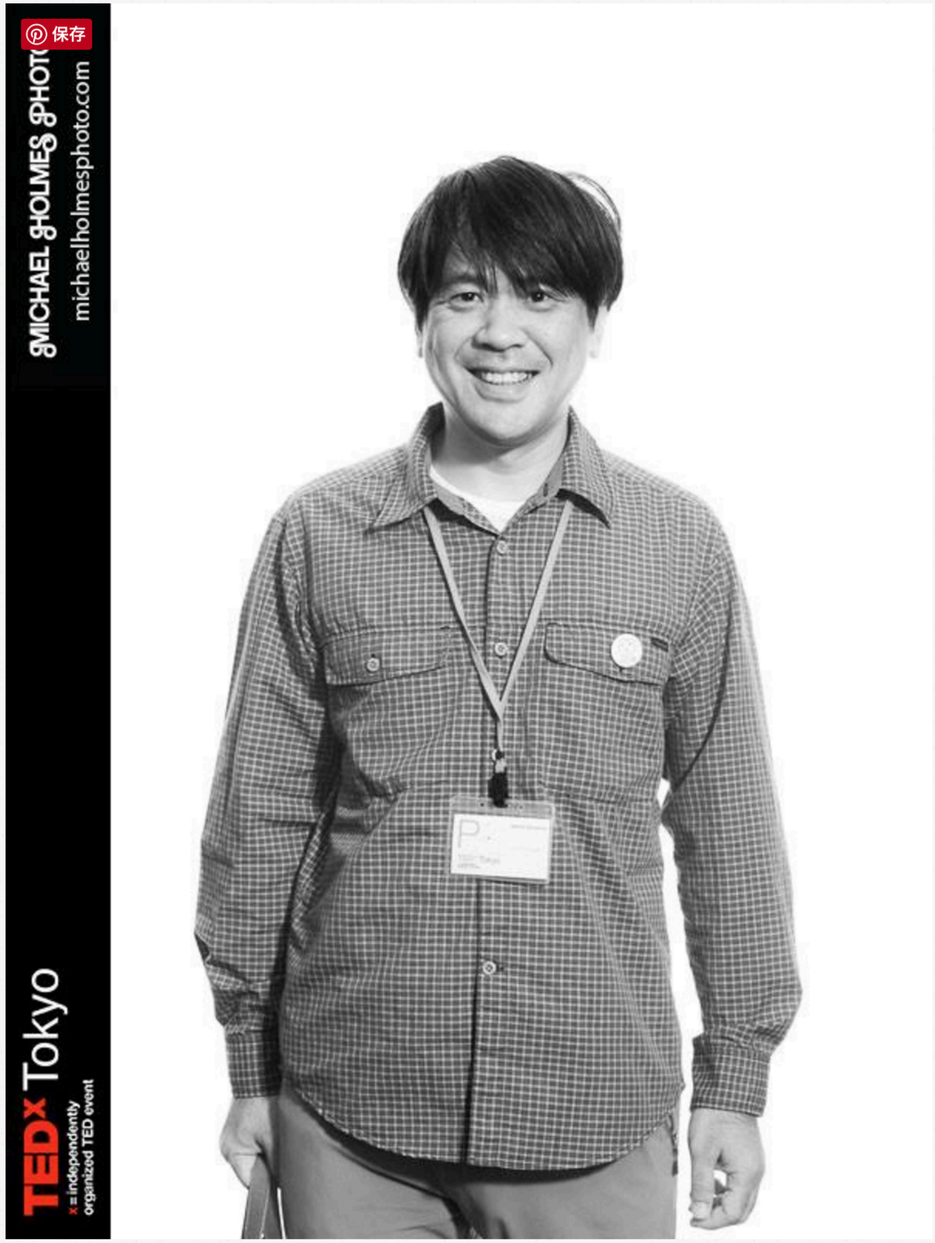
A resource orientated framework using the DI /AOP/REST Triangle



Akihito Koriyama

PHP Architect. Speaker. Traveller.

Speaking

- 2012 phpmatsuri - PHP: DIS IS IT slide
- 2013 A resource orientated framework using the DI/AOP/REST Triangle
  - PHPNW program / photo
  - BBC Sports photo
  - leedsphp photo
- 2014 PHP Conference Osaka Keynote - slide / talk
- 2015 PHP Conference Fukuoka Keynote -slide
- 2016 PHP Conference Osaka

Interview

- Code camp (as an interviewee)
- The father of PHP, Rasmus Lerdorf (as an interviewer)

TEDxTokyo Conference

- 2013 Participant
- 2014 Participant

Game Programming

- 1989 NES Quinty / Mendel Palace
- 1991 NES,GameBoy Yoshi no tamago (JP) / Yoshi (US) / Yoshi and Mario (EU)
- 1994 3DO Starblade

BEAR.Sunday is an application framework.
But it offers no libraries.

(We have good stuff)

その代わりに３つのオブジェクトフレームワーク

Instead, it offers three object frameworks.

What is a framework ?

エンドユーザコードに対する一連の設計制約

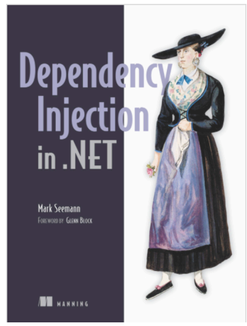
”imposes a set of design constraints on end-user code.”

Dependency Injection

定義

DIは疎結合のコーディングを可能にするソフトウエアデザイン原則とパターンのセット

“Dependency Injection is a set of software design principles and patterns that enable you to develop loosely coupled code.”



原則

実装ではなく、
インターフェイスに対してプログラムする

“Program to an interface, not an implementation.”

Gang of Four 1994

Kent Beckand, Ward Cunningham 1987

GET SHIT DONE !

.. OK ?

maintenance ?

upgrade ?

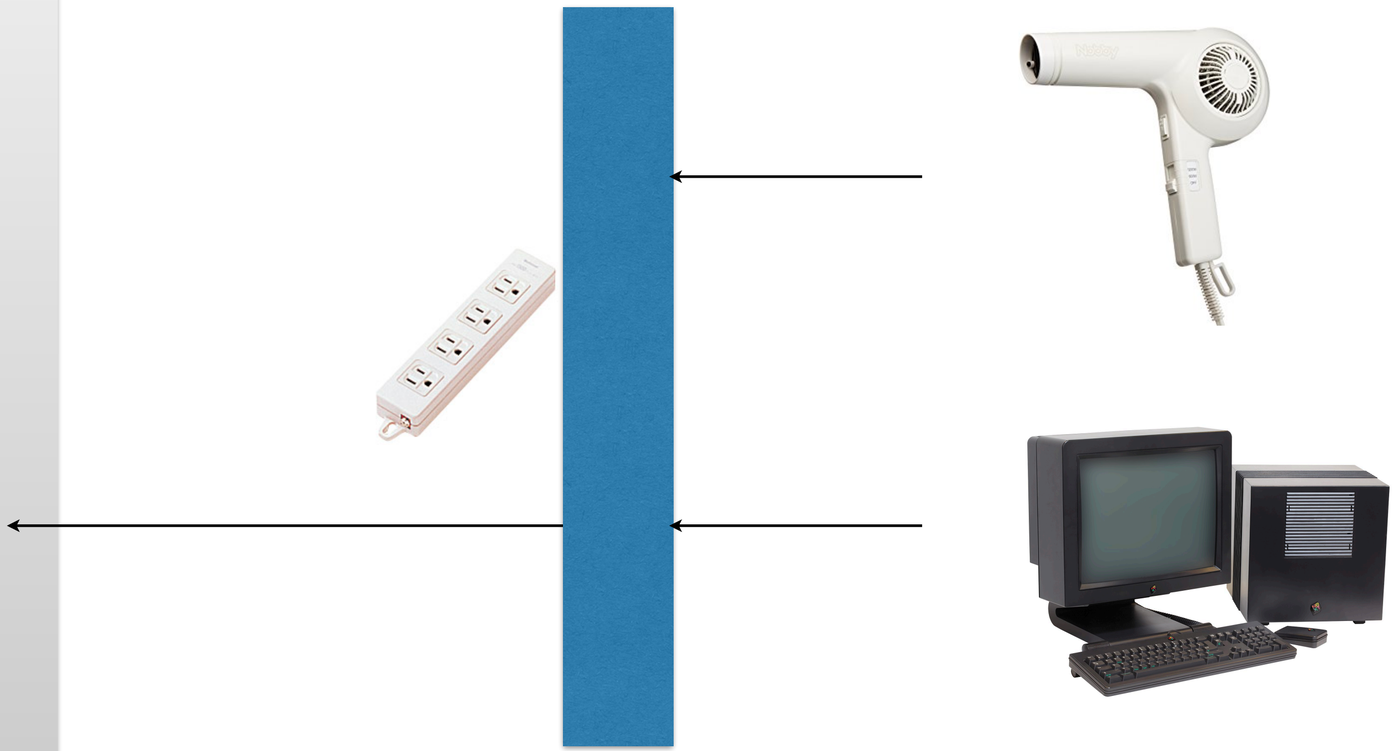
リスコフの置換原則

Liskov substitution principle

Decorator Pattern

UPS

Composite Pattern



Null Object Pattern

Adapter Pattern

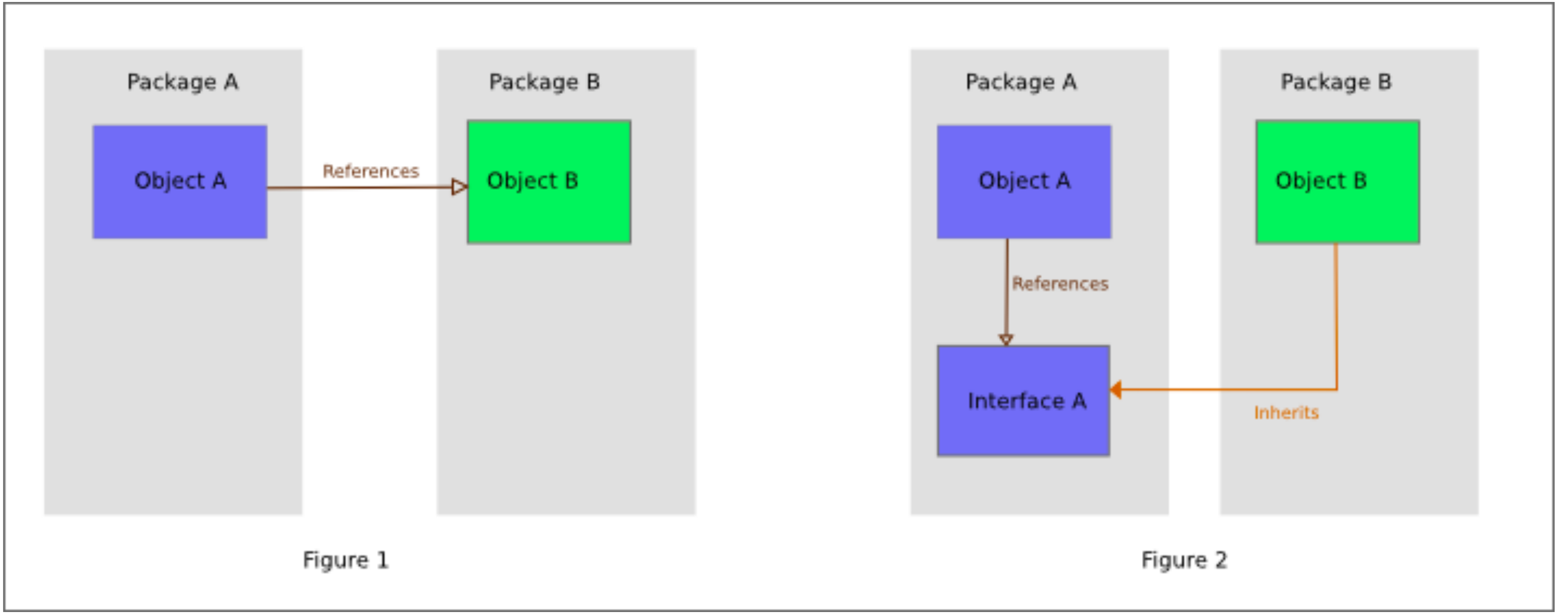


Figure 1

Figure 2

DIP: 依存関係逆転の原則

Dependency inversion principle

- コードは同等以上のレベルの抽象に依存する
- 下位レベルの詳細に依存しない

- Code should depend on things that are at the same or higher level of abstraction
- High level policy should not depend on low level details

“The principle of dependency inversion is at the root of many of the benefits claimed for object-oriented technology.

Its proper application is necessary for the creation of reusable frameworks”

“依存性逆転原則はオブジェクト指向技術の利益の根幹をなすもの。
再利用を可能にするフレームワークの開発にはその原則適用が必須”

Robert Martin (a.k.a. Uncle Bob)

Google Guice

開発元	Google
初版	2007年（11年前）
最新版	4.1 / 2016年6月17日（20か月前）
リポジトリ	github.com/google/guice
プログラミング言語	Java
対応OS	クロスプラットフォーム
種別	DIフレームワーク
ライセンス	Apache License 2.0
公式サイト	github.com/google/guice

テンプレートを表示

Ray.Di

Dependency Injection framework

Ray.DiはGoogleのGuiceの主要な機能を持つPHPのDIフレームワークです。

概要

Ray.Diには以下の機能があります。

- コンストラクタインジェクション、セッターインジェクション、アシスティドインジェクション
- 自動ワイアリング
- DSLでの束縛（リンク束縛、プロバイダー束縛、インスタンス束縛、コンストラクタ束縛）
- シングルトンスコープ

抽象に依存した上位コード

Higher-level code depends on abstraction

```
class BillingService
{
    private $processor;
    private $logger

    public function __construct(ProcessorInterface $processor, LoggerInterface $logger)
    {
        $this->processor = $processor;
        $this->logger = $logger;
    }
}
```

③読み取り専用 read-only

①インターフェイスで受け取り

passed through interface

②プロパティに代入

set to the property

抽象に具象を束縛

bind abstraction to concretion

```
class BillingModule extends AbstractModule
{
    protected function configure()
    {
        $this->bind(ProcessorInterface::class)->to(PaypalProcessor::class);
        $this->bind(LoggerInterface::class)->to(DatabaseLogger::class);
    }
}
```

