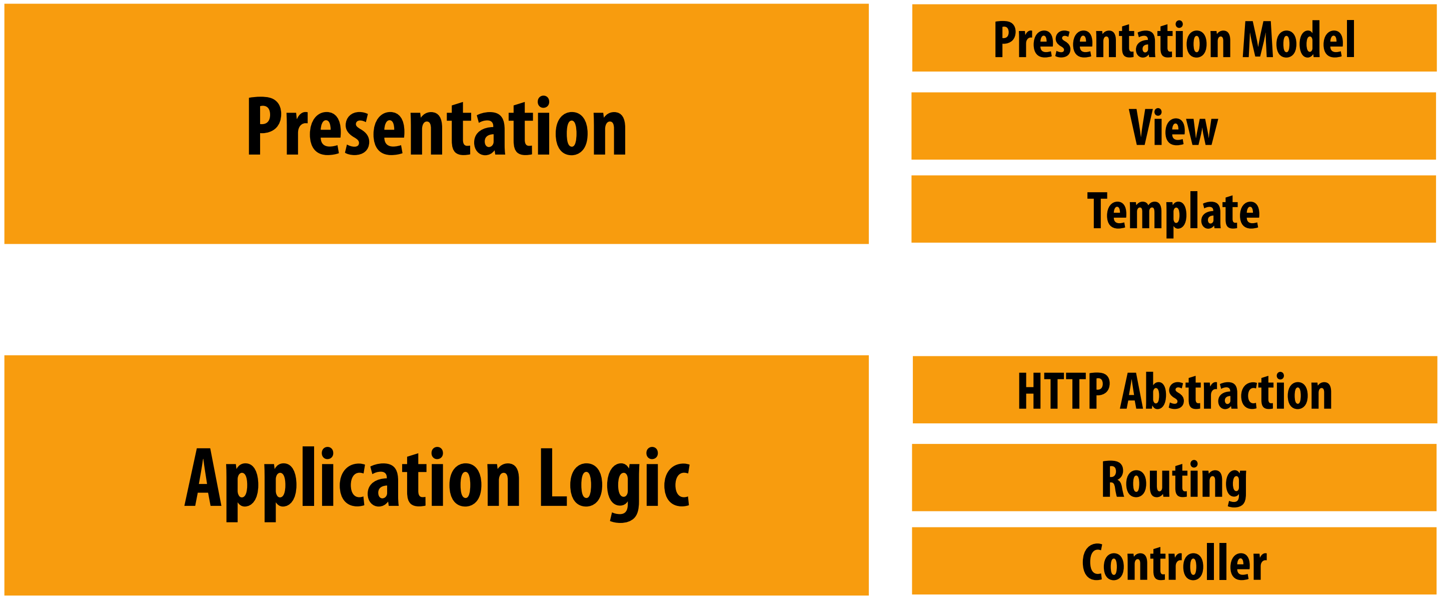

Presentation
Presentation Model
View
Template
Application Logic
HTTP Abstraction
Routing
Controller
Domain Logic
Persistence



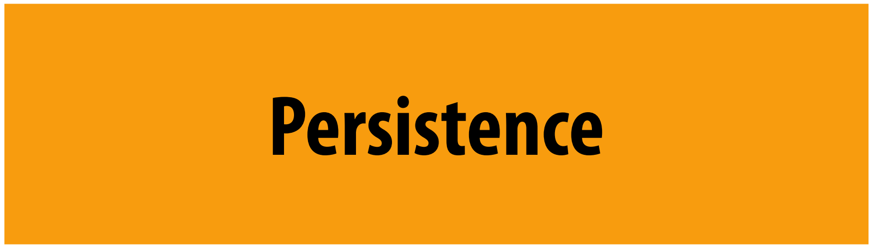
What is a framework anyway?

