

# 站在巨人的肩膀上 ——使用Symfony开发你的下一个项目

洪涛 PHPCon2017

# 个人简介

洪涛，现任携车网 CTO，喜好技术研究和分享，曾受邀参加多个技术大会并担任讲师。

从2009年开始接触Symfony，多次完整经历了使用Symfony开发的项目，积累了大量相关经验，曾在慕课网制作了《洪大师带你解读Symfony2框架》系列视频课程。

最近关注的领域是产品和技术的深度结合，以及如何用技术推动公司业务发展。

问：请问如何评价Symfony这个框架？

答：『优点：强大。缺点：太强大』

—— 来自某知乎网友

1. Symfony是什么？
2. Symfony的一些特性和功能介绍
3. 如何开始使用Symfony
4. 我使用 Symfony 8年来的一些感悟和理解

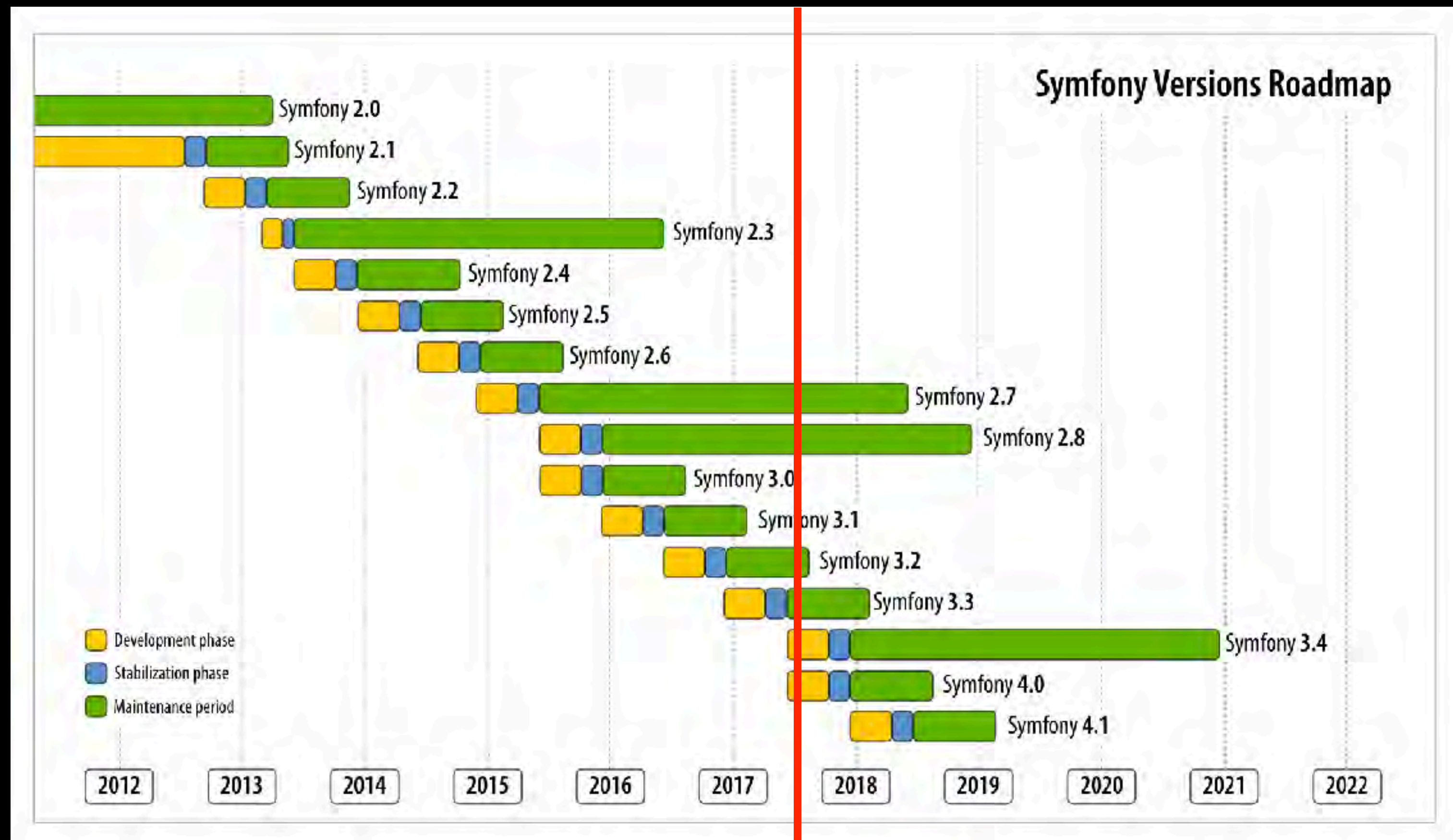
# 1. Symfony是什么

# Symfony

- Github上最活跃的PHP项目之一
- 大量来自官方和开源社区的支持
- 大量成熟的功能插件
- 以此构建并衍生出的生态环境已经影响了整个PHP开发圈

# Symfony的版本

- Symfony 1.x
  - 05年发布, 12年停止维护
- Symfony 2.x~3.x~4.x
  - 11年发布
  - 最新LTS版本2.8
  - 非LTS版本3.2



## 2. 特性和功能介绍



如何做CRUD?

# 只要三步？

1. 定义类
2. 写几个基本的方法
3. 执行几个命令

# 定义Entity

```
class MaintenanceProduct extends BaseEntity
{
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="价格", "default"=0})
     */
    protected $price = 0;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="市场价", "default"=0})
     */
    protected $market_price = 0;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="最低零售价", "default"=0}) ...*/
    protected $min_reference_price = 0;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="最高零售价", "default"=0}) ...*/
    protected $max_reference_price = 0;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="批发价", "default"=0}) ...*/
    protected $normal_price = 0;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="工时费单价", "default"=0}) ...*/
    protected $work_hour_price = 0;

    /**
     * @ORM\Column(type="integer", options={"comment"="施工工数", "unsigned"=true}) ...*/
    protected $work_hours = 1;

    /**
     * @ORM\Column(type="string", length=50, options={"comment"="品牌名称"}) ...*/
    protected $name;

    /**
     * @ORM\Column(type="boolean", options={"comment"="是否可用", "default"=0}) ...*/
    protected $enable = true;
}
```

# 定义AdminClass

```
namespace AppBundle\Admin\Maintenance;

use ...

class MaintenanceProductAdmin extends BaseAdmin
{
    /**
     * @param DatagridMapper $datagridMapper
     */
    protected function configureDatagridFilters(DatagridMapper $datagridMapper)