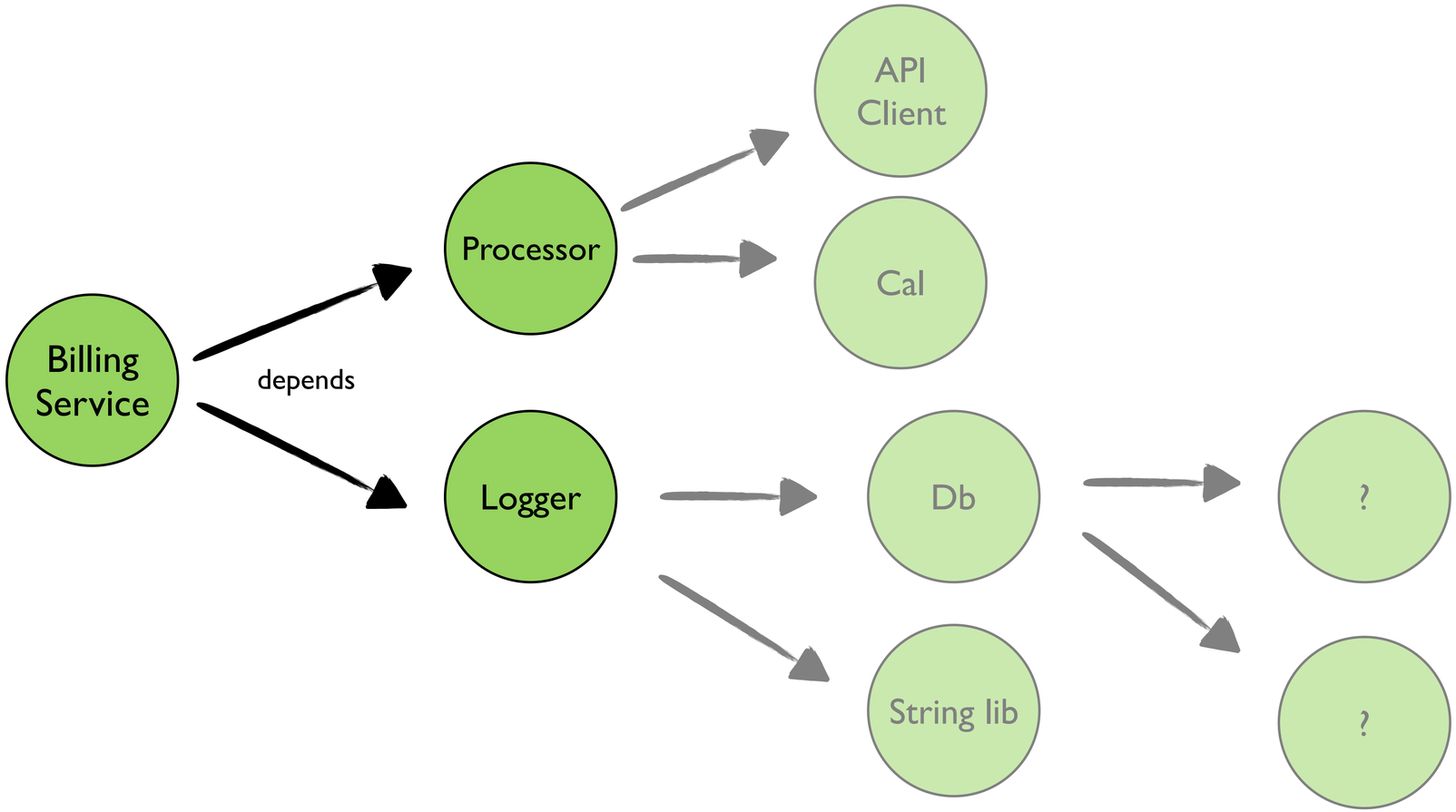
オブジェクトグラフ生成

building object graph

```
$injector = new Injector(new BillingModule);
$billingService = $injector->getInstance(BillingService::class);
```

オブジェクトグラフ

object graph

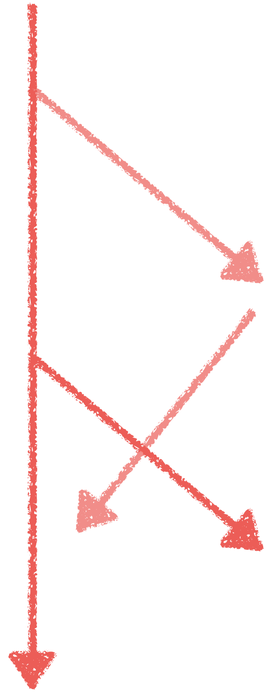


Anti Pattern
Dependency Lookup

```php
// service locator
$container->get('logger')

// static proxy (global service locator)
Cache::get($id);
```

Anti Pattern
“MODE”

```php
if ($mode === 'json') {
    (new JsonRenderer)->render($data);
} elseif ($mode === 'xml') {
    (new XmlRenderer)->render($data);
}

if (APP_DEBUG) {
    $this->logger->log($bill);
}
```



手続きをコード

code “procedures”

いくつものパス

multiple execution path

生成と利用が分離したDIパターン

DI pattern

Compile

```
$this->bind(RendererInterface::class)->to(JsonRenderer::class);
$this->bind(LoggerInterface::class)->to(NullLogger::class);
```

Runtime

```
public function __constructor(RendererInterface, $renderer, LogerInterface $logger)
{
    $this->renderer = $renderer;
    $this->logger = $logger;
}
```

```
public function render()
{
    $this->renderer->render($data);
    $this->logger->log($data);
}
```

構造をコード/ シングルパス

code “structure” / single execution path

🤔

どこでオブジェクトグラフを生成するか？

Where should we compose object graphs?

DI good practice

コードは可能な限りインジェクターを直接扱わない。
その代わり、アプリケーションのエントリーポイントで 1 つの
ルートオブジェクトを生成してアプリケーションを起動する

“Your code should deal directly with the Injector as little as possible.
Instead, you want to bootstrap your application by injecting one root object.”

Application = One Root Object

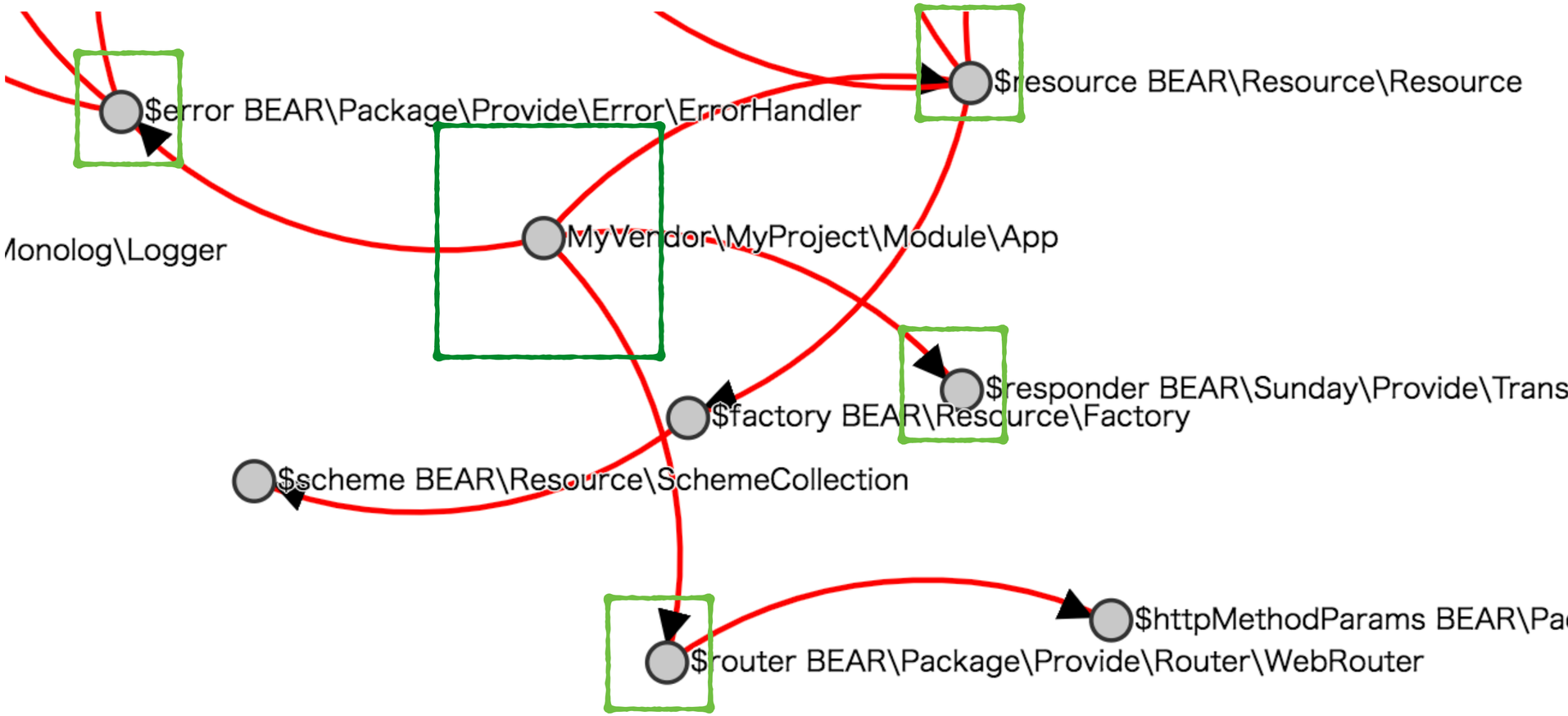
Application class

```php
/**
 * @param RouterInterface   $router    Resource router
 * @param TransferInterface $responder Resource responder
 * @param ResourceInterface $resource  BEAR.Resource client
 * @param ErrorInterface    $error     Error handler
 */
public function __construct(
    RouterInterface $router,
    TransferInterface $responder,
    ResourceInterface $resource,
    ErrorInterface $error
) {
    $this->router = $router;
    $this->responder = $responder;
    $this->resource = $resource;
    $this->error = $error;
}
```

Application Script

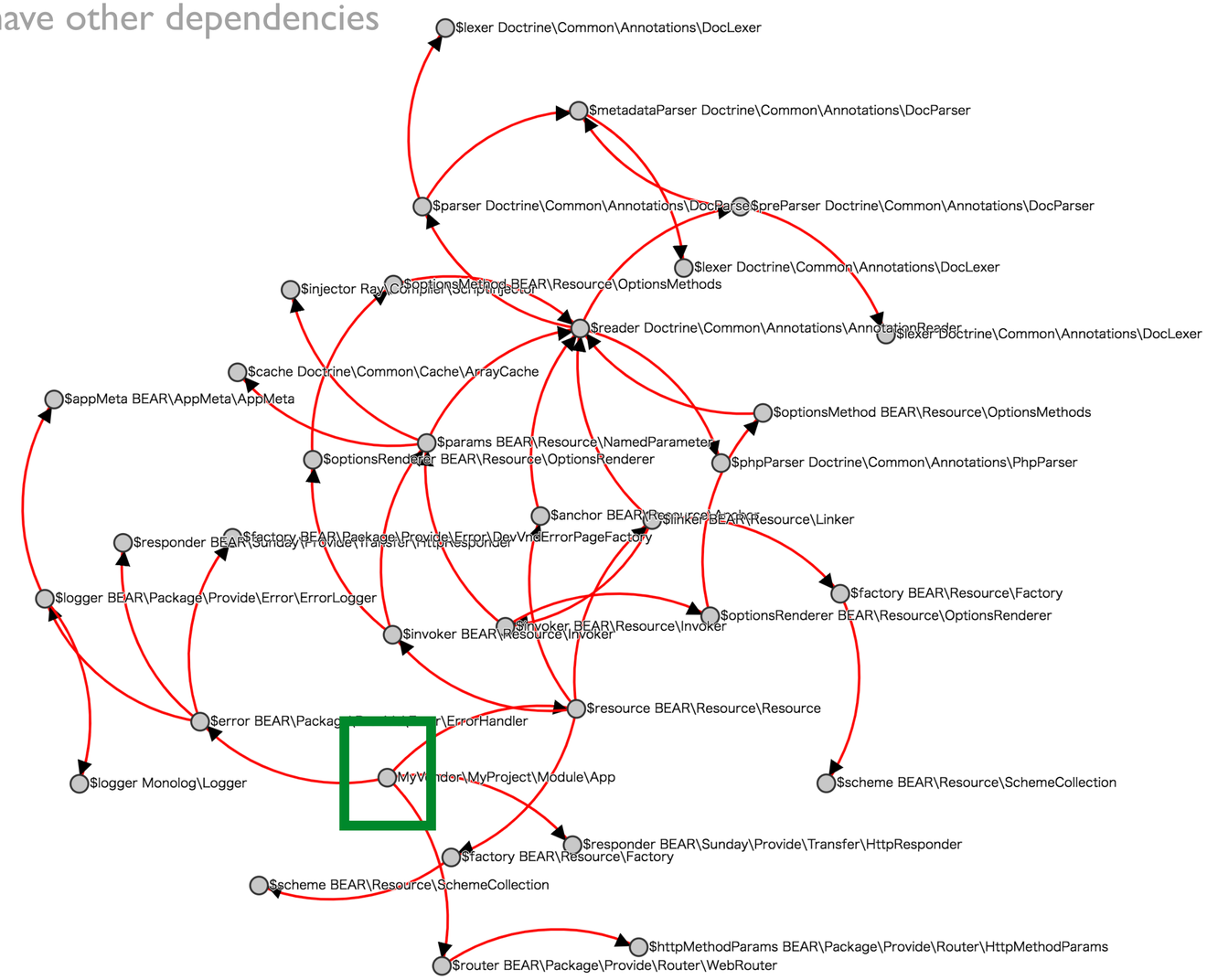
```php
// compile
$app = (new Bootstrap)->getApp('MyVendor\MyProject', $context);
// runtime
$request = $app->router->match($GLOBALS, $_SERVER);
try {
    $page = $app->resource->{$request->method}->uri($request->path)($request->query);
    $page->transfer($app->responder, $_SERVER);
    exit(0);
} catch (\Exception $e) {
    $app->error->handle($e, $request)->transfer();
    exit(1);
}
```

Root Object

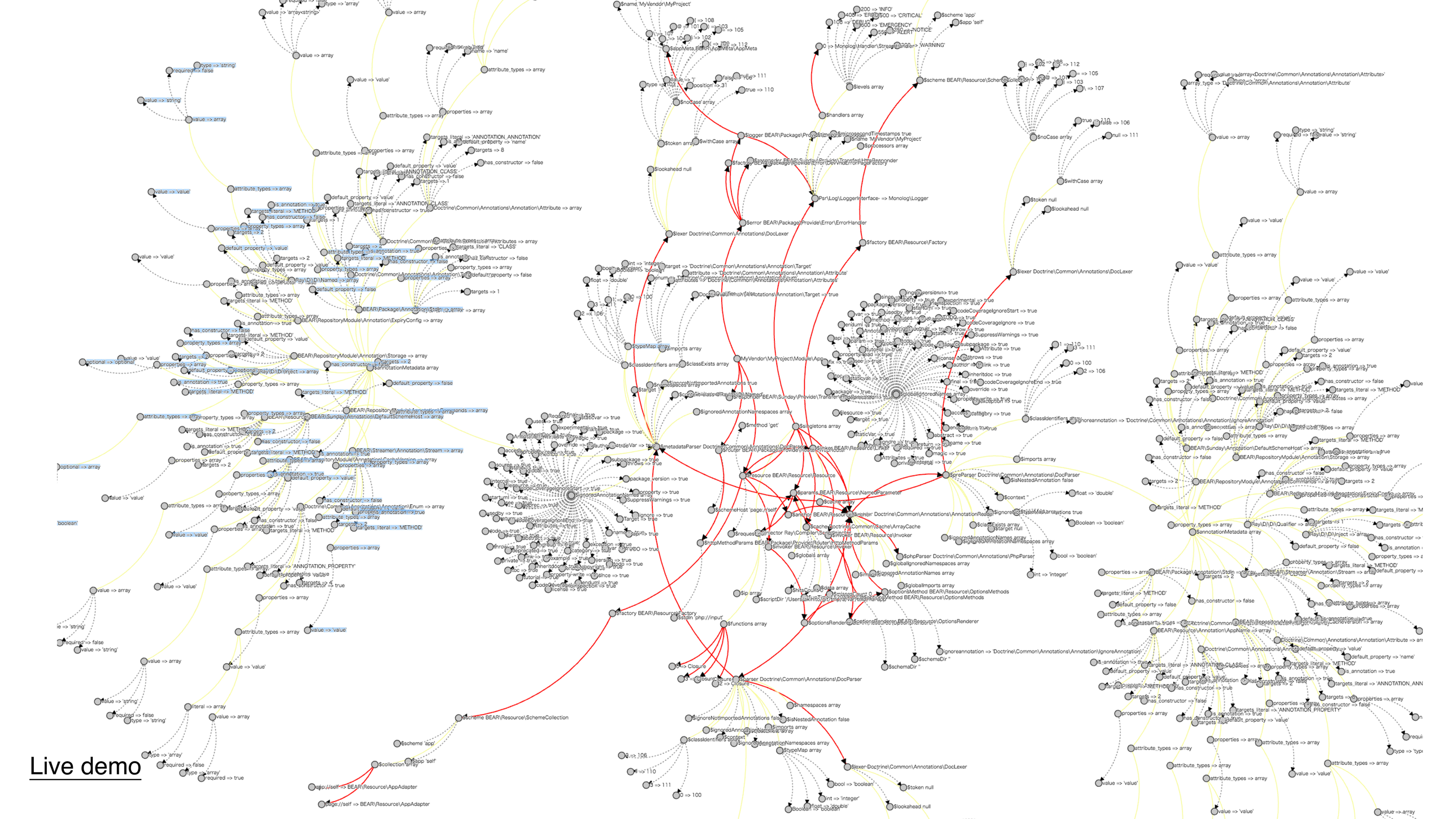

$error BEAR\Package\Provide\Error\ErrorHandler
$resource BEAR\Resource\Resource
Monolog\Logger
MyVendor\MyProject\Module\App
$responder BEAR\Sunday\Provide\Trans
$factory BEAR\Resource\Factory
$scheme BEAR\Resource\SchemeCollection
$httpMethodParams BEAR\Pa
$router BEAR\Package\Provide\Router\WebRouter

それぞれの依存は保持してるか保持されているか

Dependencies have other dependencies



Each object either *contains* or *belongs to*.