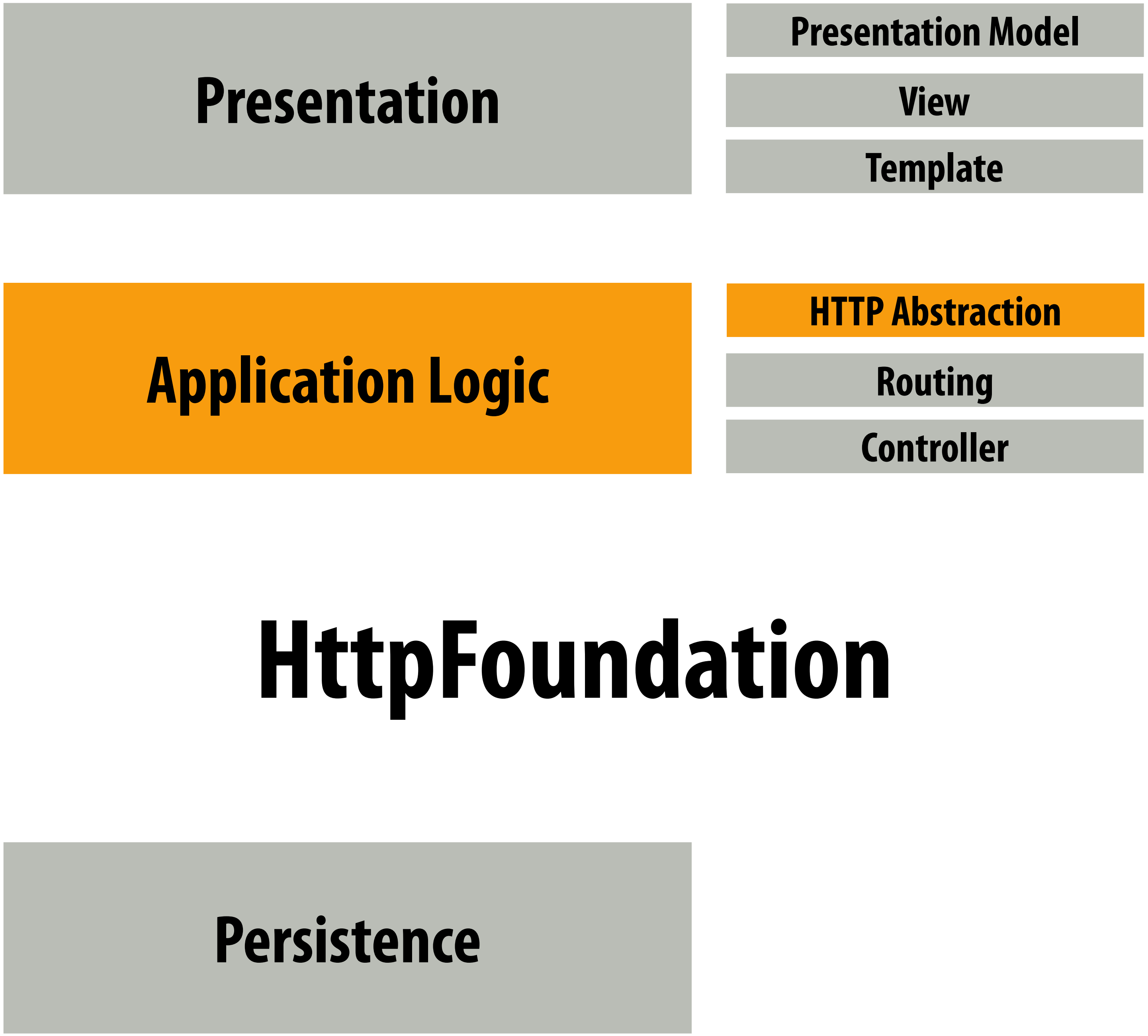

20%
80%
80%
20%


Presentation
Presentation Model
View
Template
Application Logic
HTTP Abstraction
Routing
Controller
Domain Logic
Persistence



A framework can only help with these aspects of a web application.





HttpFoundation

Presentation

Presentation Model

View

Template

Application Logic

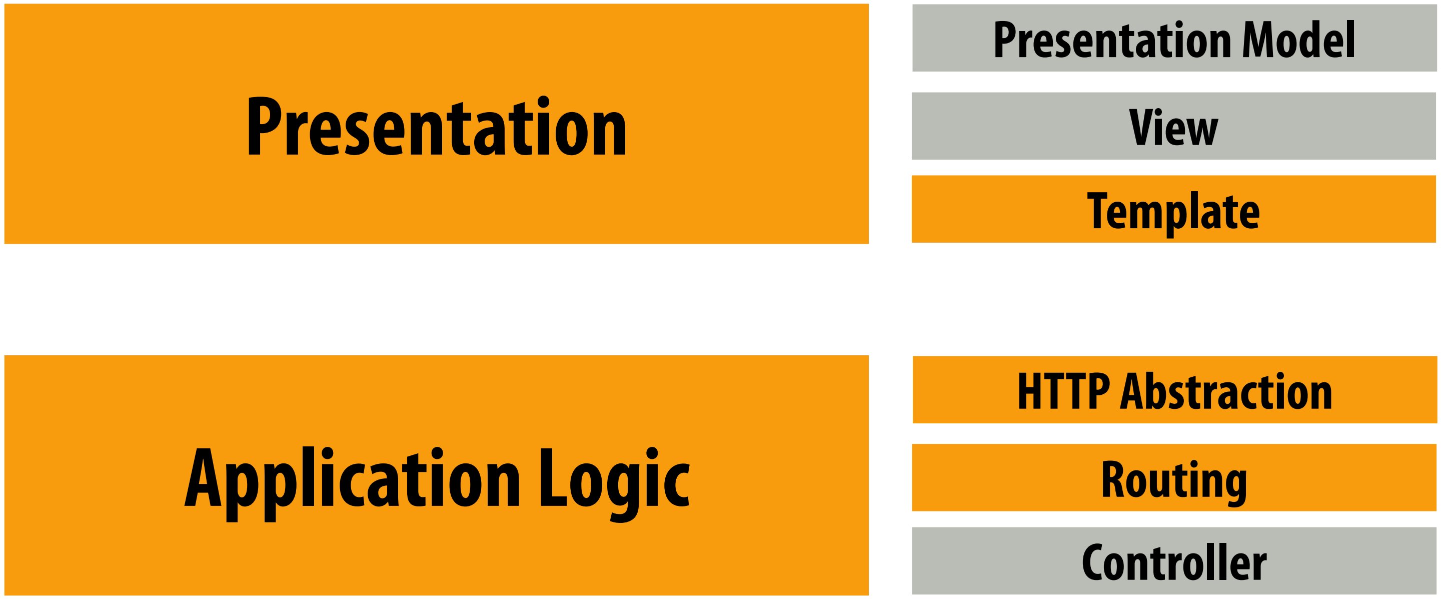
HTTP Abstraction

Routing

Controller

HttpFoundation + Routing

Persistence



HttpFoundation + Routing + Twig

Persistence

```php
<?php
$filename = __DIR__ . preg_replace('#(\?.*)$#', '', $_SERVER['REQUEST_URI']);

if (PHP_SAPI === 'cli-server' && is_file($filename)) {
    return false;
}

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

use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;

$routes      = new RouteCollection;
$application = new Application($routes);
$request     = Request::createFromGlobals();
$response    = new Response;

$application->run($request, $response);
$response->send();
```

```php
<?php
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Matcher\UrlMatcher;
use Symfony\Component\Routing\RequestContext;

class Application
{
    private $routes;

    public function __construct(RouteCollection $routes)
    {
        $this->routes = $routes;
    }

    public function run(Request $request, Response $response)
    {
        $context = new RequestContext;
        $context->fromRequest($request);

        $matcher    = new UrlMatcher($this->routes, $context);
        $parameters = $matcher->match($request->getRequestUri());

        switch ($parameters['controller']) {
            case 'BankAccount': {
                $controller = new BankAccountController;

                $response->setContent($controller->execute($parameters['id']));
            }
            break;
        }
    }
}
```

```php
<?php
use Symfony\Component\Routing\RouteCollection as SymfonyRouteCollection;
use Symfony\Component\Routing\Route;

class RouteCollection extends SymfonyRouteCollection
{
    public function __construct()
    {
        $this->add(
            'bankaccount',
            new Route(
                '/bankaccount/{id}',
                array('controller' => 'BankAccount'),
                array('id' => '[0-9]*')
            )
        );
    }
}
```

```php
<?php
class BankAccountController
{
    public function execute($id)
    {
        $mapper = new BankAccountMapper;
        $view   = new BankAccountView;

        return $view->render($id, $mapper->findById($id));
    }
}
```

```php
<?php
class BankAccountController
{
    public function execute($id)
    {
        $mapper = new BankAccountMapper;
        $view   = new BankAccountView;

        return $view->render($id, $mapper->findById($id));
    }
}
```