    {
    }

    /**
     * @param ListMapper $listMapper
     */
    protected function configureListFields(ListMapper $listMapper)
    {
    }

    /**
     * @param FormMapper $formMapper
     */
    protected function configureFormFields(FormMapper $formMapper)
    {
    }

    /**
     * @param ShowMapper $showMapper
     */
    protected function configureShowFields(ShowMapper $showMapper)
    {
    }
}
```



[illegible]

**+ 江渡橋**

ID

Price

Market Price

Mark-Down Price

Mark-Round

Name

Credits

a-Coupon Price

a-Coupon Mark-Down

a-Coupon Mark-Round Price

Has Material

Type

维修项



ID \*

Price \*

Market Price \*

Work Hour Price \*

Work Hours \*

Name \*

Enable



Is Certain Price



# 零代码实现复杂表关系的CRUD

Admin Edit "Perferendis fa" x

sonata.vm/admin/sonata/news/post/2/edit

Sonata Project

Posts Perferendis facere pariatur eveniet molestiae.

Options

Enabled ☒

Image

No selection

List Add new Delete

Publication start date \*

2014-04-30

Comments closed at

Comments enabled ☒

April 2014

Su	Mo	Tu	We	Th	Fr	Sa
30	31	1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	1	2	3
4	5	6	7	8	9	10

```
/**
 * This type is used to render an hidden input text and 3 links
 * - an add form modal
 * - a list modal to select the targeted entities
 * - a clear selection link.
 */
class ModelTypeList extends AbstractType
{
    /**
     * {@inheritdoc}
     */
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $builder
            ->resetViewTransformers()
            ->addViewTransformer(new ModelToIdTransformer($options['model_manager'], $options['class']));
    }
}
```

SonataAdminBundle:ModelTypeList

# 为什么能做到?

数据库

```
'work_hours' smallint(5) unsigned NOT NULL COMMENT '总工时',  
'name' varchar(50) COLLATE utf8_unicode_ci NOT NULL COMMENT '维修项名',  
'enable' tinyint(1) NOT NULL DEFAULT '1' COMMENT '是否可用',  
'is_show' tinyint(1) NOT NULL DEFAULT '1' COMMENT '是否显示',  
'is_certain_price' tinyint(1) NOT NULL DEFAULT '0' COMMENT '是否固定价格',  
'is_certain_work_hour' tinyint(1) NOT NULL DEFAULT '0' COMMENT '是否固定工时'
```