Live demo

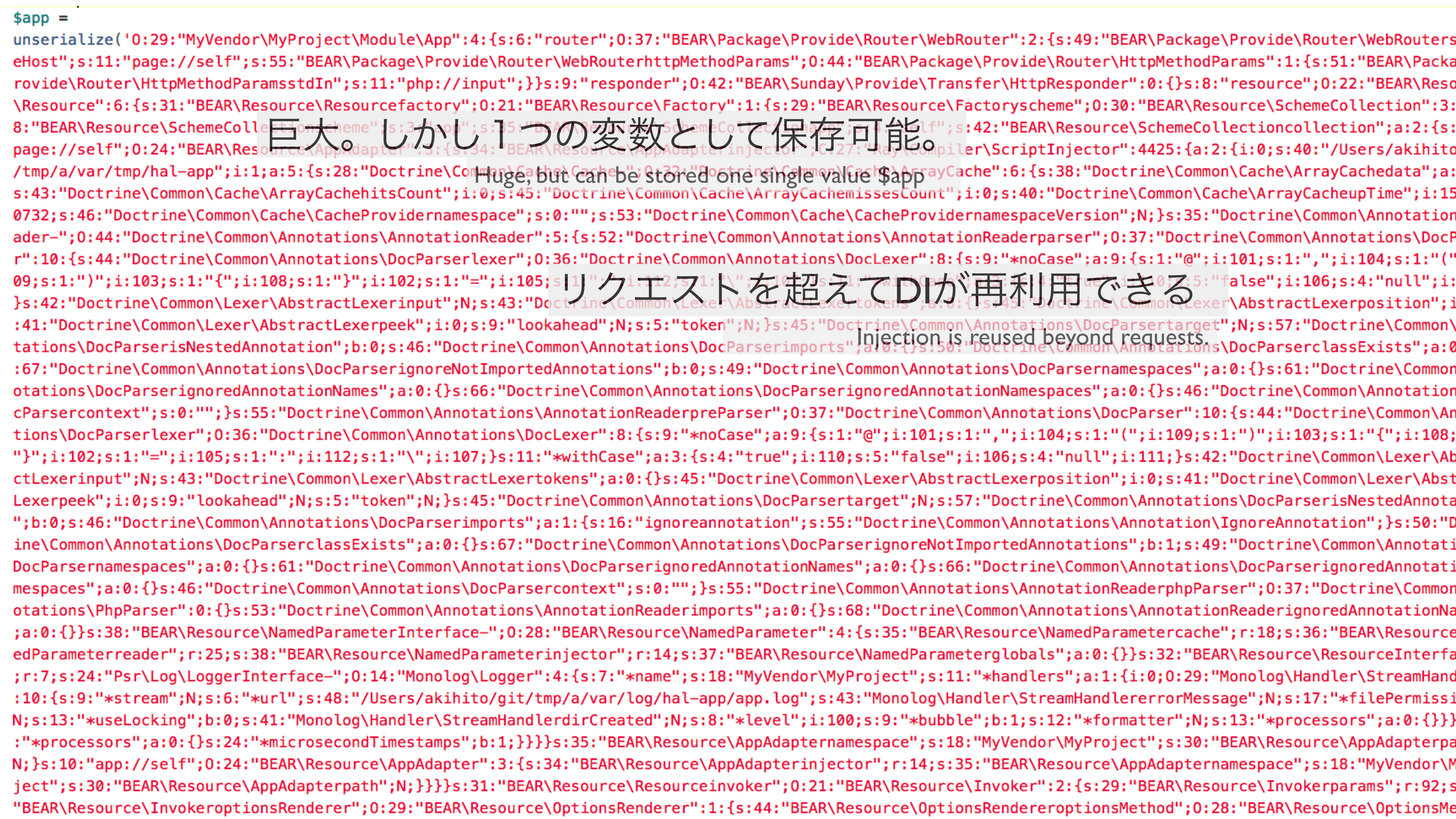
setLogDir => array
$hash array
setWriter => array
0 => null
$logs array
$writers array
$serializer Any\Serializer\Serializer
setNamedArgs => array
$setters array
0 => '/Users/kooriyama/git/BEAR.Package/vendor/bear/demo-apps/Demo.Sandbox/var/tmp'
$globalImports array
$phpParser Doctrine\Common\Annotations\PhpParser
$imports array
$ignoredAnnotationNames array
ignoreannotation => 'Doctrine\Common\Annotations\Annotation\IgnoreAnnotation'
$globalIgnoredNames array
$reader Doctrine\Common\Annotations\AnnotationReader
$preParser Doctrine\Common\Annotations\DocParser
$instance Doctrine\Common\Annotations\CachedReader
$class 'Doctrine\Common\Annotations\AnnotationReader'
$interceptors null
$postConstruct null
setReader => array
setSqlLogger => array
$args array
$consoleOutput BEAR\Package\Provide\ConsoleOutput\ConsoleOutput
$response BEAR\Package\Provide\WebResponse\HttpFoundation
$router BEAR\Package\Provide\Router\WebRouter
$globals null
$logger Ray\Di\CompilationLogger
$config null
$factory BEAR\Resource\Factory
Ray\Di\DependencyFactory
Ray\Di\DependencyReference
$resource BEAR\Resource\Resource
$newRequest BEAR\Resource\Request
$class 'BEAR\Resource\Resource'
$uri null
$links array
$options array
$query array
$result null
$method ''
0 => 'Demo\Sandbox'
setAppName => array
2 => 106
1 => 110
0 => 100
3 => 111
setSchemeCollection => array
$parser Doctrine\Common\Annotations\DocParser
$namespaces array
$target null
$typeMap array
Boolean => 'boolean'
float => 'double'
int => 'integer'
bool => 'boolean'
setResourceLogger => array
3 => null
$scheme null
$http null
$page null
$app null
$host null

koriym/printo

```
composer require koriym/printo
```

An object graph visualizer.

```
print_o($obj);
```


$app =
巨大。しかし1つの変数として保存可能。
Huge, but can be stored one single value $app
リクエストを超えてDIが再利用できる
Injection is reused beyond requests.

ランタイムコンパイルも必要

Still, We need runtime compile

- どのページかまだ分からない No sure which page will be load
- PDOオブジェクトやクロージャはシリアライズできない Not everything can be serialized

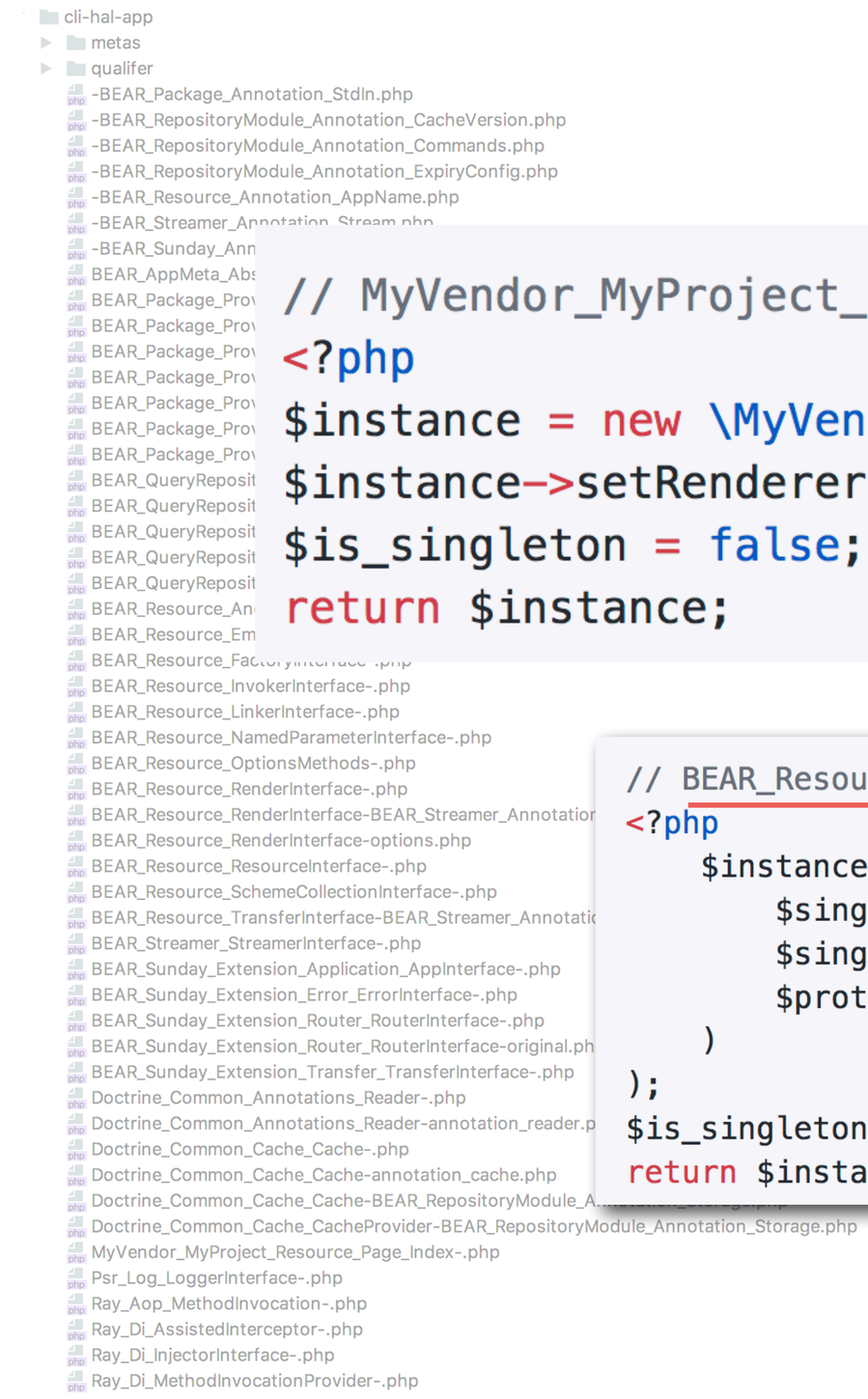
Dependency Injection Compiler

```php
// DI DSL
$this->bind(LoggerInterface::class)->to(DatabaseLogger::class);
```

Compile

```php
// LoggerInterface.php
<?php
return DatabaseLogger();
```

```php
($instance = require 'LoggerInterface.php';)
```



Instance script

```php
// MyVendor_MyProject_Resource_Page_Index-.php
<?php
$instance = new \MyVendor\MyProject\Resource\Page\Index();
$instance->setRenderer($singleton('BEAR\\Resource\\RenderInterface-'));
$is_singleton = false;
return $instance;
```

```php
// BEAR_Resource_RenderInterface-.php
<?php
    $instance = new \BEAR\Package\Provide\Representation\HalRenderer(
        $singleton('Doctrine\\Common\\Annotations\\Reader-'),
        $singleton('BEAR\\Resource\\ResourceInterface-'),
        $prototype('BEAR\\Package\\Provide\\Representation\\HalLink-'
    )
);
$is_singleton = true;
return $instance;
```

```php
// Doctrine_Common_Annotations_Reader-.php
$instance = new \Doctrine\Common\Annotations\AnnotationReader(null);
$is_singleton = true;
return $instance;
```

DIの利点

DI benefit

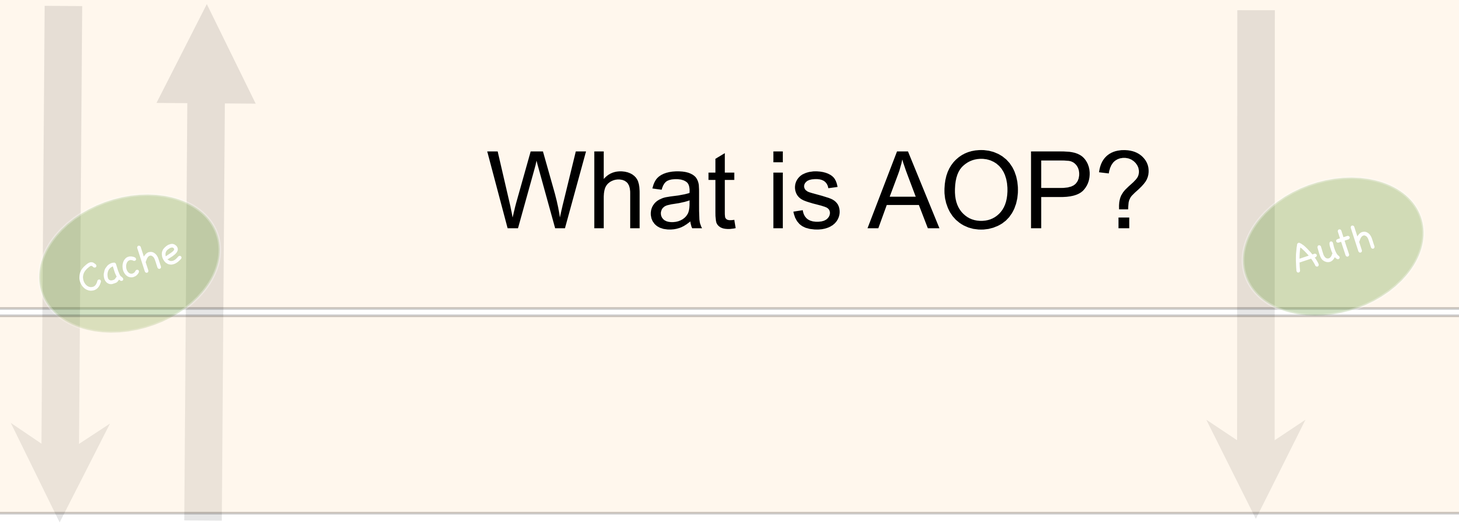
- 拡張性 Extensibility
- 並行開発 Parallel development
- メンテナンス Maintainability
- テスタビリティ Testability
- オブジェクトと利用の生成の分離 Clear separation of object instantiation and object usage
- オブジェクトデリバリー Object delivery