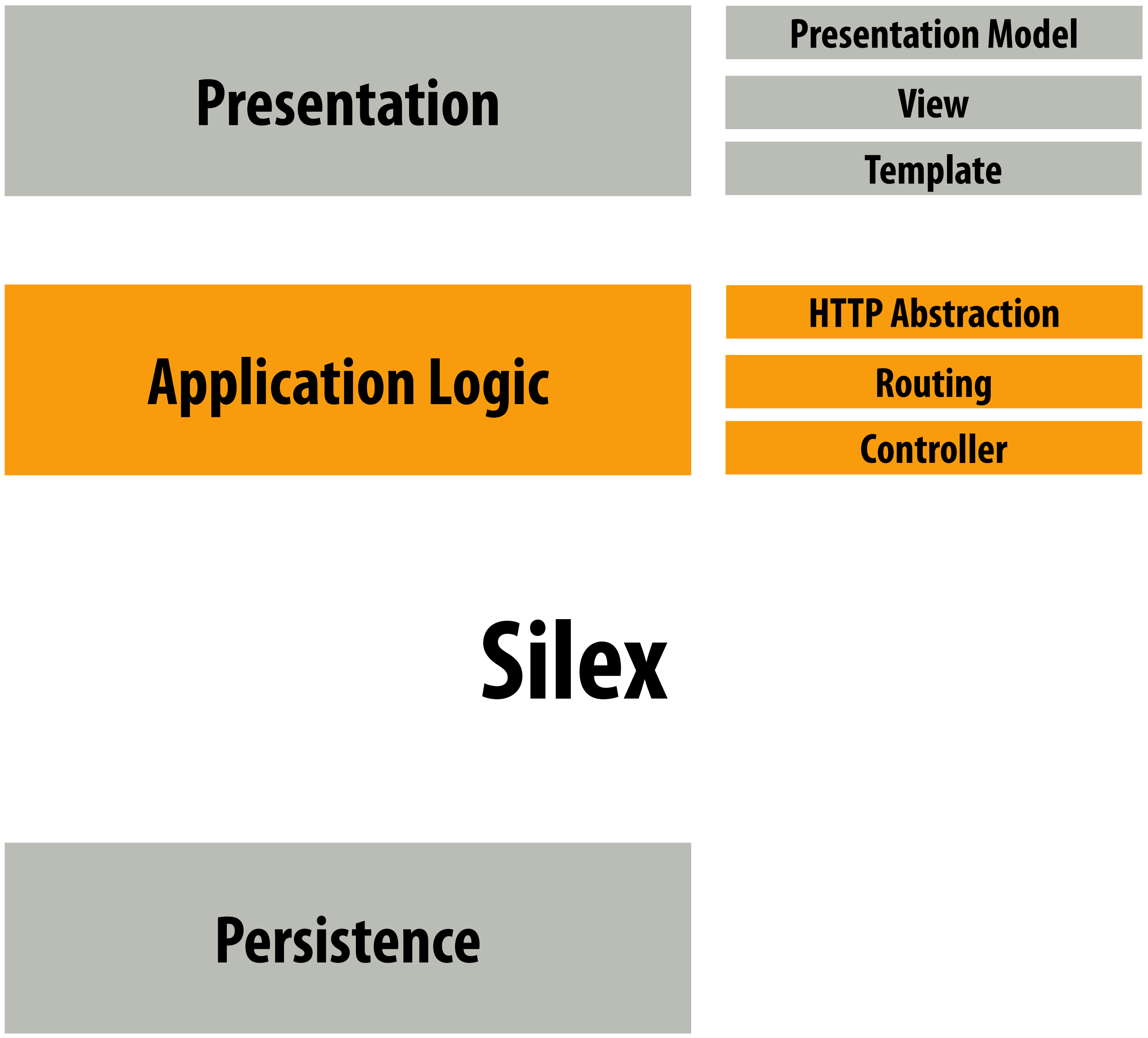
Not injecting these dependencies will haunt us later ...

```php
<?php
use SebastianBergmann\Money\Currency;
use SebastianBergmann\Money\Money;

class BankAccountMapper
{
    public function findById($id)
    {
        $class  = new ReflectionClass('BankAccount');
        $object = $class->newInstanceWithoutConstructor();

        $attribute = new ReflectionProperty($object, 'balance');
        $attribute->setAccessible(true);
        $attribute->setValue(
            $object,
            new Money(100, new Currency('EUR'))
        );

        return $object;
    }
}
```