PHP代码

```
/**  
 * @ORM\Column(type="boolean", options={"comment"="是否可用", "default"=1})  
 */  
protected $enable = true;
```

用户操作  
前端页面

更换燃油滤清器

Enable  
☒

Is Certain Price

```
/**  
 * Type that maps an SQL boolean to a PHP boolean.  
 *  
 * @since 2.0  
 */  
class BooleanType extends Type  
{  
    ...  
}
```

DoctrineBundle  
BooleanType

```
class BooleanType extends AbstractType  
{  
    const TYPE_YES = 1;  
    const TYPE_NO = 2;  
  
    /**  
     * {@inheritdoc}  
     */  
    public function buildForm(FormBuilderInterface $builder, array $options)  
    {  
        if ($options['transform']) {  
            $builder->addModelTransformer(new BooleanTypeToBooleanTransformer());  
        }  
    }  
}
```

SonataCoreBundle  
BooleanType



# 数据库ORM - Doctrine

# 使用ORM操作数据库

```
$user = new User();  
$user->setCity("shanghai");  
$user->setNickname("scourgen");  
  
$this->getEntityManager()->persist($user);  
$this->getEntityManager()->flush();
```

新增

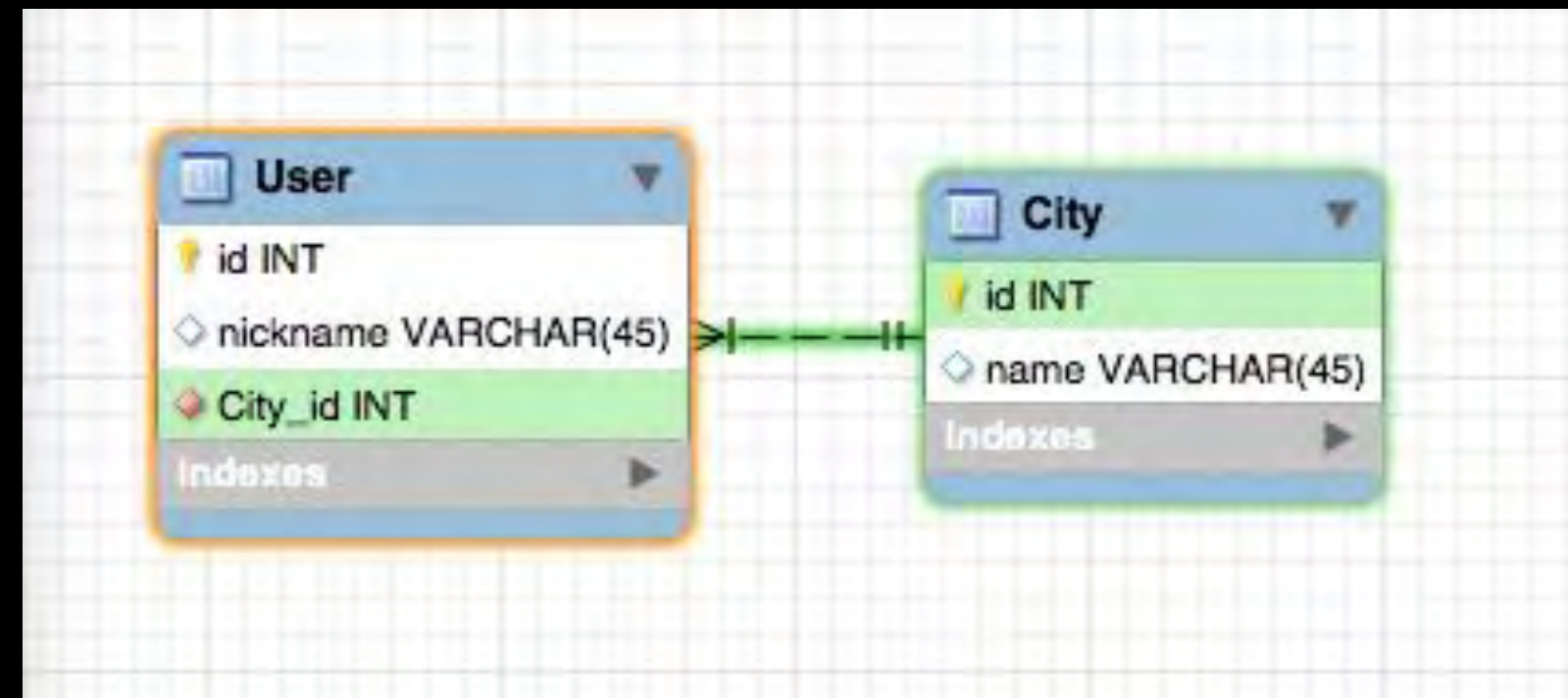
```
$user = $this->getUserRepository()->findOneBy(["nickname"=>"scourgen"]);  
$user->setNickname("scourgen xxxx");  
$this->getEntityManager()->flush();
```