1st framework: DI Framework

- DSLとアノテーションを使ったDIフレームワーク
  DI framework with DSL w/ annotation
- 疎結合 loosely coupled
- コンパイルとランタイムの区別 compile time and runtime
- アプリケーションは単一のオブジェクト application root object
- Codegenで高速 fast

“Ray”

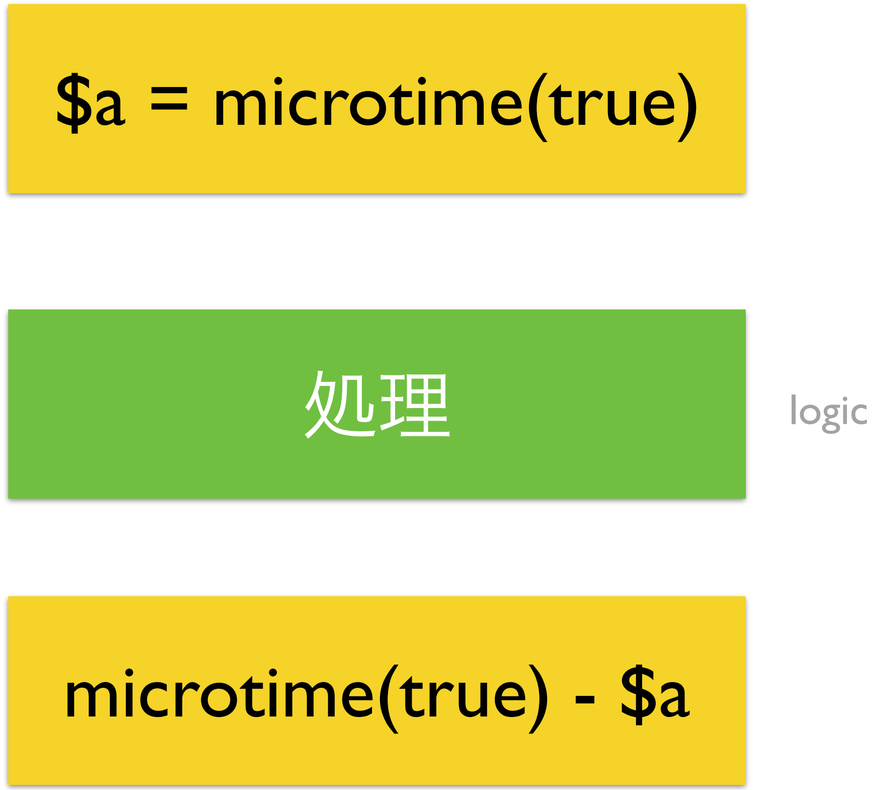
Aspect Oriented Programming



What is AOP?

横断的関心事を分離可能にすることにより
モジュラリティーを増加させることを目的とした
プログラミングパラダイム

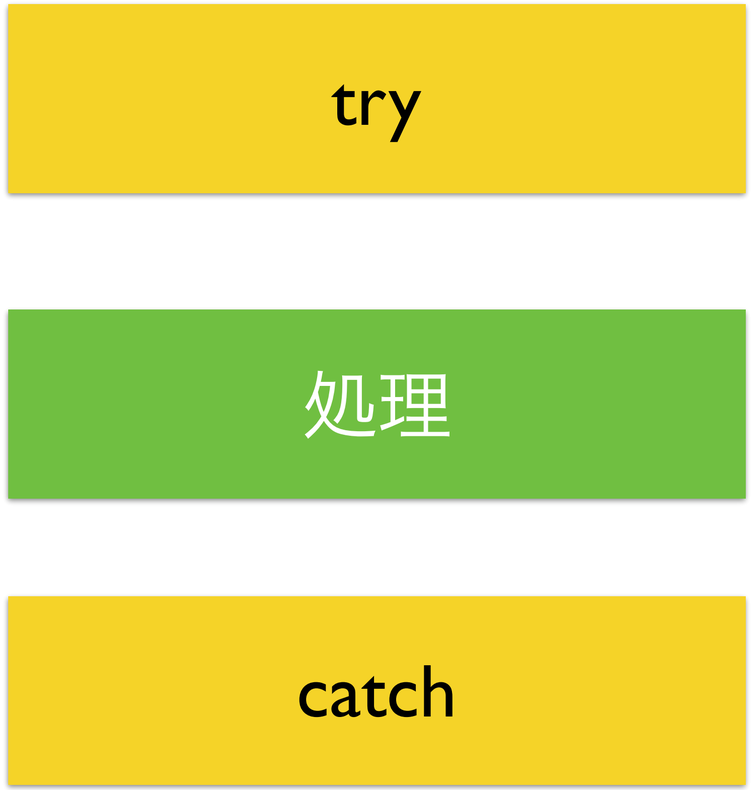
A programming paradigm that aims to increase modularity
by allowing the separation of cross-cutting concerns


$a = microtime(true)
処理
logic
microtime(true) - $a

BEGIN

処理

COMMIT

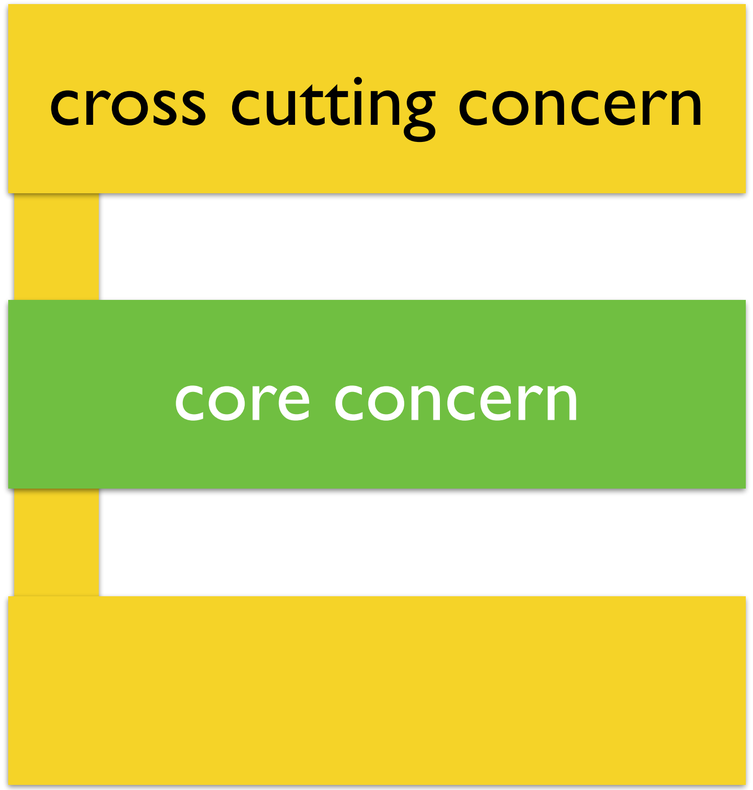

try
処理
catch

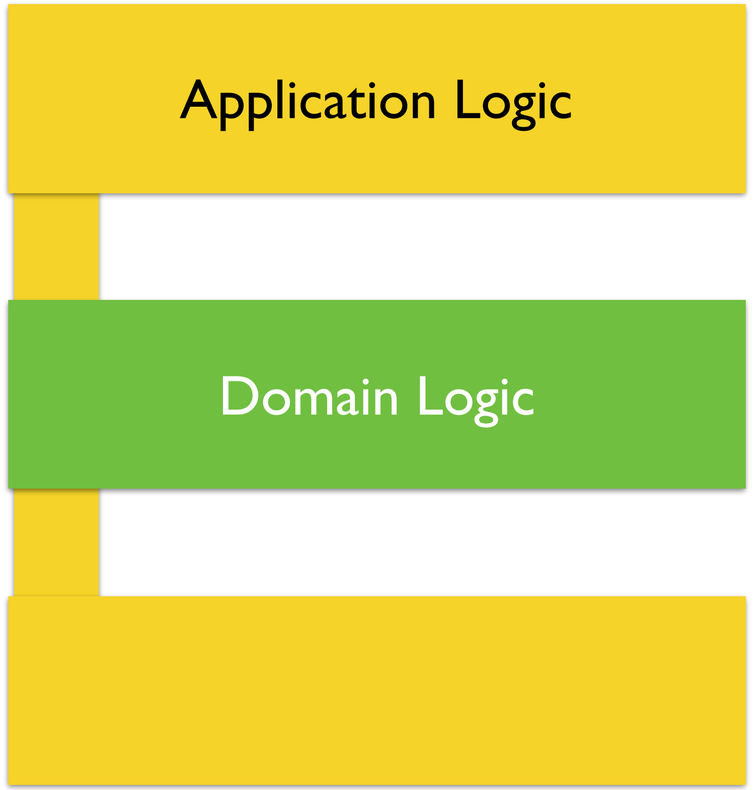

cache check
処理
save cache


cache check
処理
処理
処理
save cache

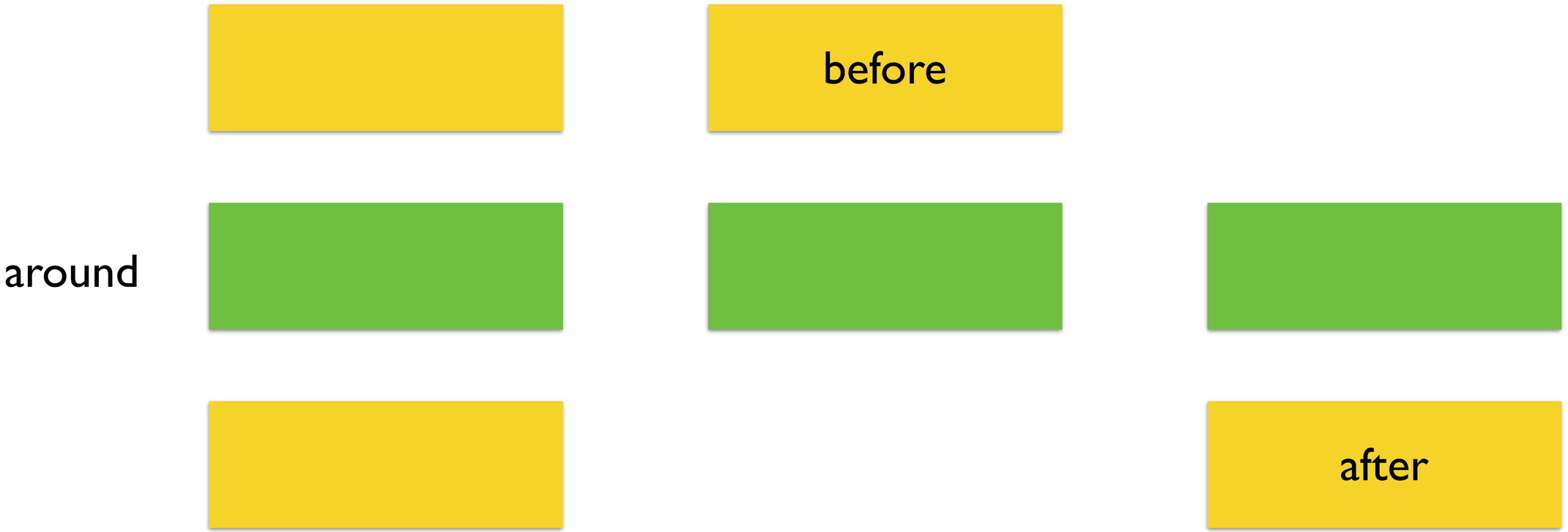
横断的関心

本質的関心


cross cutting concern
core concern


Application Logic
Domain Logic

Aspect Pattern



Rock Concert Example

Gregor Kiczales
Professor
Software Practices Lab
Department of Computer Science
University of British Columbia
201-2366 Main Mall
Vancouver, B.C., Canada V6T 1Z4

+1.604.822.4806 [voice]
+1.604.822.5485 [FAX]
gregor@cs.ubc.ca

Office: ICICS/CS 211

My research is directed at enabling programmers to write programs that, as much as possible, look like their design. I believe that programs that clearly express the design structure they implement are easier to maintain, because questions about what a part of the program does, why it does that, and what other parts of the program depend on that behavior become easier to answer.

In pursuit of this goal, most of my research has been in programming language design and implementation, but I am also interested in programming environments, coding styles, and other kinds of tools.

I have worked extensively in the area of aspect-oriented programming (AOP). I had the good fortune to lead the fantastic Xerox PARC team that developed aspect-oriented programming and the AspectJ programming language. Some of my current research in the Software Practices Lab is AOP related. I am also beginning to work on a new project, as outlined in my 2007 OOPSLA Keynote talk, which is available from my talks page. Another recent focus is the development of CPSC 110, the department's new introductory course for first year students. The course is based

Gregor Kiczales

AOP Alliance (Java/J2EE AOP standards)

AOP Alliance, presentation

The AOP Alliance project is a joint open-source project between several software engineering people who are interested in AOP and Java.

LICENCE: all the source code provided by AOP Alliance is **Public Domain**.

view the members' list....

We believe that Aspect-Oriented Programming (AOP) offers a better solution to many problems than do existing technologies such as EJB. AOP Alliance intends to facilitate and standardize the use of AOP to enhance existing middleware environments (such as J2EE), or development environements (e.g. JBuilder, Eclipse). The AOP Alliance also aims to ensure interoperability between Java/J2EE AOP implementations to build a larger AOP community.

read more details...

Materials

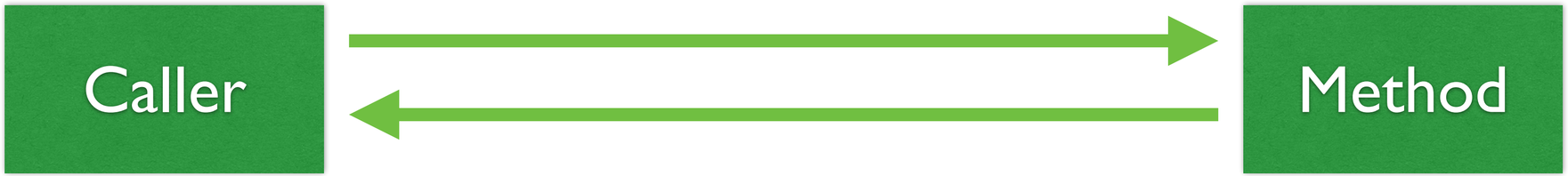
- Sourceforge site
- White paper draft (The AOP Alliance: Why Did We Get In?): (pdf) (ps) (html online version)
- Online Sourceforge CVS Repository
- Online Javadoc-generated AOP Alliance specifications preview (this is unstable work in progress).
- Mailing list:
    - mailto:aopalliance-discuss@lists.sourceforge.net
    - archive
    - (un)subscribe

```php
class Post extends AppModel {

    public function newest() {
        $result = Cache::read('newest_posts', 'longterm');
        if (!$result) {
            $result = $this->find('all');
            Cache::write('newest_posts', $result, 'longterm');
        }

        return $result;
    }
}
```

```php
/**
 * @Cacheable
 */
class Post
{
    public function onGet($id)
    {
        // ...
        $this->body = $stmt->fetchAll(PDO::FETCH_ASSOC);

        return $this;
    }
```

メソッド実行

method invocation



インターセプト

intercepting method

複数のインターセプト

multiple interception

キャッシュがあればメソッドはコールされない

If the cache is warm the method is never called.