Simplified data mapper implementation with faked storage

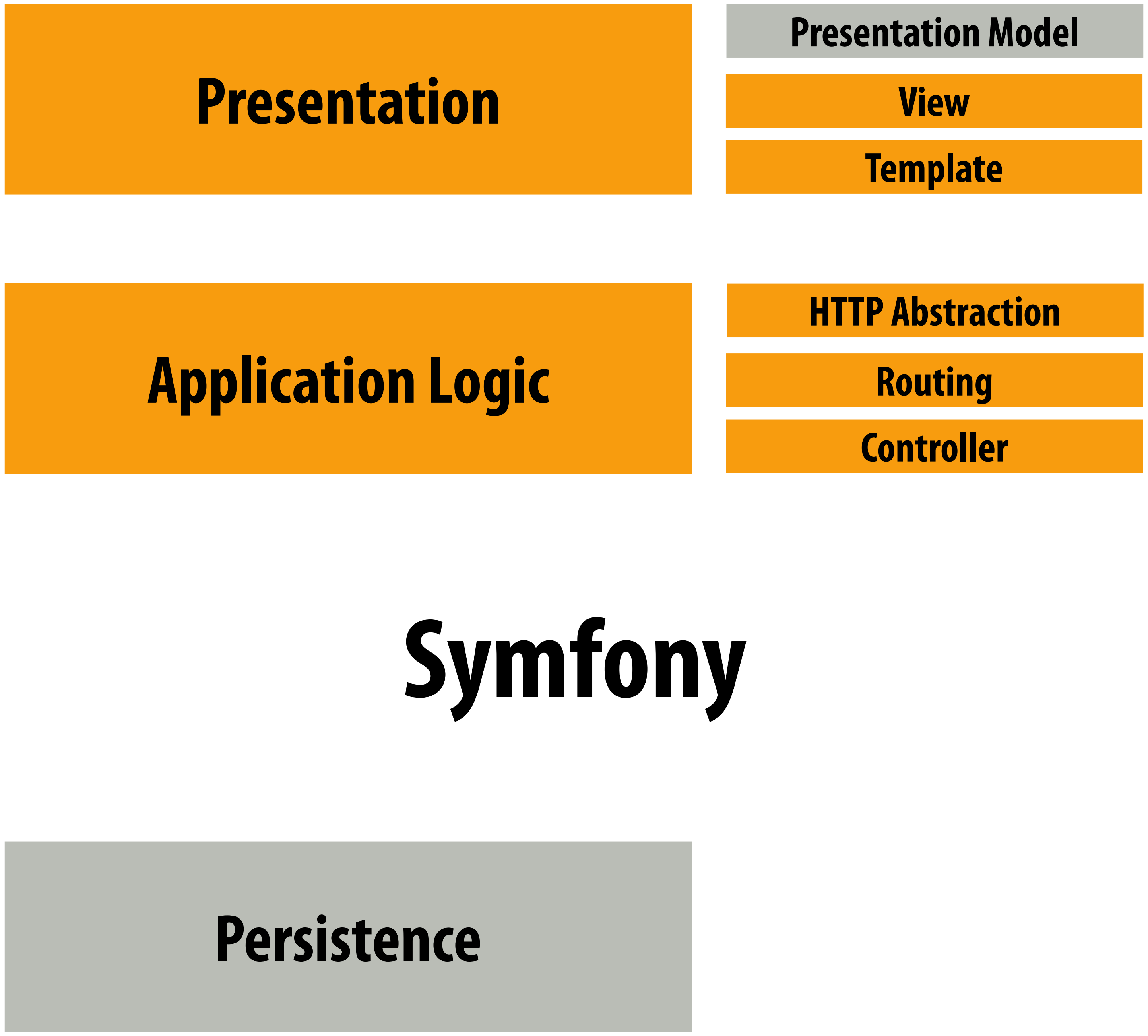
```php
<?php
use SebastianBergmann\Money\IntlFormatter;

class BankAccountView
{
    public function render($id, BankAccount $bankAccount)
    {
        $formatter = new IntlFormatter('de_DE');

        $twig = new Twig_Environment(
            new Twig_Loader_Filesystem(__DIR__ . '/../templates')
        );

        $template = $twig->loadTemplate('bankaccount.html');

        return $template->render(
            array('id'      => $id,
                  'balance' => $formatter->format(
                                   $bankAccount->getBalance()
                               )
            )
        );
    }
}
```

```php
<?php
use SebastianBergmann\Money\Currency;
use SebastianBergmann\Money\Money;

class BankAccount
{
    private $balance;

    public function __construct()
    {
        $this->balance = new Money(0, new Currency('EUR'));
    }

    public function getBalance()
    {
        return $this->balance;
    }
}
```


Presentation
Presentation Model
View
Template
Application Logic
HTTP Abstraction
Routing
Controller
Silex
Persistence

```php
<?php
$filename = __DIR__ . preg_replace('#(\?.*)$#', '',
$_SERVER['REQUEST_URI']);

if (PHP_SAPI === 'cli-server' && is_file($filename)) {
    return false;
}

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

$app = new Application;
$app->run();
```

```php
<?php
use Silex\Application as SilexApplication;

class Application extends SilexApplication
{
    public function __construct(array $values = array())
    {
        parent::__construct($values);

        $this->get(
            '/bankaccount/{id}',
            function ($id)
            {
                $controller = new BankAccountController;
                return $controller->execute($id);
            }
        );
    }
}
```


Presentation
Presentation Model
View
Template
Application Logic
HTTP Abstraction
Routing
Controller
Symfony
Persistence

```php
<?php
namespace SebastianBergmann\BankAccountBundle\Controller;

use SebastianBergmann\Money\IntlFormatter;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;

class DefaultController extends Controller
{
    public function indexAction($id)
    {
        $formatter   = new IntlFormatter('de_DE');
        $mapper      = $this->get('bankaccount_mapper');
        $bankAccount = $mapper->findById($id);

        return $this->render(
            'SebastianBergmannBankAccountBundle:Default:index.html.twig',
            array(
                'id' => $id,
                'balance' => $formatter->format(
                    $bankAccount->getBalance()
                )
            )
        );
    }
}
```