更新

```
$user = $this->getUserRepository()->findOneBy(["nickname"=>"scourgen"]);  
$this->getEntityManager()->remove($user);  
$this->getEntityManager()->flush();
```

删除

# 自动获取关联表的数据



```
$user = $this->getUserRepository()->findOneBy(["nickname"=>"scourgen"]);  
$user->getCity()->getName();
```

- 当执行第二句时，Doctrine会自动帮我们生成一个包含join的sql语句并运行，然后把city表里的name字段返回给我们



- 原生SQL

```
$sql='select *  
      from user u  
      left join books b  
      where u.id = b.user_id  
      and u.id = 1  
      .....#此处太长省略572字';
```

- 面向对象的类操作

```
$user = $this->getDoctrine()  
      ->getManager()  
      ->getRepository('ScourgenWebBundle:User')->findOneById(1);  
  
echo $user->getBooks()[0]->getAuthores()[0]->getName();
```

- Doctrine  
Query Language

```
public function getDomainIps()  
{  
    return $this->getEntityManager()->createQueryBuilder()  
        ->select( select: 'i')  
        ->from( from: 'Ip', alias: 'i')  
        ->where( predicates: 'i.domain != :domain')  
        ->andWhere('i.enabled = :enabled')  
        ->setParameter( key: 'domain', value: '')  
        ->setParameter( key: 'enabled', value: true)  
        ->getQuery()  
        ->getResult()  
    ;  
}
```

```
public function getAllActivedUser(){  
    $users = $this->getEntityManager()->createQueryBuilder();  
  
    $this->setSelects($users);  
    $this->setFroms($users);  
    $this->setJoins($users);  
    $this->setLimits($users);  
  
    $users->getQuery()  
        ->getResult();  
  
    return $users;  
}
```

```
$user = $this->getDoctrine()  
      ->getManager()  
      ->createQuery(' SELECT u, b, a  
                      FROM ScourgenWebBundle:User u  
                      left join u.books b  
                      left join b.authores a  
                      where u.id=1')  
      ->getResult();  
  
echo $user->getBooks()[0]->getAuthores()[0]->getName();
```



# Doctrine很慢吗？

```
$this->getEntityManager()->persist($user);  
$this->getEntityManager()->flush();
```

- Doctrine能把数据库操作进行预处理和合并：  
persist(): 把操作“暂存”起来  
flush(): 把各种操作进行合并计算，得到结果后一并在数据库里操作并commit
- Doctrine能把对数据库的操作规范并统一，方便集中优化SQL语句和表结构

Annotation

```

class MaintenanceProduct extends BaseEntity
{
    /**
     * @ORM\Id
     * @ORM\Column(type="integer")
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="价格", "default"=0})
     */
    protected $price = 0;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="市场价", "default"=0})
     */
    protected $market_price = 0;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="最低参考价", "default"=0}) ...*/
    protected $min_reference_price = 0;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="最高参考价", "default"=0}) ...*/
    protected $max_reference_price = 0;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="零售价", "default"=0}) ...*/
    protected $normal_price = 0;

    /**
     * @ORM\Column(type="decimal", precision=18, scale=2, options={"comment"="工时费单价", "default"=0}) ...*/
    protected $work_hour_price = 0;

    /**
     * @ORM\Column(type="integer", options={"comment"="工时数", "unsigned"=true}) ...*/
    protected $work_hours = 1;

    /**
     * @ORM\Column(type="string", length=50, options={"comment"="维修描述"}) ...*/
    protected $name;

    /**
     * @ORM\Column(type="boolean", options={"comment"="是否可用", "default"=1}) ...*/
    protected $enable = true;
}