Miss

Hit

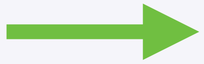
MethodInterceptor

```
class NullInterceptor implements MethodInterceptor
{
    public function invoke(MethodInvocation $invocation)
    {
        return $invocation->proceed();
    }
}
```

```
$invocation->proceed();
$invocation->getThis();       // object
$invocation->getMethod();     // ReflectionMethod
$invocation->getArguments(); // ArrayObject
```

```php
class CacheInterceptor implements MethodInterceptor
{
    private $cache;

    public function __construct(Cache $cache)
    {
        $this->cache = $cache;
    }

    public function invoke(MethodInvocation $invocation)
    {
        $id = get_class($invocation->getThis()) . serialize($invocation->getArguments());
        $saved = $this->cache->fetch($id);
        if ($saved) {
            return $saved;
        }
        $result = $invocation->proceed();
        $this->cache->save($id, $result);

        return $result;
    }
}
```

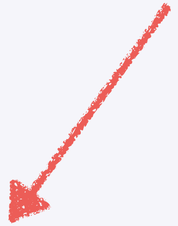
Cache Hit



Original method or next interceptor

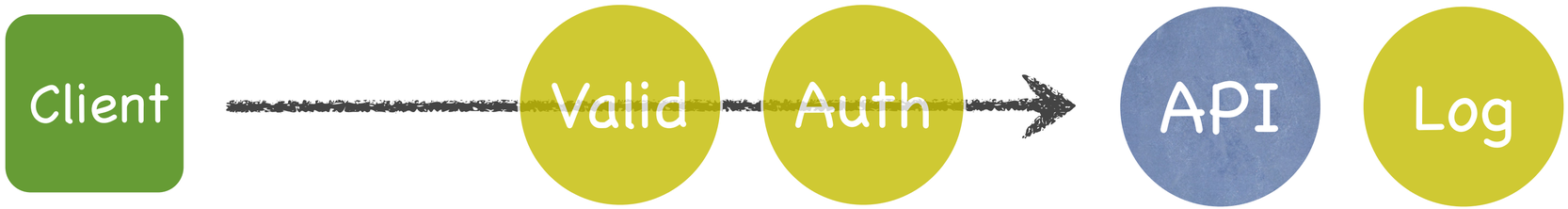
Bind The Interceptor with Matcher

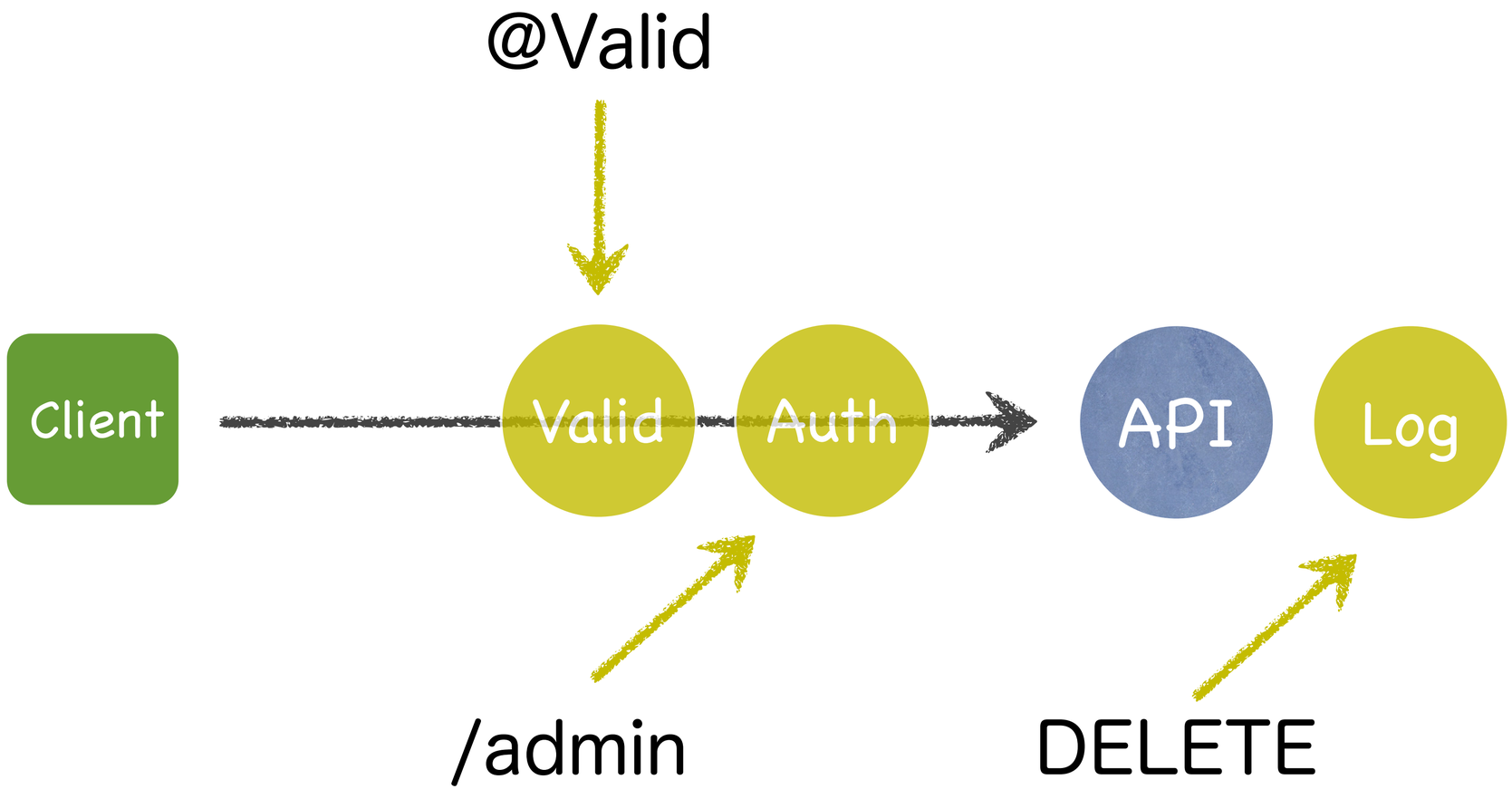
```php
$this->bindInterceptor(
    $this->matcher->annotatedWith(Cacheable::class), // class match
    $this->matcher->startsWith('onGet'),             // method match
    [CacheInterceptor::class]
);
```



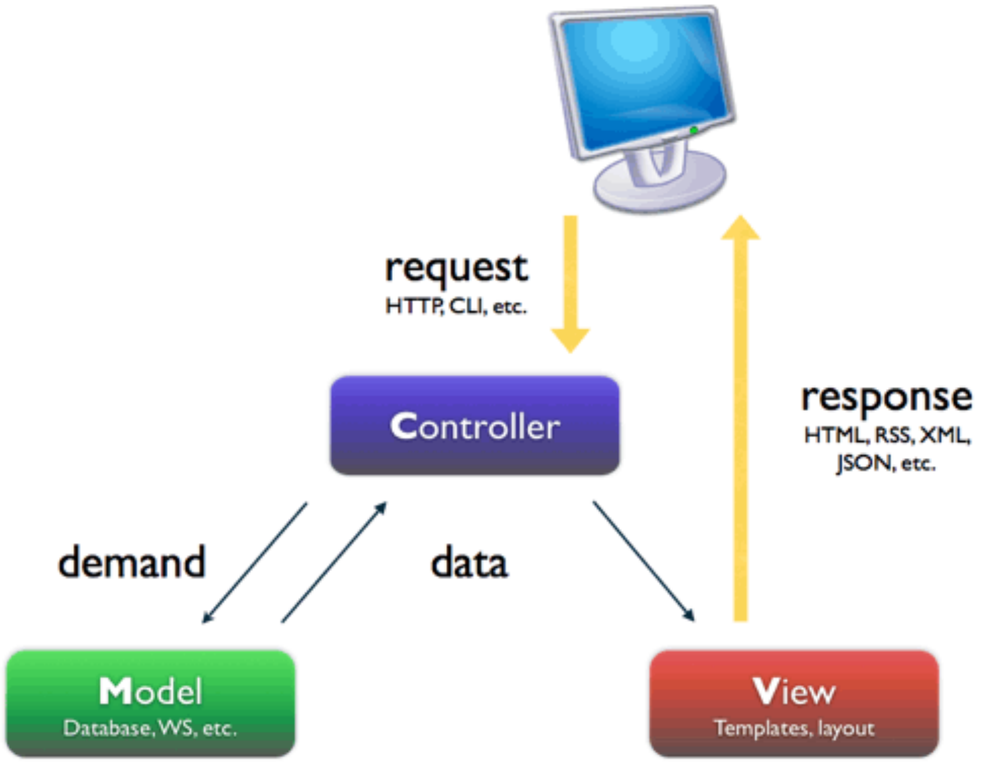
```php
/**
 * @Cacheable
 */
class Post
{
    public function onGet($id)
    {
        // ...
        $this->body = $stmt->fetchAll(PDO::FETCH_ASSOC);
```


Client
API


Client
Valid
Auth
API
Log

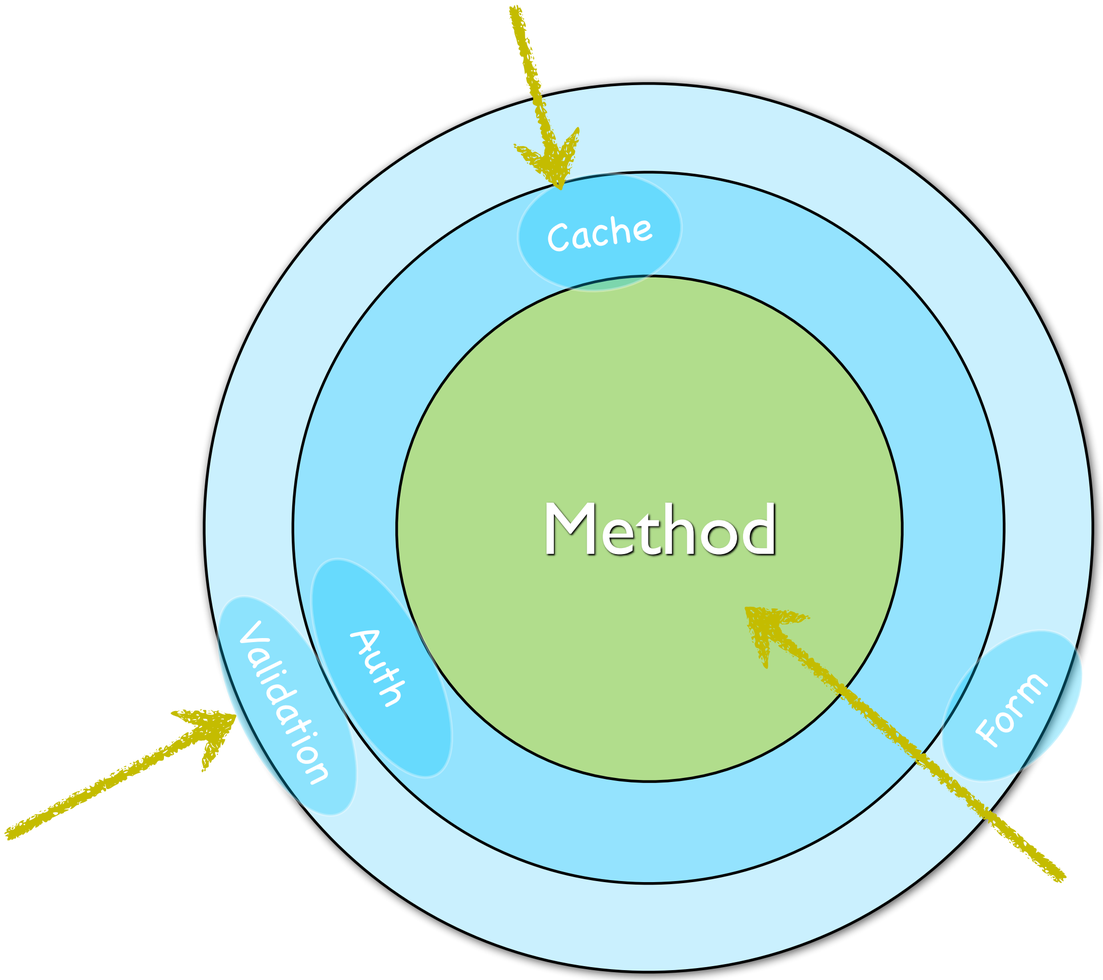

@Valid
Client
Valid
Auth
API
Log
/admin
DELETE

MVC, Is 3 enough ?



laying by context

可変レイヤリング



PHPでのAOP実装方法

How to implement AOP in PHP

- Proxy (1.x)
- Codegen (2.x)

継承クラスを動的に生成した型安全なインターセプト

Under the hood: Method interception sub class is created n order enable this interception and keep type safety.

```
class MyVendor_Weekday_Resource_App_Todo_OUNAVYk extends MyVendor\Weekday\Resource\App\Todo implements Ra
{
    private $isIntercepting = true;
    public $bind;
    public $methodAnnotations = 'a:4:{s:6:"onPost";a:2:{s:41:"Ray\\CakeDbModule\\Annotation\\Transactiona
    public $classAnnotations = 'a:1:{s:42:"BEAR\\RepositoryModule\\Annotation\\Cacheable";O:42:"BEAR\\Rep
    function onGet(int $id) : BEAR\Resource\ResourceObject
    {
        if (isset($this->bindings[__FUNCTION__]) === false) {
            return call_user_func_array('parent::' . __FUNCTION__, func_get_args());
        }
        if ($this->isIntercepting === false) {
            $this->isIntercepting = true;
            return call_user_func_array('parent::' . __FUNCTION__, func_get_args());
        }
        $this->isIntercepting = false;
        $invocationResult = (new \Ray\Aop\ReflectiveMethodInvocation($this, new \ReflectionMethod($this,
        $this->isIntercepting = true;
```

2nd framework: Aspect Oriented Framework

- AOPアライアンス AOP Alliance
- コンテキストによるレイヤリング Laying by context
- 型安全 Type safety
- Codegenで高速 Fast

Representational State Transfer


ロイ・フィールディング

BEAR.Resource

Hypermedia framework for object as a service

BEAR.Resource はオブジェクトがリソースの振る舞いを持つHypermediaフレームワークです。 クライアントーサーバー、統一インターフェイス、ステートレス、相互接続したリソース表現、レイヤードコンポーネント等の RESTのWebサービスの特徴をオブジェクトに持たせます。

既存のドメインモデルやアプリケーションの持つ情報を柔軟で長期運用を可能にするために、 アプリケーションをRESTセントリックなものにしAPI駆動開発を可能にします。

リソースオブジェクト

リソースとして振る舞うオブジェクトがリソースオブジェクトです。

- １つのURIのリソースが1クラスにマップされ、リソースクライアントを使ってリクエストします。
- 統一されたリソースリクエストに対応したメソッドを持ち名前付き引き数でリクエストします。
- メソッドはリクエストに応じてリソース状態を変更して自身 `$this` を返します。

```php
<?php
namespace MyVendor\Weekday\Resource\Page;

use BEAR\Resource\ResourceObject;

class Greeting extends ResourceObject
{
    public $code = 200;
    public $headers = [];
    public $body;

    public function onGet(string $name) : ResourceObject
    {
        $this->body = [
            'greeting' => 'Hello ' . $name
        ];

        return $this;
    }
}
```

```
curl -i -X DELETE http://127.0.0.1:8080/
```