Presentation

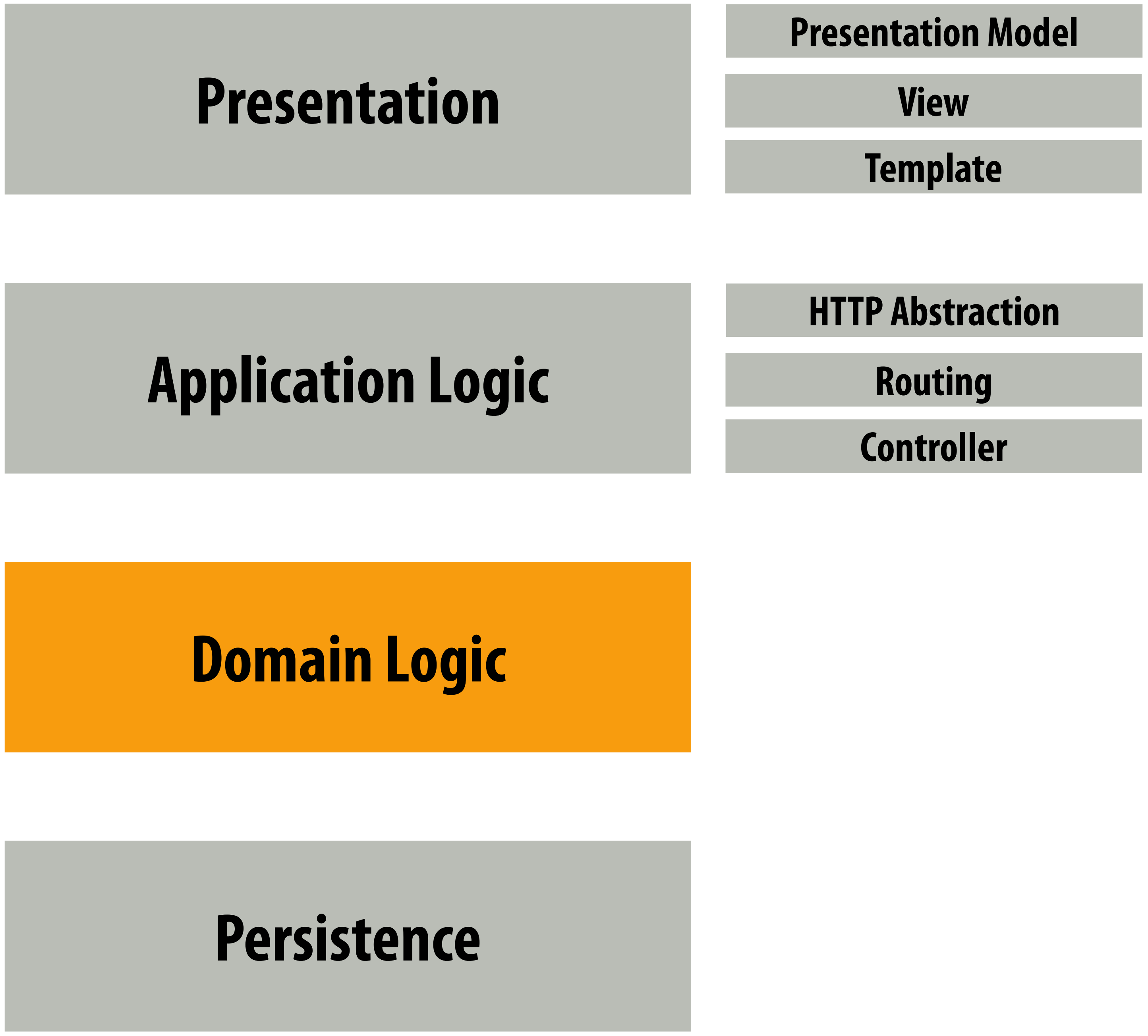
- Presentation Model
- View
- Template

Application Logic

- HTTP Abstraction
- Routing
- Controller

Symfony + Doctrine

Persistence


Presentation
Presentation Model
View
Template
Application Logic
HTTP Abstraction
Routing
Controller
Domain Logic
Persistence

“Responsible for representing concepts of the business, information about the business situation, and business rules.

State that reflects the business situation is controlled and used here, even though the technical details of storing it are delegated to the infrastructure.

This layer is the heart of business software.”

– Eric Evans

Domain Logic

```php
<?php
use SebastianBergmann\Money\Currency;
use SebastianBergmann\Money\Money;

class BankAccount
{
    private $balance;

    public function __construct()
    {
        $this->balance = new Money(0, new Currency('EUR'));
    }

    public function getBalance()
    {
        return $this->balance;
    }
}
```

```php
<?php
use SebastianBergmann\Money\Currency;
use SebastianBergmann\Money\Money;

class BankAccountTest extends PHPUnit_Framework_TestCase
{
    public function testBalanceIsInitiallyZero()
    {
        $ba = new BankAccount;

        $this->assertEquals(
            new Money(0, new Currency('EUR')),
            $ba->getBalance()
        );
    }
}
```

```
➜ wget https://phar.phpunit.de/phpunit.phar

➜ php phpunit.phar --version
PHPUnit 3.7.24 by Sebastian Bergmann.
```

```
➜ wget https://phar.phpunit.de/phpunit.phar

➜ mv phpunit.phar /usr/local/bin/phpunit

➜ chmod +x /usr/local/bin/phpunit

➜ phpunit --version
PHPUnit 3.7.24 by Sebastian Bergmann.
```

```
→ phpunit --bootstrap vendor/autoload.php BankAccountTest
PHPUnit 3.7.24 by Sebastian Bergmann.

.

Time: 72 ms, Memory: 4.00Mb

OK (1 test, 1 assertion)
```

```
➜ tree
.
├── phpunit.xml.dist
├── src
│   └── BankAccount.php
├── tests
│   └── BankAccountTest.php
└── vendor
    └── autoload.php
```

```
<?xml version="1.0" encoding="UTF-8"?>
<phpunit bootstrap="vendor/autoload.php">
 <testsuite name="example">
  <directory>./tests</directory>
 </testsuite>
</phpunit>
```

```
➜ phpunit
PHPUnit 3.7.24 by Sebastian Bergmann.

Configuration read from /home/sb/example/phpunit.xml.dist

.

Time: 80 ms, Memory: 4.00Mb

OK (1 test, 1 assertion)
```

```
→ phpunit --testdox
PHPUnit 3.7.24 by Sebastian Bergmann.

Configuration read from /home/sb/example/phpunit.xml.dist

BankAccount
 [x] Balance is initially zero
```

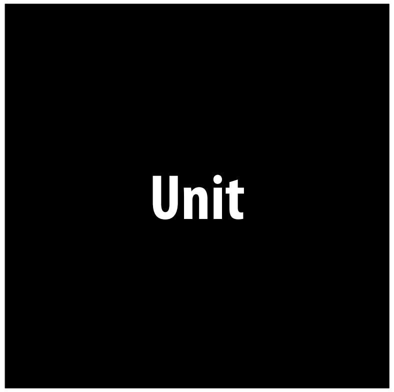
```
→ phpunit --coverage-html /tmp/coverage
PHPUnit 3.7.24 by Sebastian Bergmann.

Configuration read from /home/sb/example/phpunit.xml.dist

.

Time: 91 ms, Memory: 7.25Mb

OK (1 test, 1 assertion)

Generating code coverage report in HTML format ... done
```