```



```
/**
 * @Route("/get_city_by_location", name="get_city_by_location")
 * @Method({"GET"})
 * @ApiDoc(
 *     section="Area",
 *     description="根据坐标获取城市ID",
 *     parameters={
 *         {"name"="location", "dataType"="string", "required"=true, "description"="坐标", "format"="39.984154,116.307490"},
 *     },
 *     tags={
 *         "doing"
 *     },
 * )
 */
public function getCityByLocation(Request $request)
{
```

- 路由，该方法可以通过/get\_city\_by\_location访问到
- 访问类型，限制了只能通过GET访问
- 定义了API参数，能够生成API调试页面

对于Controller Annotation还可以实现：  
定义Cache、定义访问权限、HTTP参数的预处理.....



一般情况下，Controller里，真正业务逻辑所占的比例

[illegible]



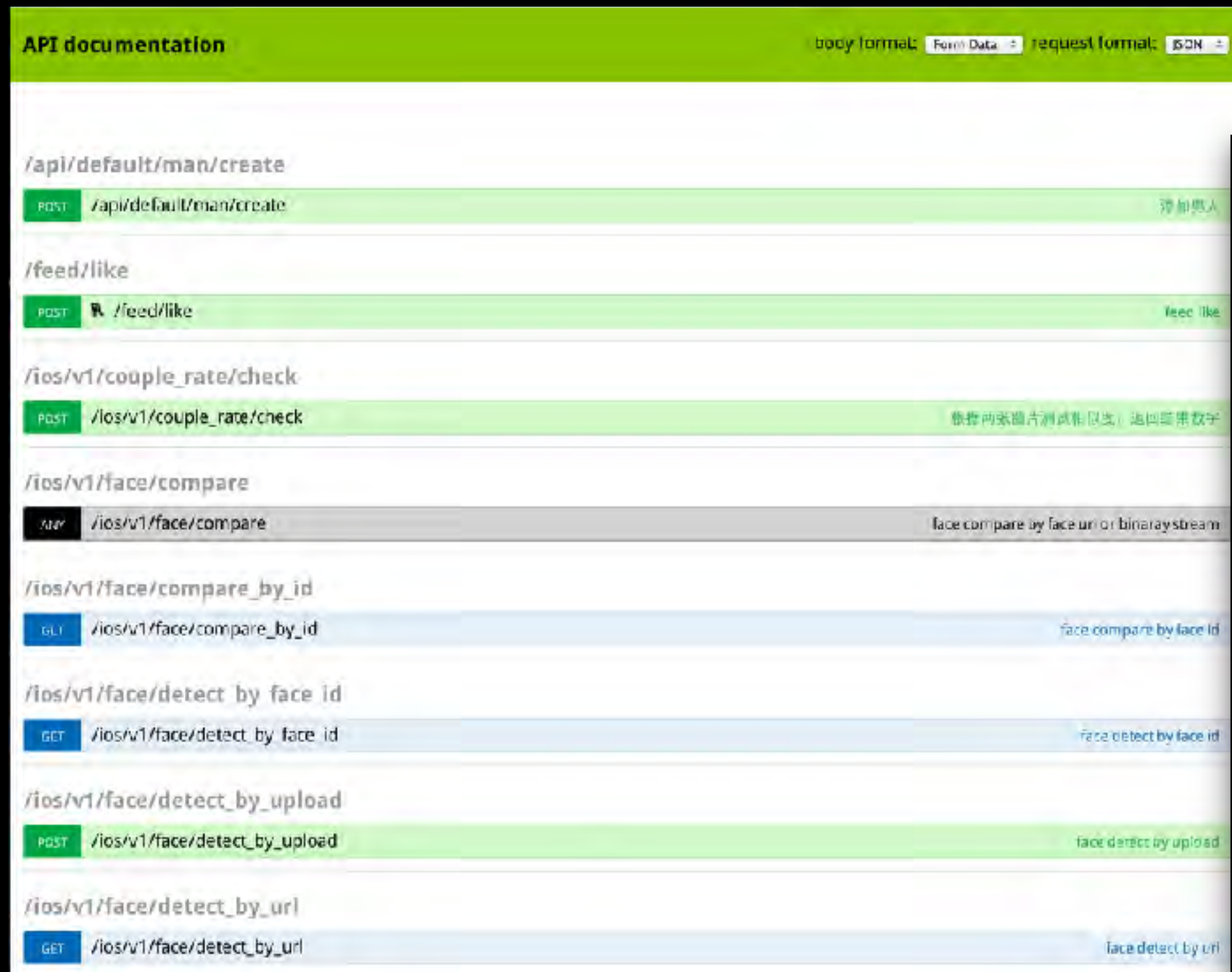
快速进行接口开发

# 使用Annotation快速定义 API接口文档

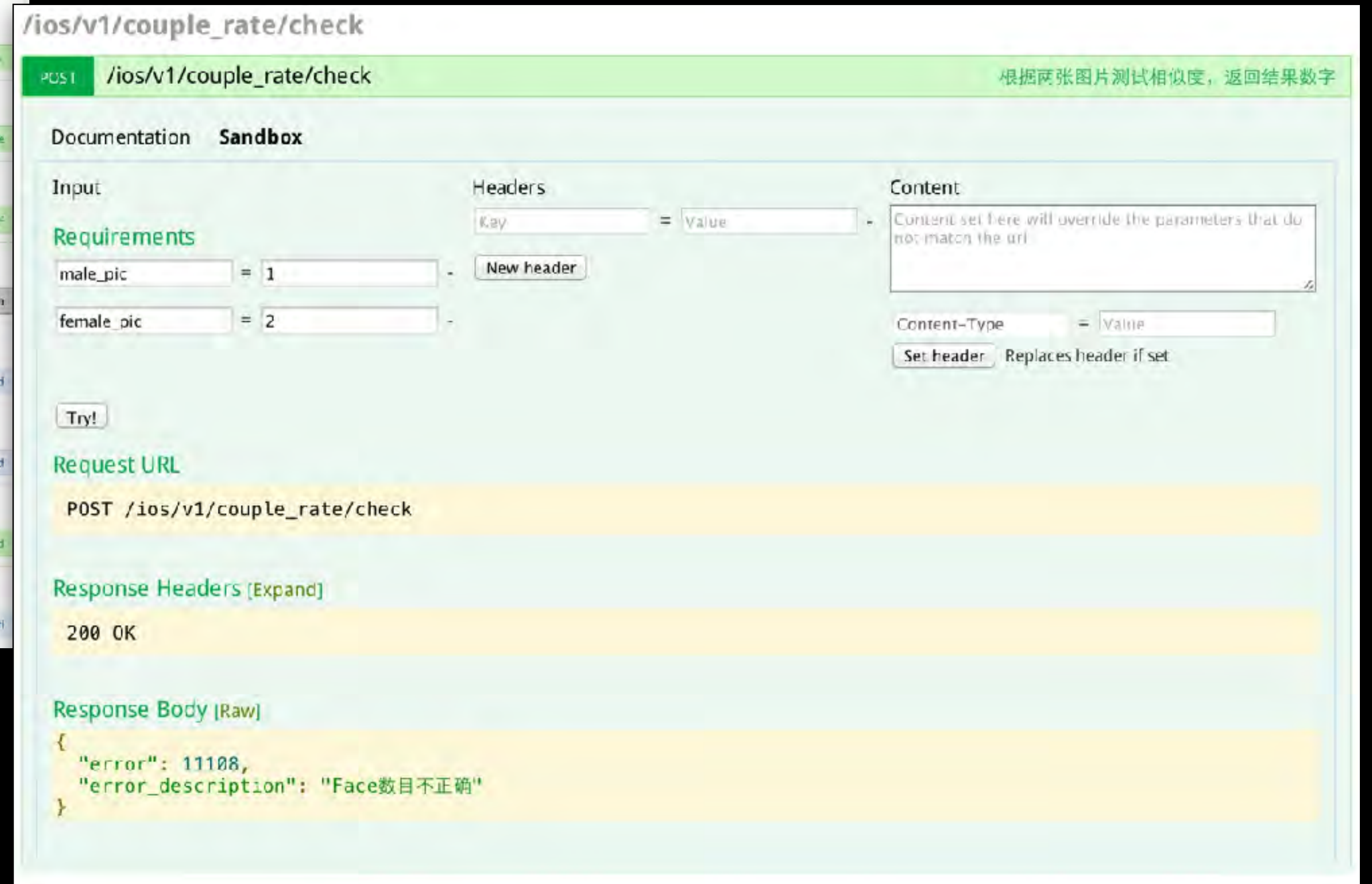
```
/**
 * @Route("/get_city_by_location", name="get_city_by_location")
 * @Method({"GET"})
 * @ApiDoc(
 *     section="Area",
 *     description="根据坐标获取城市ID",
 *     parameters={
 *         {"name"="location", "dataType"="string", "required"=true, "description"="坐标", "format"="39.984154,116.307490"},
 *     },
 *     tags={
 *         "doing"
 *     },
 * )
 */
public function getCityByLocation(Request $request)
{
```