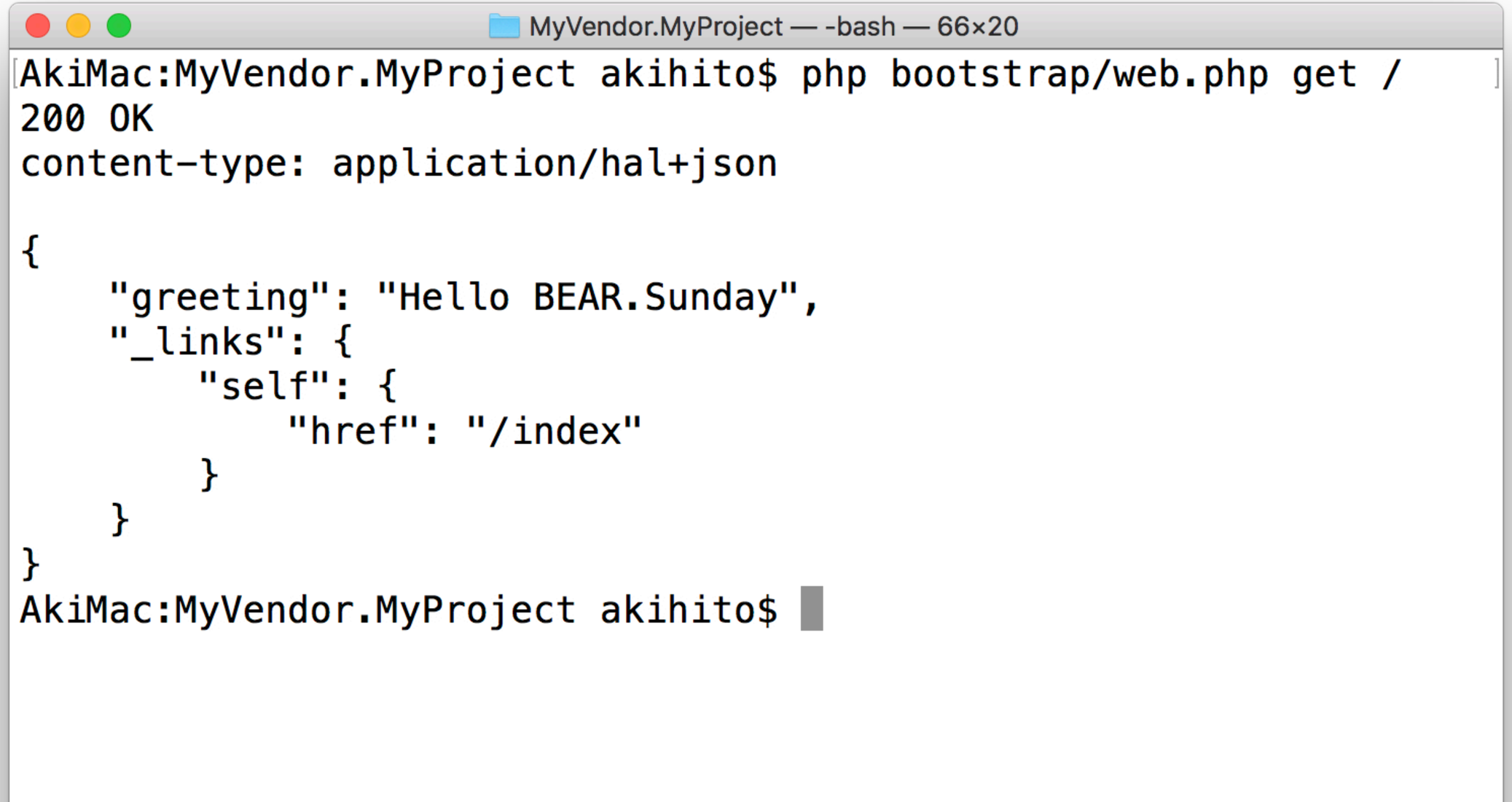
```
HTTP/1.1 405 Method Not Allowed
Host: 127.0.0.1:8080
Date: Wed, 31 May 2017 23:09:17 +0200
content-type: application/json

{
    "message": "Method Not Allowed"
}
```

HTTP

MyVendor.MyProject — -bash — 66×20

```
AkiMac:MyVendor.MyProject akihito$ curl -i http://127.0.0.1:8080/
HTTP/1.1 200 OK
Host: 127.0.0.1:8080
Date: Sat, 10 Mar 2018 01:19:10 +0100
Connection: close
X-Powered-By: PHP/7.1.10
content-type: application/hal+json

{
    "greeting": "Hello BEAR.Sunday",
    "_links": {
        "self": {
            "href": "/index"
        }
    }
}
```

Console


MyVendor.MyProject — -bash — 66×20
AkiMac:MyVendor.MyProject akihito$ php bootstrap/web.php get /
200 OK
content-type: application/hal+json
{
"greeting": "Hello BEAR.Sunday",
"_links": {
"self": {
"href": "/index"
}
}
}
AkiMac:MyVendor.MyProject akihito$

BEAR.Sunday

```php
class Index extends ResourceObject
{
    private $resource;

    public function __construct(ResourceInterface $resource)
    {
        $this->resource = $resource;
    }

    public function onGet()
    {
        $ro = $this->resource->uri('app://self/index')(['name' => 'BEAR']);
        $this->body = $ro['message'];

        return $this;
    }
}
```

PHP

```php
$app = (new Bootstrap)->getApp('MyVendor\MyProject', 'prod-hal-app');
$ro = $app->resource->get->uri('page://self/greeting')(['name' => 'BEAR']);

var_dump($ro->code);
//int(200)

print_r($ro->body) . PHP_EOL;
//(
//[greeting] => Hello BEAR
//)

var_dump($ro['greeting']);
//string(10) "Hello BEAR"
```

```php
echo $ro;
//{
//    "greeting": "Hello BEAR",
//    "_links": {
//    "self": {
//        "href": "/greeting?name=BEAR"
//        }
//    }
//}
```

```php
<?php
namespace MyVendor\Weekday\Resource\Page;

use BEAR\Resource\ResourceObject;

class Greeting extends ResourceObject
{
    public $code = 200;
    public $headers = [];
    public $body;

    public function onGet(string $name) : ResourceObject
    {
        $this->body = [
            'greeting' => 'Hello ' . $name
        ];

        return $this;
    }
}
```