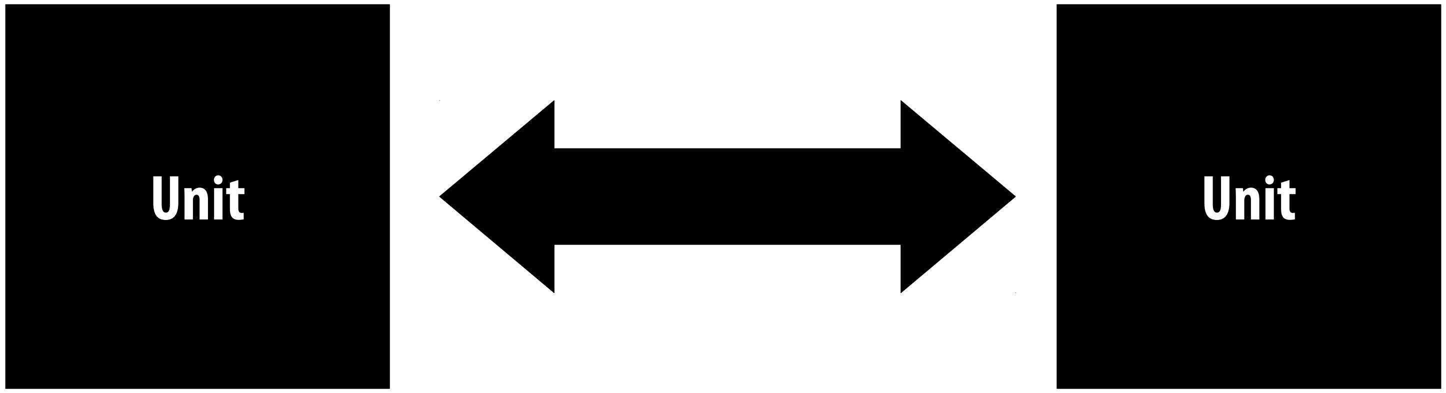
```xml
<?xml version="1.0" encoding="UTF-8"?>
<phpunit bootstrap="vendor/autoload.php"
         backupGlobals="false"
         strict="true"
         verbose="true">

 <testsuite name="example">
  <directory>./tests</directory>
 </testsuite>

 <filter>
  <whitelist addUncoveredFilesFromWhitelist="true">
   <directory>./src</directory>
  </whitelist>
 </filter>
</phpunit>
```

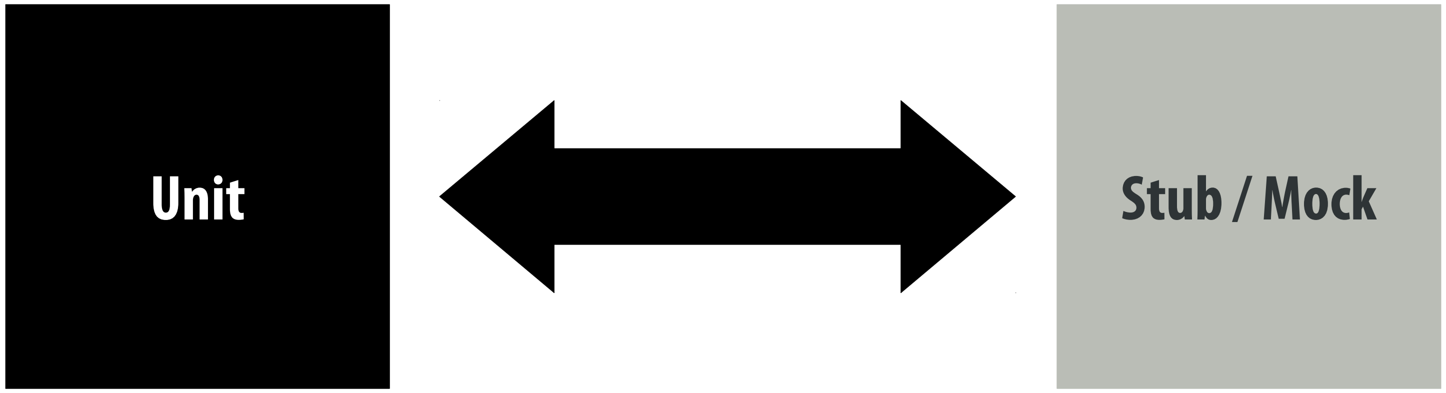
```php
<?php
use SebastianBergmann\Money\Currency;
use SebastianBergmann\Money\Money;

/**
 * @covers BankAccount
 */
class BankAccountTest extends PHPUnit_Framework_TestCase
{
    public function testBalanceIsInitiallyZero()
    {
        $ba = new BankAccount;

        $this->assertEquals(
            new Money(0, new Currency('EUR')),
            $ba->getBalance()
        );
    }
}
```