# 自动生成接口文档和调试页面



接口列表



接口详情页（支持在线调试）

IoC / DI / Service



# 举个例子：发送邮件

```
$message = \Swift_Message::newInstance()  
    ->setSubject($subject)  
    ->setFrom($from)  
    ->setTo($to)  
    ->setBody($body)  
;  
  
$this->getContainer()->get('mailer')->send($message);
```

Controller：负责调用发送邮件的Service

```
/** @author Thibault Duplessis <thibault.duplessis@gmail.com> ...*/  
class Mailer implements MailerInterface  
{  
    /** @var \Swift_Mailer ...*/  
    protected $mailer;  
  
    /** @var UrlGeneratorInterface ...*/  
    protected $router;  
  
    /** @var EngineInterface ...*/  
    protected $templating;  
  
    /** @var array ...*/  
    protected $parameters;  
  
    /** Mailer constructor. ...*/  
    public function __construct($mailer, UrlGeneratorInterface $router, EngineInterface $templating)  
    {  
        $this->mailer = $mailer;  
        $this->router = $router;  
        $this->templating = $templating;  
        $this->parameters = array();  
    }  
  
    /** {@inheritdoc} ...*/  
    public function sendConfirmationEmailMessage(UserInterface $user){...}  
  
    /** {@inheritdoc} ...*/  
    public function sendResettingEmailMessage(UserInterface $user){...}  
  
    /** @param string $renderedTemplate ...*/  
    protected function sendEmailMessage($renderedTemplate, $fromEmail, $toEmail){...}
```

Mail Service：负责实现具体如何发送邮件

# 使用Service的优势

- 在整个程序执行上下文环境中，不管调用Service多少次，实例化的对象都可以是同一个，Symfony的Service管理器会为你管理所有Service对象
- Service具有Lazy-loaded特性，可以在你调用的时候再进行初始化
- 在程序的任何角落都可以调用Service（模板，Action，其他的Service，各种监听事件.....）
- Service是优秀和先进的软件架构，帮助你管理复杂的业务逻辑和系统功能逻辑。可以让你以最小的代价响应业务逻辑的迅速变化