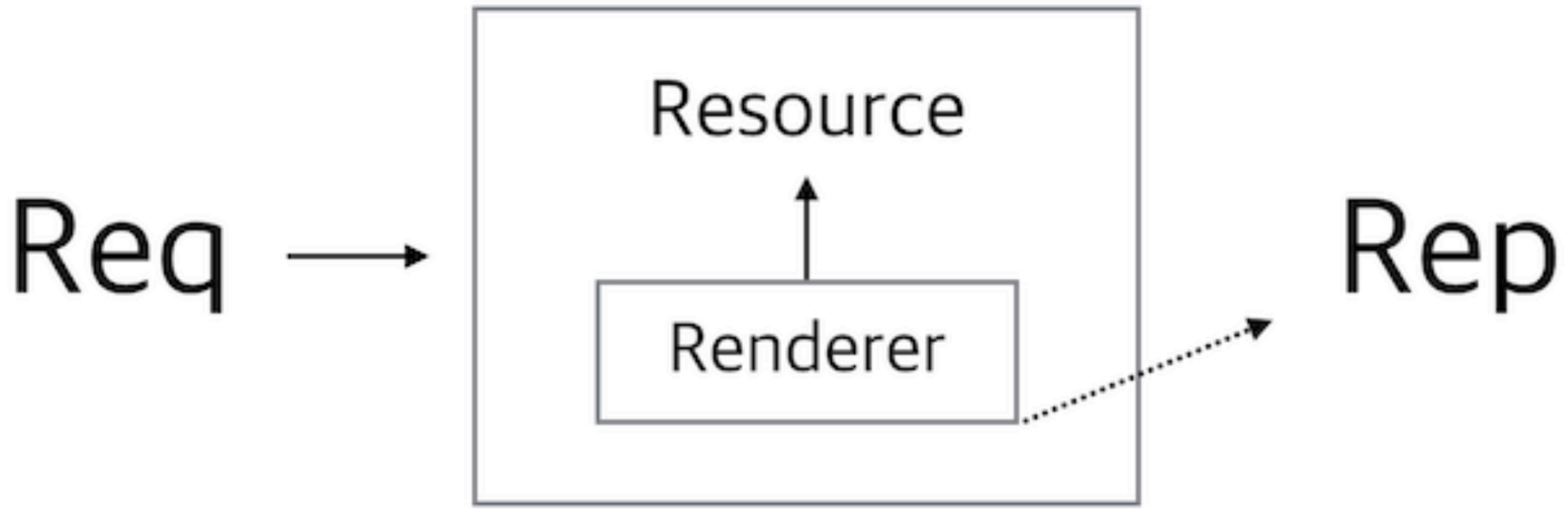
HTTP

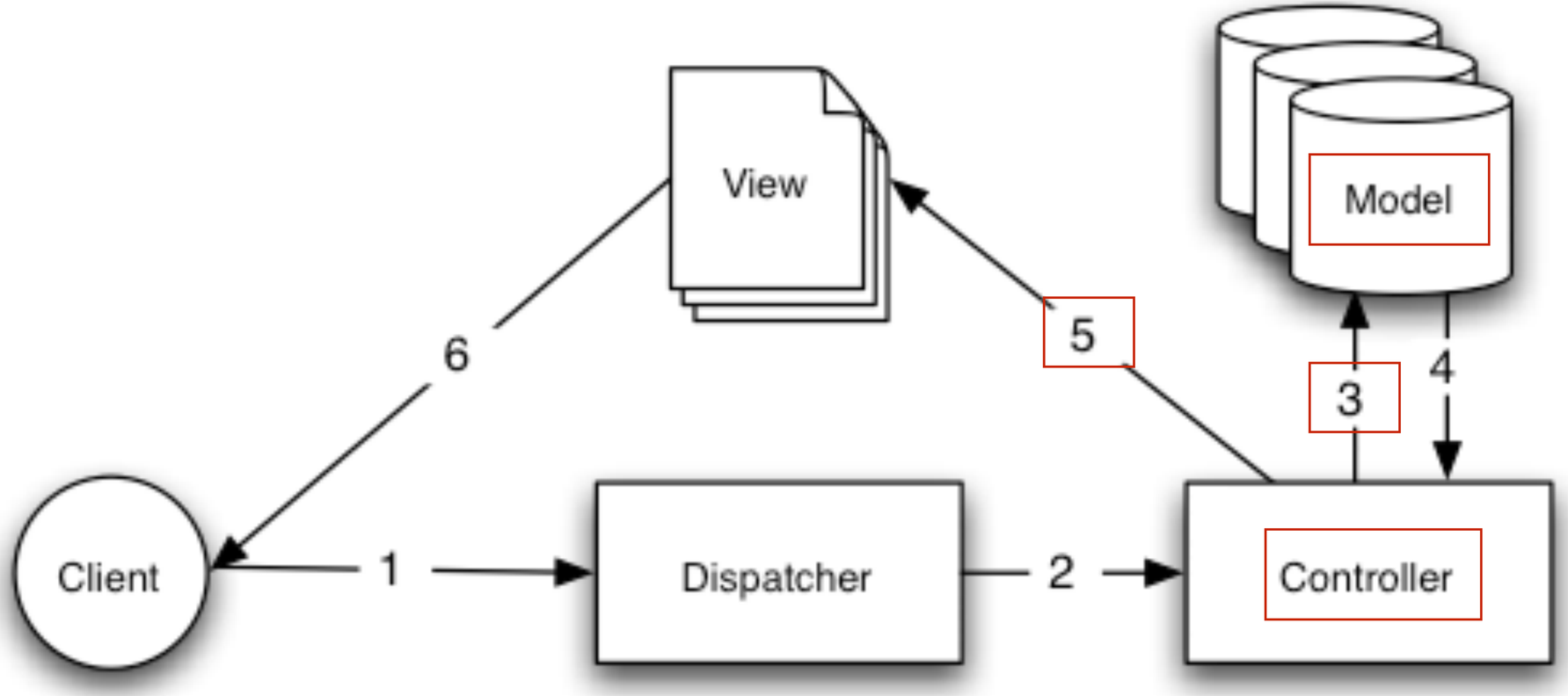
Console

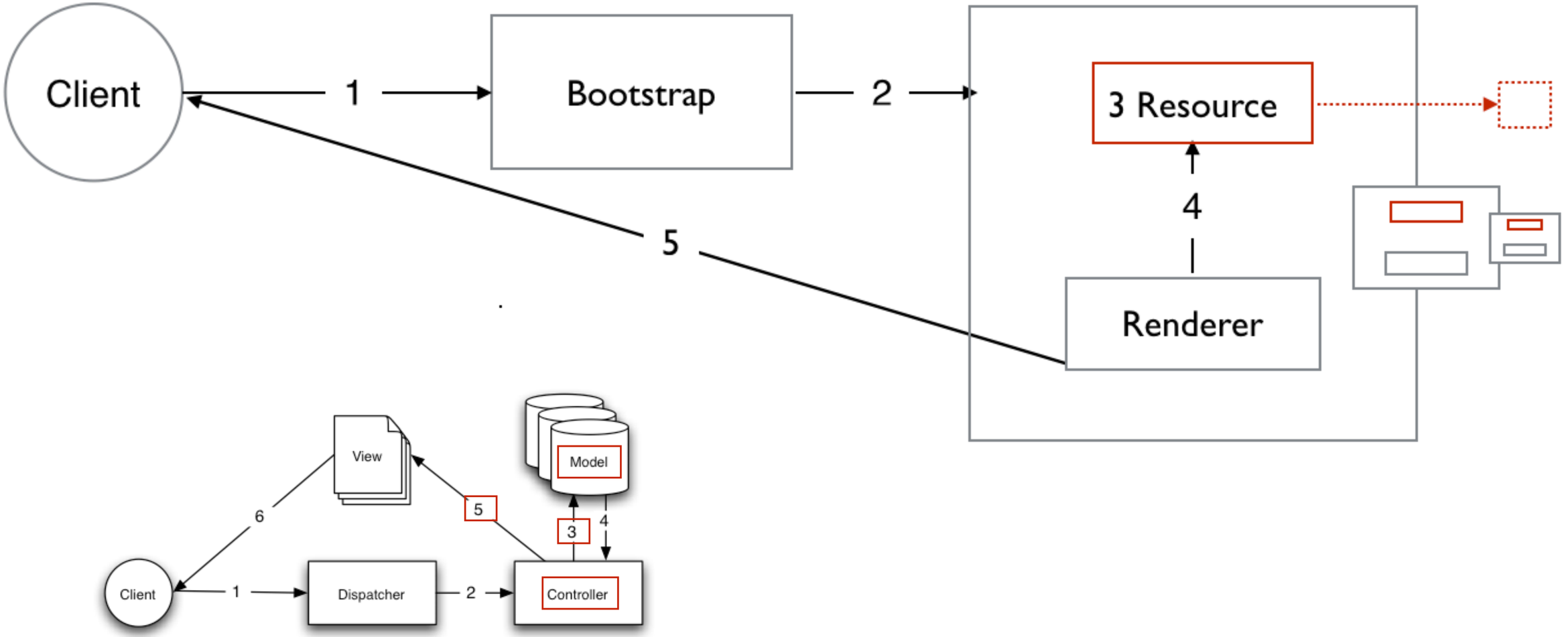
PHP

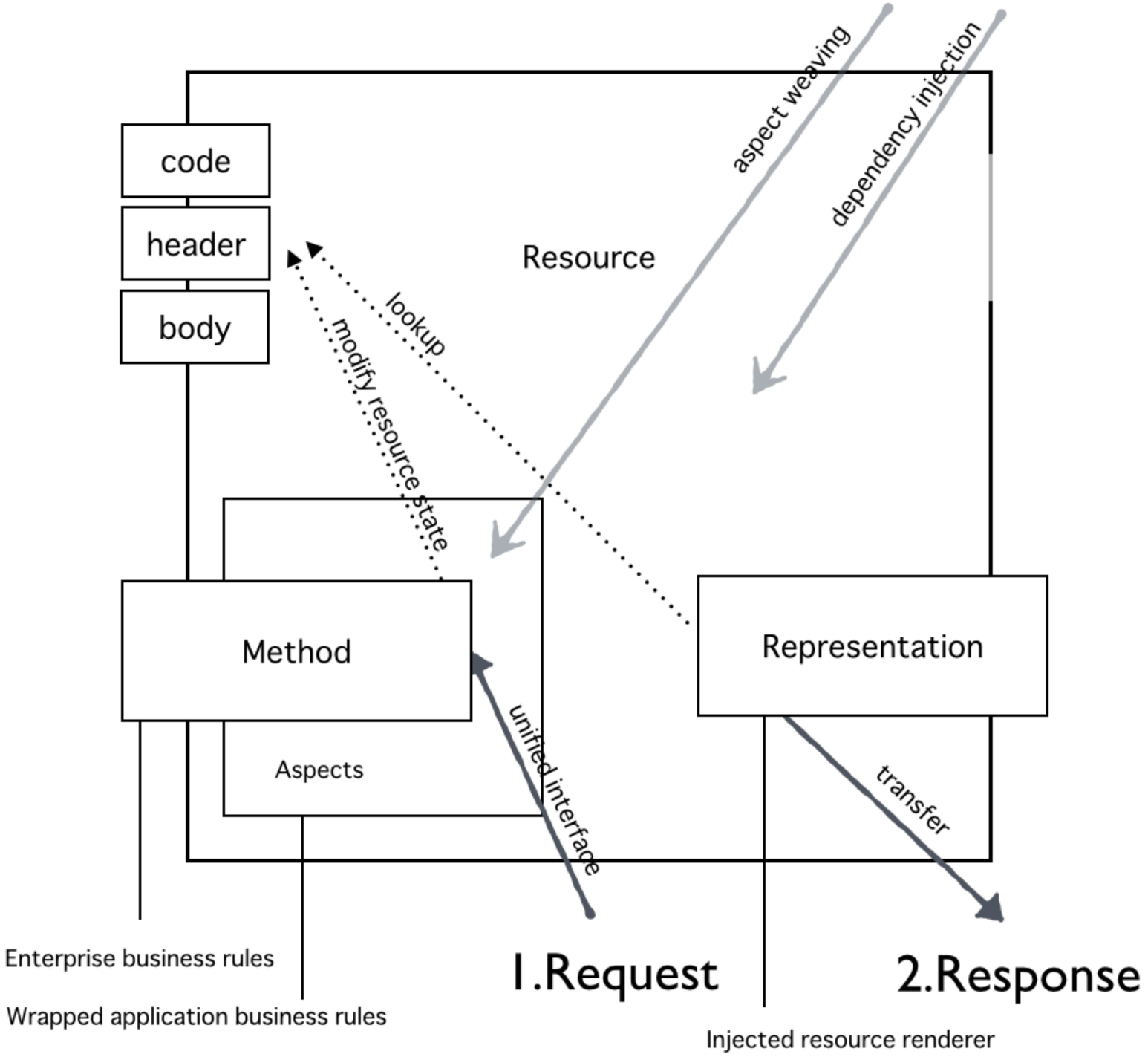
BEAR

Resource-Method-Representation




View
Model
6
5
3
4
Client
1
Dispatcher
2
Controller


Client
1
Bootstrap
2
3 Resource
4
Renderer
5
View
Model
6
5
3
4
Client
1
Dispatcher
2
Controller


code
header
body
Resource
aspect weaving
dependency injection
lookup
modify resource state
Method
Representation
Aspects
unified interface
transfer
Enterprise business rules
Wrapped application business rules
1.Request
2.Response
Injected resource renderer

@Embed

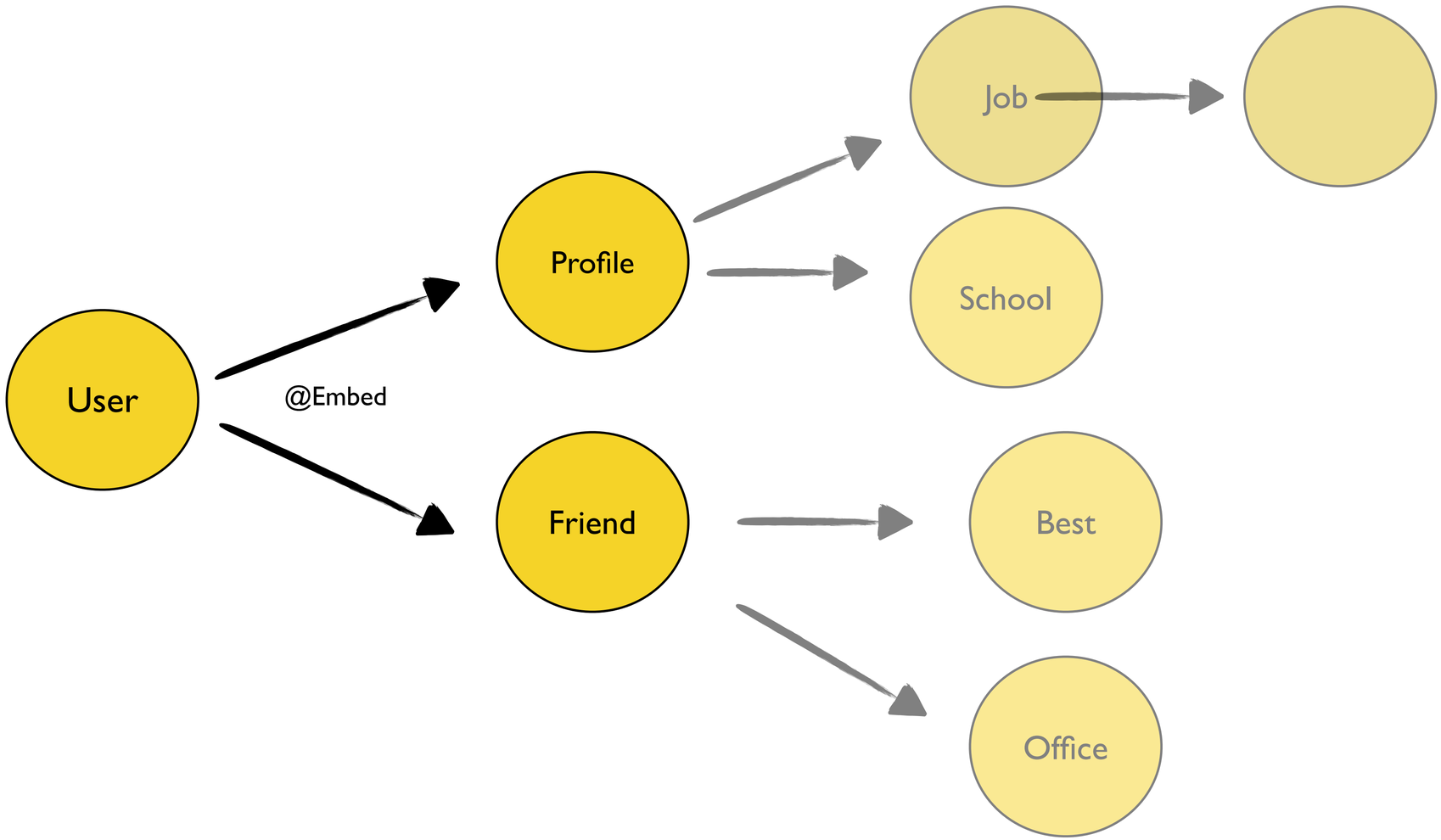
```
class News extends ResourceObject
{
    /**
     * @Embed(rel="weather", src="/weather{?date}")
     */
    public function onGet($date) : ResourceObject
    {
        $this->body += [
            'headline' => "40th anniversary of Rubik's Cube invention.",
            'sports' => "Pieter Weening wins Giro d'Italia."
        ];

        return $this;
    }
}
```

HAL: Hypertext Application Language

```
content-type: application/hal+json
```

```
{
    "headline": "40th anniversary of Rubik's Cube invention.",
    "sports": "Pieter Weening wins Giro d'Italia.",
    "_embedded": {
        "weather": {
            "today": "the weather of today is sunny",
            "_links": {
                "self": {
                    "href": "/weather?date=today"
                }
            }
        }
    },
    "_links": {
        "self": {
            "href": "/news?date=today"
        }
    }
}
```

Resource Graph



Inject Resource Parameter

```
/**
 * @ResourceParam(param="name", uri="app://self//login#nickname")
 */
public function onGet(string $name) : ResoureObject
{
}
```

Method base cache update

```php
/**
 * @Cacheable
 */
class Todo
{
    public function onGet(string $id) : ResoureObject
    {
        // read
    }

    public function onPut(string $id, string $name) : ResoureObject
    {
        // update
    }
```

304 Not Modified

```php
if ($app->httpCache->isNotModified($_SERVER)) {
    http_response_code(304);
    exit(0);
}
```

```
curl -i -H 'If-None-Match: 3652022809' http://127.0.0.1:8080/
```

```
HTTP/1.1 304 Not Modified
Host: 127.0.0.1:8080
Date: Wed, 31 May 2017 23:56:31 +0200
```

Content-Negotiation

```
$available = [
    'Accept' => [
        'text/html' => 'html-app',
        'cli' => 'cli-html-app'
    ],
    'Accept-Language' => [
        'ja' => 'ja',
        'ja-JP' => 'ja',
        'en' => 'en',
        'en-US' => 'en',
        'en-GB' => 'en'
    ]
];
$accept = new Accept($available);
list($context, $vary) = $accept($_SERVER);
```

prefer English

```
curl -H 'Accept-Language: en' http
```

prefer Japanese

```
curl -H 'Accept-Language: ja' http
```

OPTION method

```
curl -i -X OPTIONS http://127.0.0.1:8080/
```

```
HTTP/1.1 200 OK
Host: 127.0.0.1:8080
Date: Wed, 31 May 2017 23:04:50 +0200
Connection: close
X-Powered-By: PHP/7.1.4
Content-Type: application/json
Allow: GET, POST

{
    "GET": {
        "summary": "Todos list",
        "description": "Returns the todos list specified by status",
        "request": {
            "parameters": {
                "status": {
                    "type": "string",
                    "description": "todo status"
```

JsonSchema Validation

```php
class User extends ResourceObject
{
    /**
     * @JsonSchema(schema="user.json")
     */
    public function onGet()
    {
        $this->body = [
            'firstName' => 'mucha',
            'lastName' => 'alfons',
            'age' => 12
        ];

        return $this;
    }
}
```

```json
{
  "type": "object",
  "properties": {
    "firstName": {
      "type": "string",
      "maxLength": 30,
      "pattern": "[a-z\\d~+-]+"
    },
    "lastName": {
      "type": "string",
      "maxLength": 30,
      "pattern": "[a-z\\d~+-]+"
    }
  },
  "required": ["firstName", "lastName"]
}
```

todo

/todo/{id}

GET

Return a todo

Parameters

Name	Type	Description
id	string	todo id

Required id

Schema

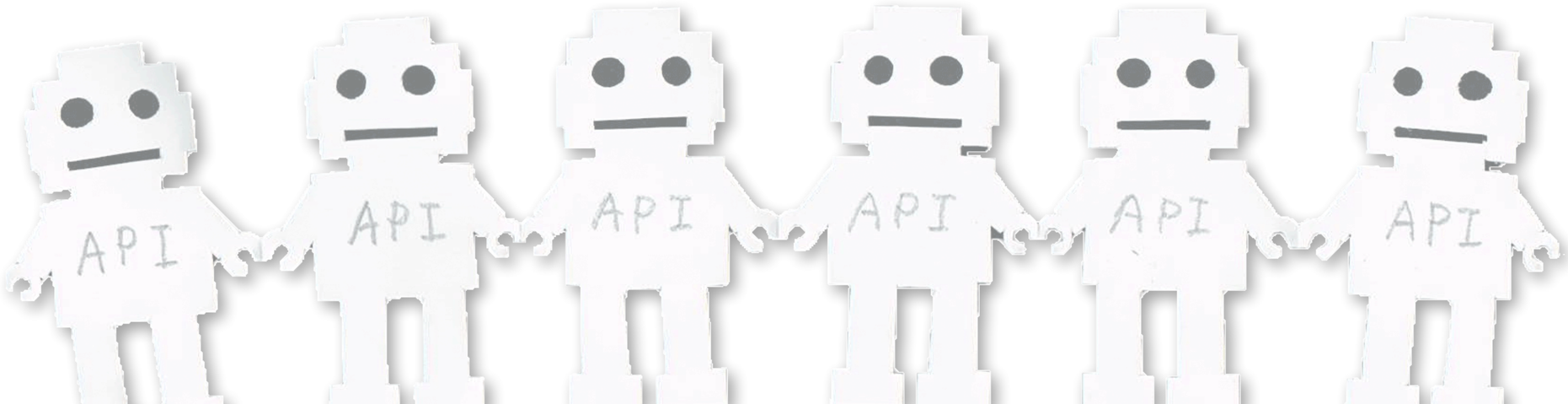
todo.json

Property	Type	Description	Constraints
id	string	The unique identifier for a todo.	
title	string	The title of the todo	minLength 1 maxLength 255
status	string	2 if task has been completed	enum ["1","2"]
status_name	string	The name of the status	maxLength 255

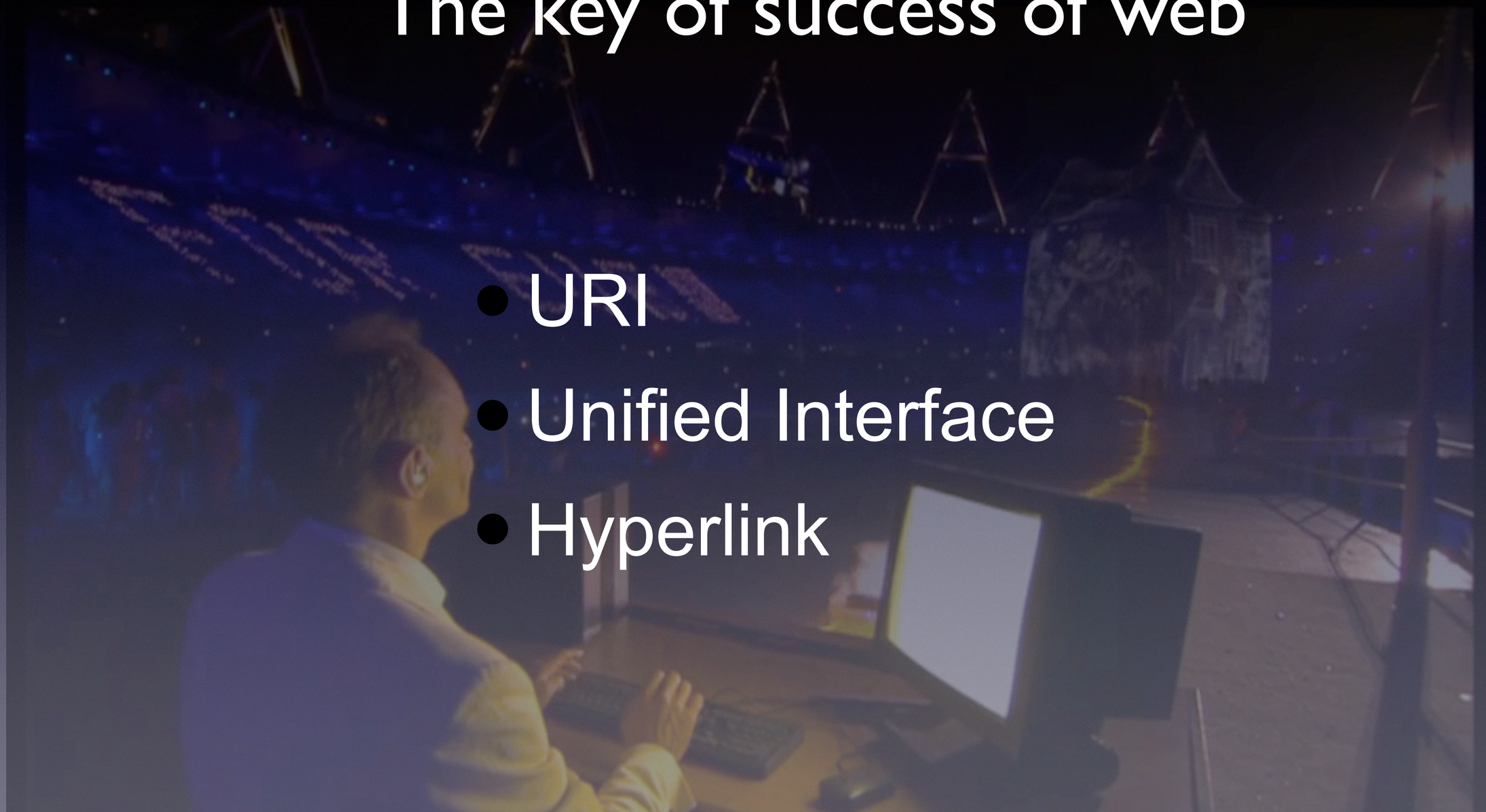
Hypermedia

```
{
    "drink": "latte",
    "order_id": 1865641357,
    "_links": {
        "self": {
            "href": "/order?drink=latte"
        },
        "payment": {
            "href": "/payment?order_id=1865641357"
        }
    }
}
```