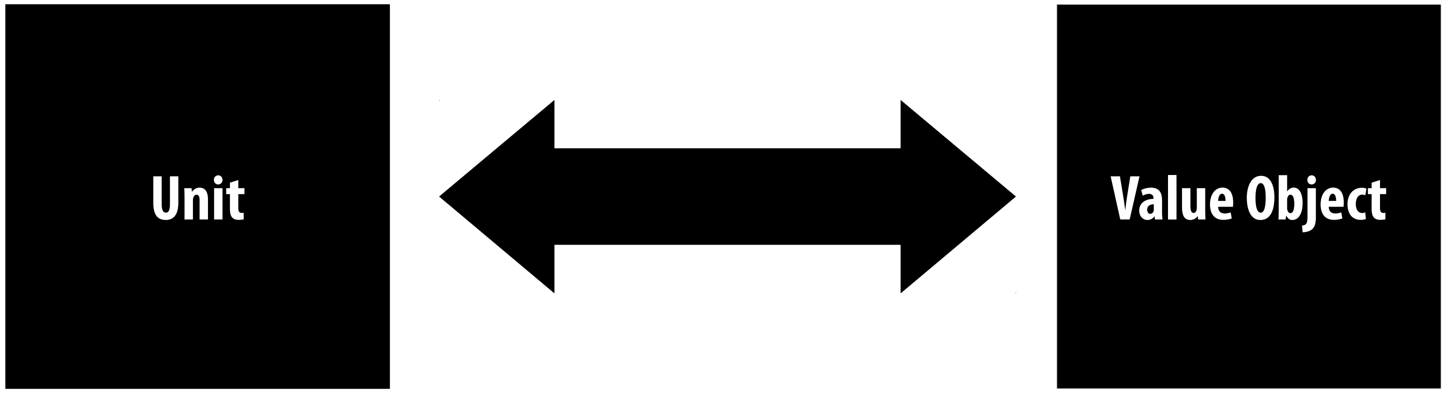
```php
<?php
use SebastianBergmann\Money\Currency;
use SebastianBergmann\Money\Money;

class BankAccountTest extends PHPUnit_Framework_TestCase
{
    /**
     * @covers BankAccount::__construct
     * @covers BankAccount::getBalance
     */
    public function testBalanceIsInitiallyZero()
    {
        $ba = new BankAccount;

        $this->assertEquals(
            new Money(0, new Currency('EUR')),
            $ba->getBalance()
        );
    }
}
```

```
➜ phpunit --coverage-html /tmp/coverage
PHPUnit 3.8.0 by Sebastian Bergmann.

Configuration read from /home/sb/example/phpunit.xml.dist

F

Time: 123 ms, Memory: 7.25Mb

There was 1 failure:

1) BankAccountTest::testBalanceIsInitiallyZero
This test executed code that is not listed as code
to be covered or used


FAILURES!
Tests: 1, Assertions: 1, Failures: 1.

Generating code coverage report in HTML format ... done
```


Unit


Unit
Unit

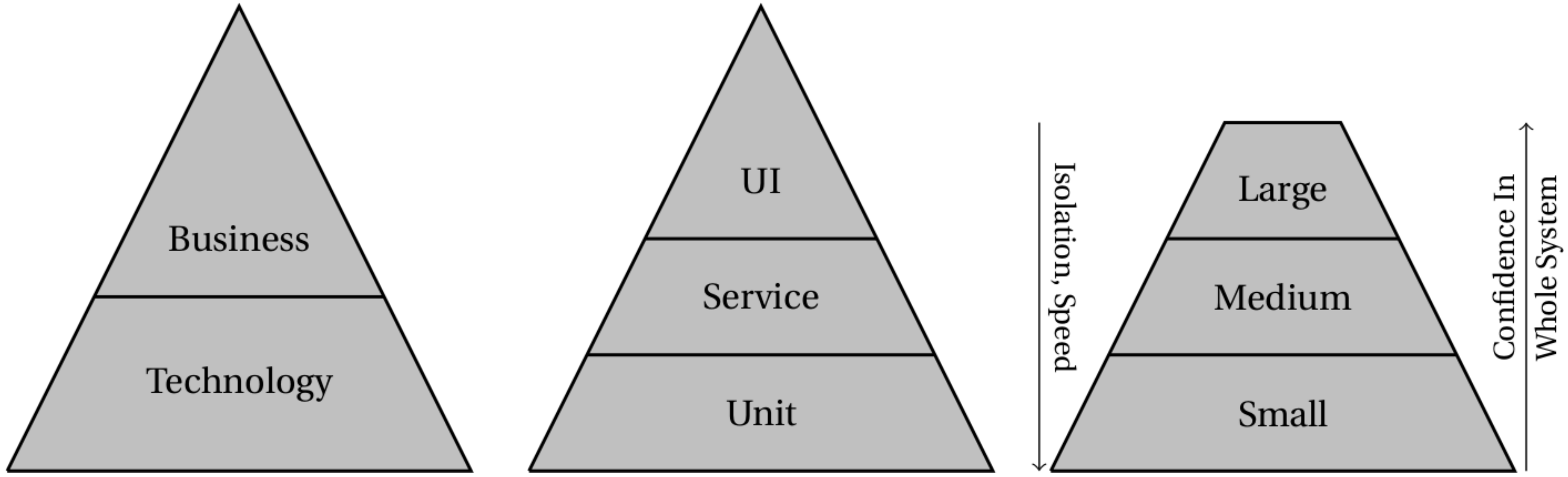

Unit
Stub / Mock

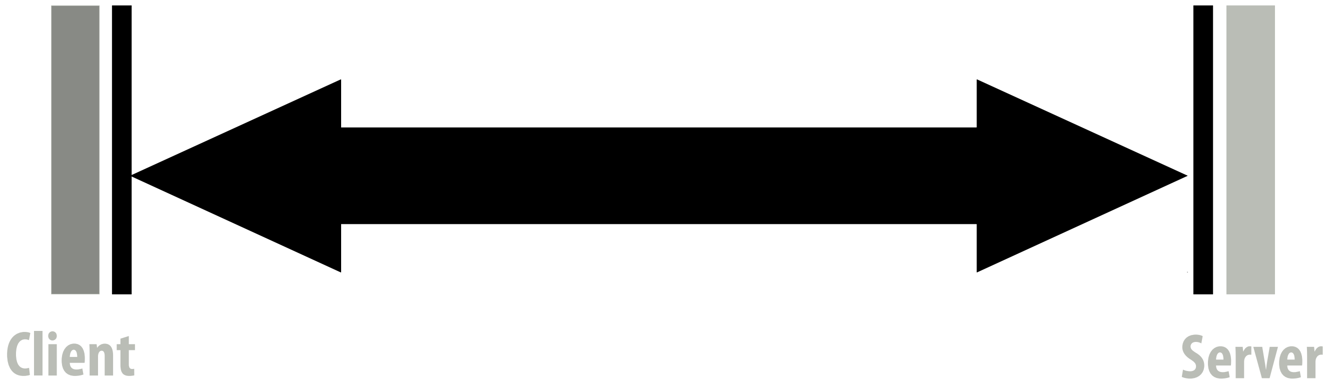

Unit
Value Object

```php
<?php
use SebastianBergmann\Money\Currency;
use SebastianBergmann\Money\Money;

class BankAccountTest extends PHPUnit_Framework_TestCase
{
    /**
     * @covers BankAccount::__construct
     * @covers BankAccount::getBalance
     * @uses   SebastianBergmann\Money\Currency
     * @uses   SebastianBergmann\Money\Money
     */
    public function testBalanceIsInitiallyZero()
    {
        $ba = new BankAccount;

        $this->assertEquals(
            new Money(0, new Currency('EUR')),
            $ba->getBalance()
        );
    }
}
```