Hypermedia API



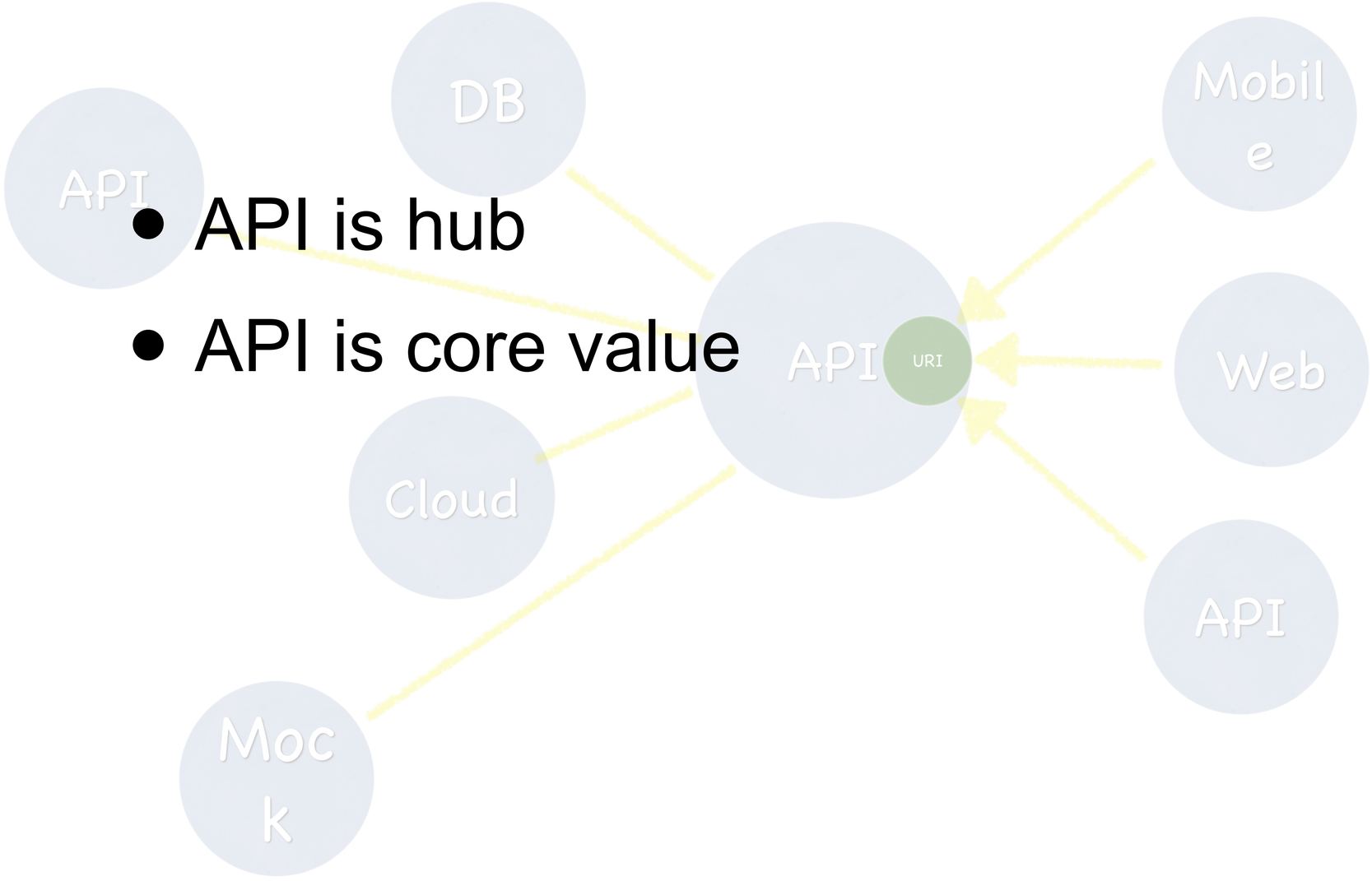
The key of success of web



- URI
- Unified Interface
- Hyperlink

API駆動開発

API driven development



- API is hub
- API is core value

self/index

pen online

Feature News Blog Tokyo Guide Shopping Collection

Pen 最新号

search

app://self/article/feature/top/loading

ウォーホルたちの名作に、目が眩む。

app://self/article/feature/top

Feature

新アメリカンスタイルの立役者

シャネル、香りの革命史。

Ranking

app://self/article/ranking

2

シャネル、香りの革命史。

3

新色ネイビーをの
た、「ブリーフィ

3rd framework: REST Framework

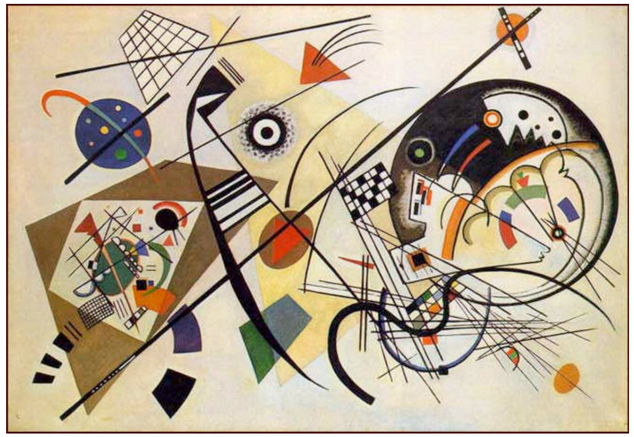
- RESTfulなリソースコンテナ RESTful container
- HTTP標準 HTTP standard
- 再利用性 Reusability
- 強力なキャッシュ Cache-friendly

Performance

- *annotation ? dependency injection ?*
  method interception ? DSL ? named parameter ?
- 高速 Fast
  - オブジェクトグラフキャッシュ Object graph cache
  - Codegen

Abstraction frameworks

- DSL
- Annotation
- URI
- Interface
- Aspects
- Hypermedia



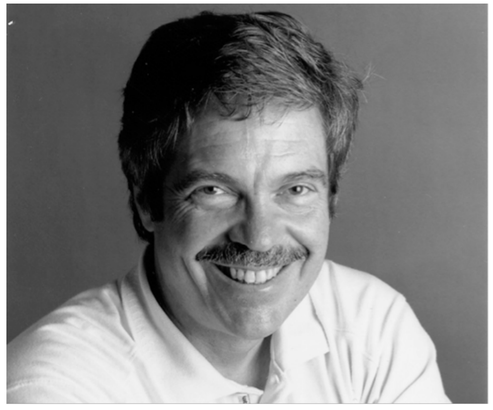
Connecting frameworks

- DI - オブジェクトを結ぶ Connecting objects
- AOP - Domain LogicとApplication Logicを結ぶ
  Connecting domain logic and application logic
- REST -リソースを結ぶ Connecting resources

AOP (Gregor Kiczales)

DI (Martin Fowler)

REST (Roy Fielding)



OOP (Allan Kay)

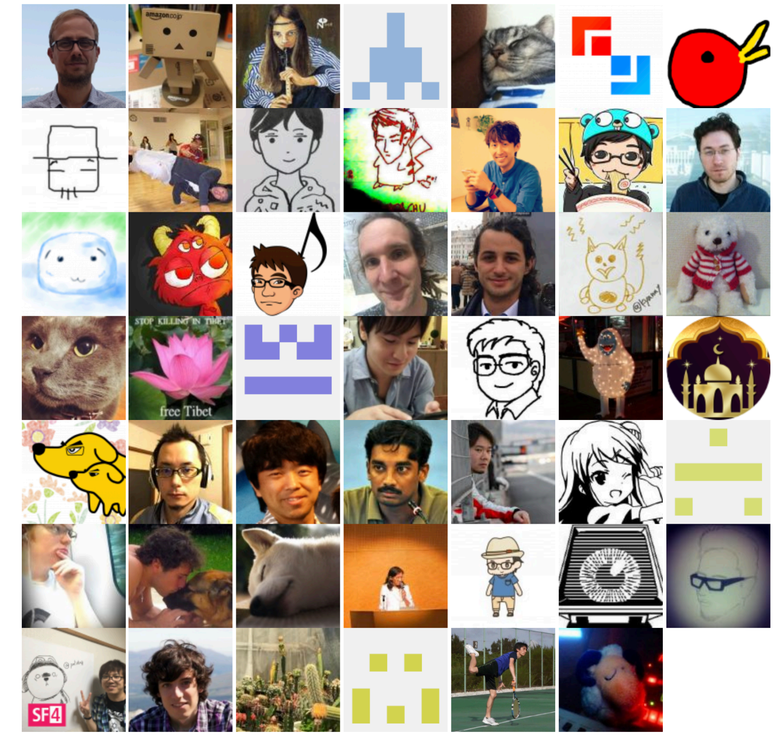
Annotation (Anders Hejlsberg)

Guice (Bob Lee)

Why BEAR.Sunday ?

- RESTfullクリーンアーキテクチャ RESTful clean architecture
- API駆動開発 API driven development
- 再利用性 Reusability
- パフォーマンス Performance
- 後方互換 NO BC break

Contributors




DEDICATED
TO GOLF

jose_zap
@jose_zap

フォローする

@bartosz_golek @bar4bor Take a look at AuraPHP and Bear.Sunday projects for inspiration. They are really good at that

2014年5月

Nate Abele
@nateabele

フォローする

In summary, if I were forced to choose between @laravel & @cakephp, I would choose @BEARSunday. /cc @savant

2013年10月19日 06:40

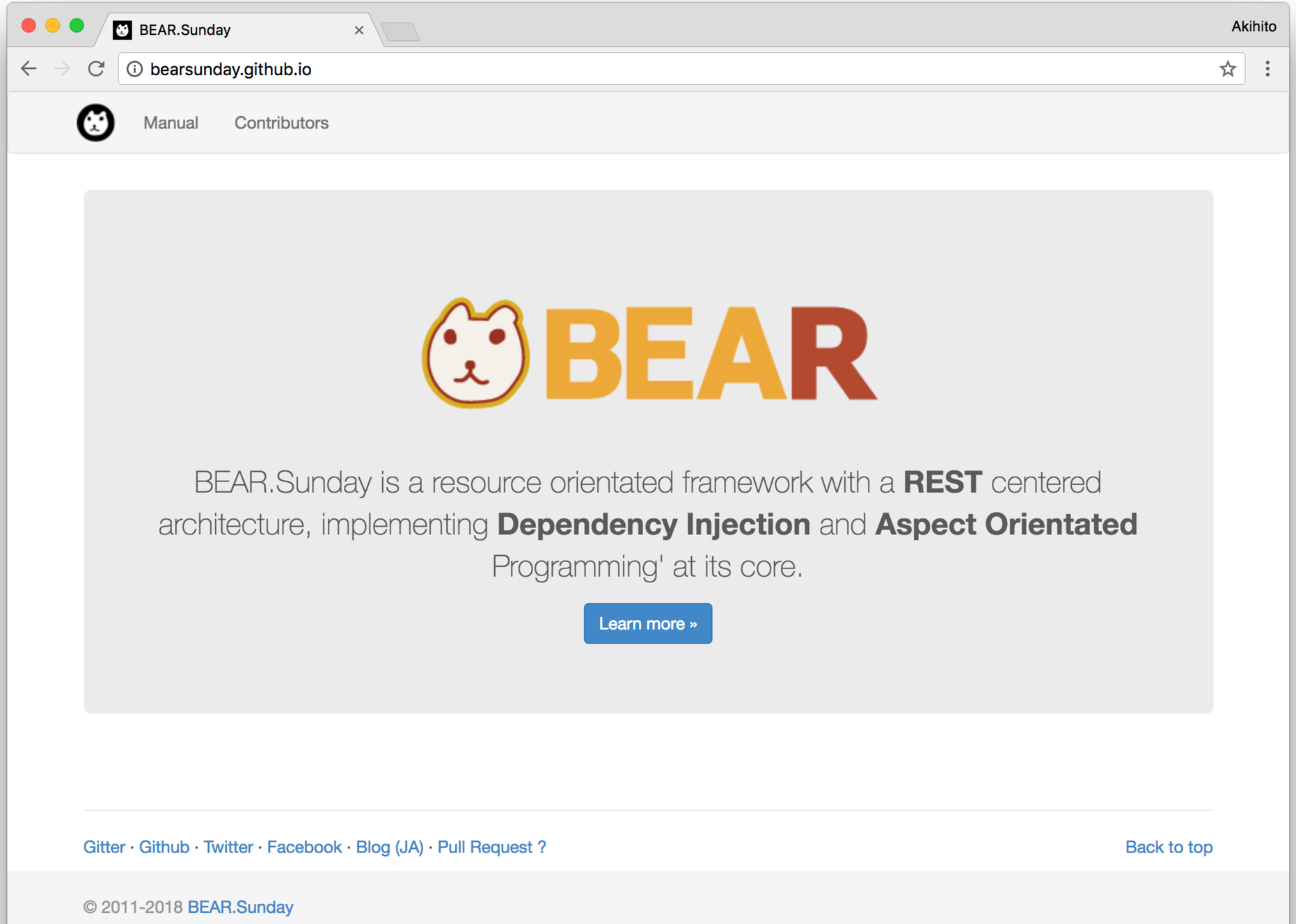
Chris Hartjes
@grmpyprogrammer

フォローする

Looking to try a new PHP framework with a non-MVC approach? BEAR.Sunday looks pretty good and is 1.0! twitter.com/BEARSunday/sta…

2015年5月31日 23:20

7 5


BEAR.Sunday
Akihito
bearsunday.github.io
Manual
Contributors
BEAR
BEAR.Sunday is a resource orientated framework with a **REST** centered architecture, implementing **Dependency Injection** and **Aspect Orientated** Programming' at its core.
Learn more »
Gitter · Github · Twitter · Facebook · Blog (JA) · Pull Request ?
Back to top
© 2011-2018 BEAR.Sunday

@koriym

@BEARSunday