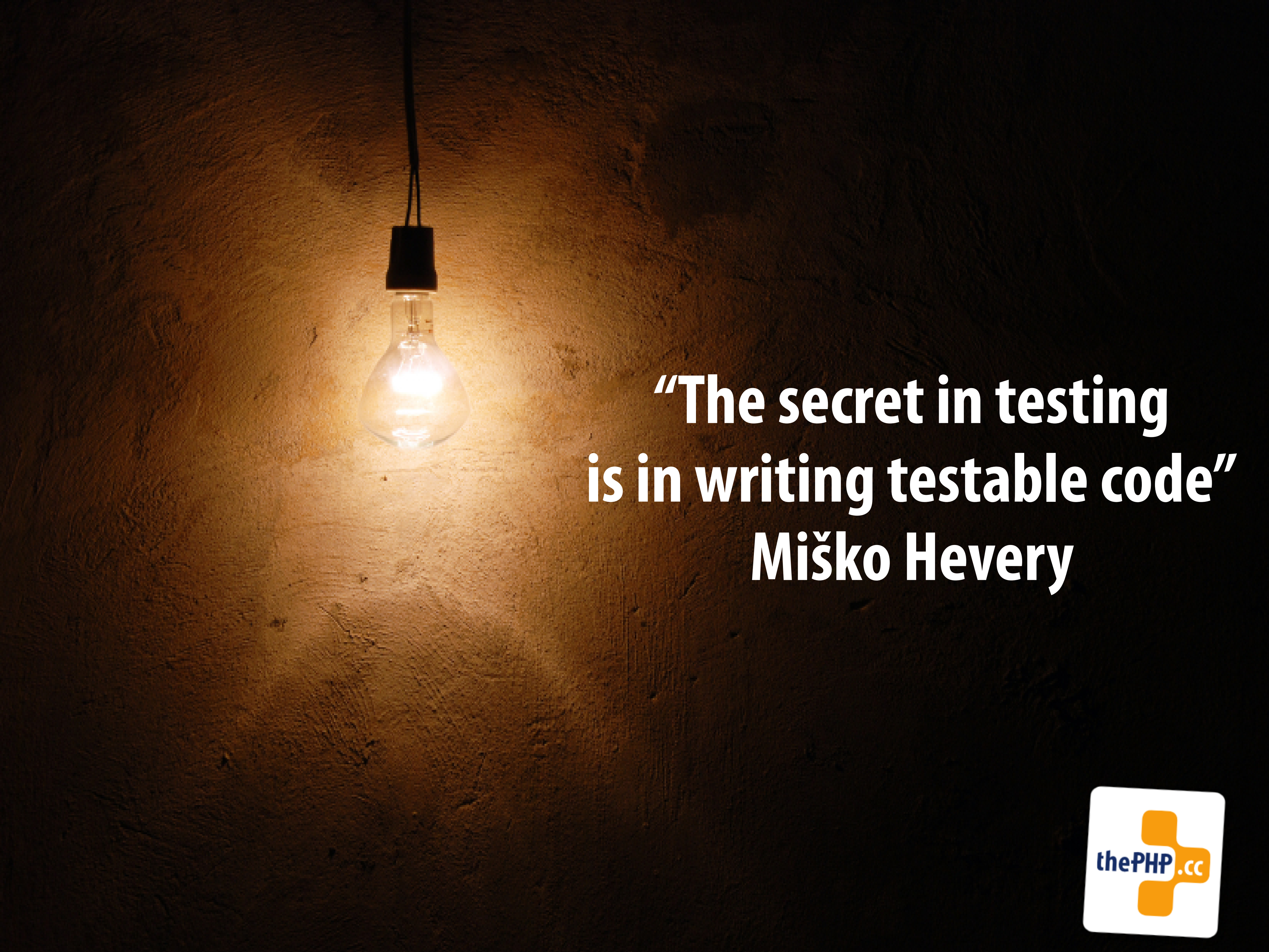
Client
Server

Client
Server


Business
Technology
UI
Service
Unit
Isolation, Speed
Large
Medium
Small
Confidence In Whole System


Client
Server

```php
<?php
use Symfony\Bundle\FrameworkBundle\Test\WebTestCase;

class DefaultControllerTest extends WebTestCase
{
    public function testIndex()
    {
        $client = $this->createClient();
        $client->request('GET', '/bankaccount/1');

        $this->assertContains(
            'The balance of bank account #1 is 1,00 €.',
            $client->getResponse()->getContent()
        );
    }
}
```



“The secret in testing
is in writing testable code”
Miško Hevery

“When applying a framework, the team needs to focus on its goal: building an implementation that expresses a domain model and uses it to solve important problems.

The team must seek ways of employing the framework to those ends, even if it means not using all of the framework's features.

Judiciously applying only the most valuable of framework features reduces the coupling of the implementation and the framework, allowing more flexibility in later design decisions.”

– Eric Evans

Recommended Reading

- http://whitewashing.de/2013/09/04/decoupling_from_symfony_security_and_fosuserbundle.html
- http://williamdurand.fr/2013/08/07/ddd-with-symfony2-folder-structure-and-code-first/
- http://whitewashing.de/2013/06/27/extending_symfony2__controller_utilities.html
- http://verraes.net/2013/04/decoupling-symfony2-forms-from-entities/
- http://whitewashing.de/2013/02/19/extending_symfony2__paramconverter.html

talks.thePHP.cc
sebastian@thePHP.cc
@s_bergmann

thePHP.cc

sharing experience

thePHP.cc