性能

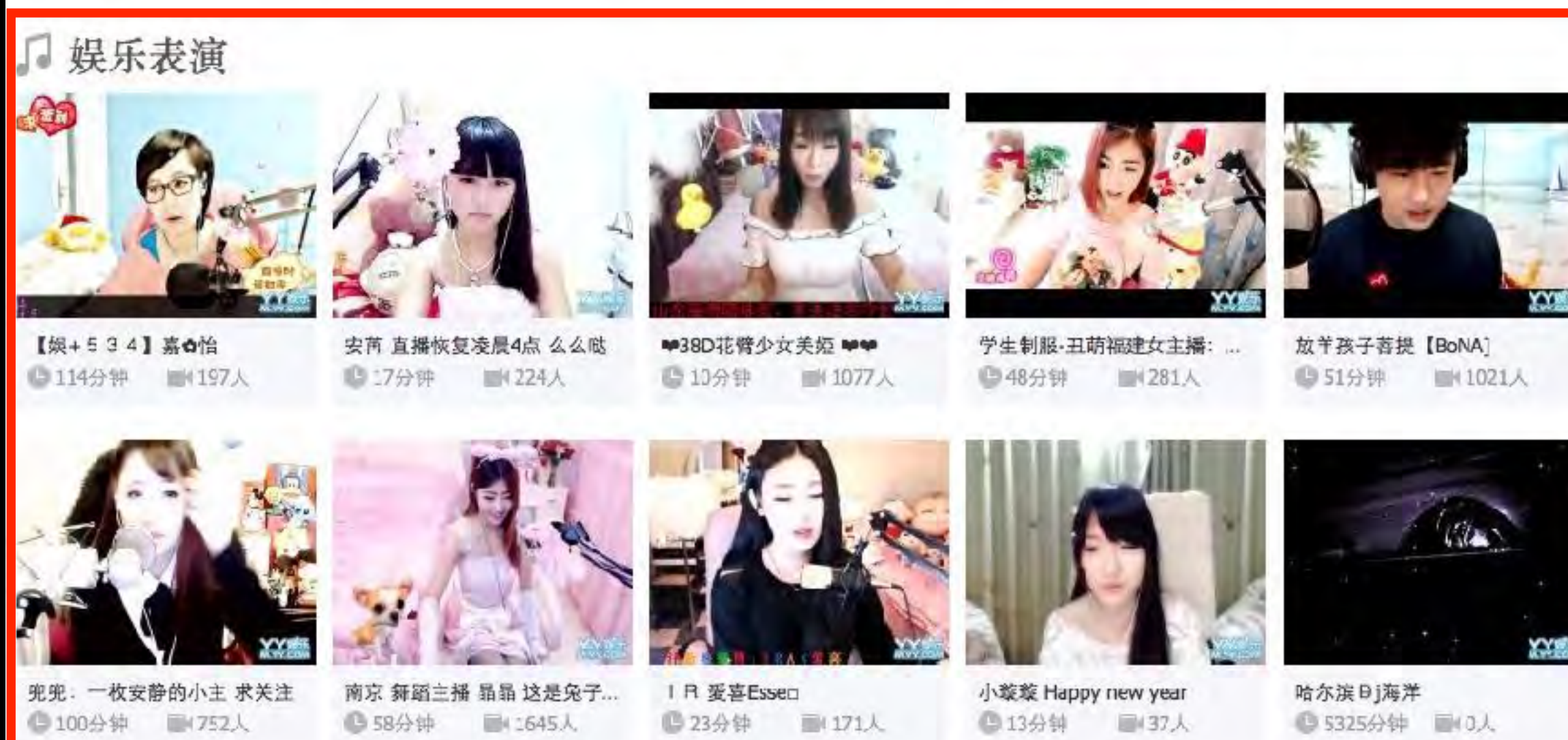
# 这么多『花里胡哨』的功能， Symfony性能会很差吗？

- 100%兼容HHVM
- 支持多环境（dev/test/staging/prod/.....），每个环境都可以自行配置
- Lazy-loaded：使你的程序只加载它用得到的代码
- 强大的代码预处理功能：Annotation/Twig模板/Service等

# Symfony Proxy with Varnish

## — RPS性能提升百倍的利器







# Browser

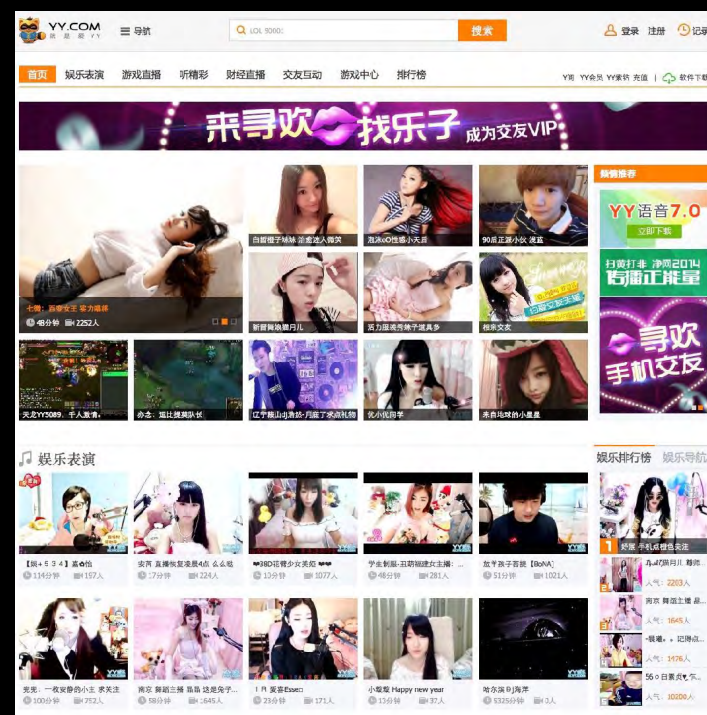
GET /

# Symfony Proxy

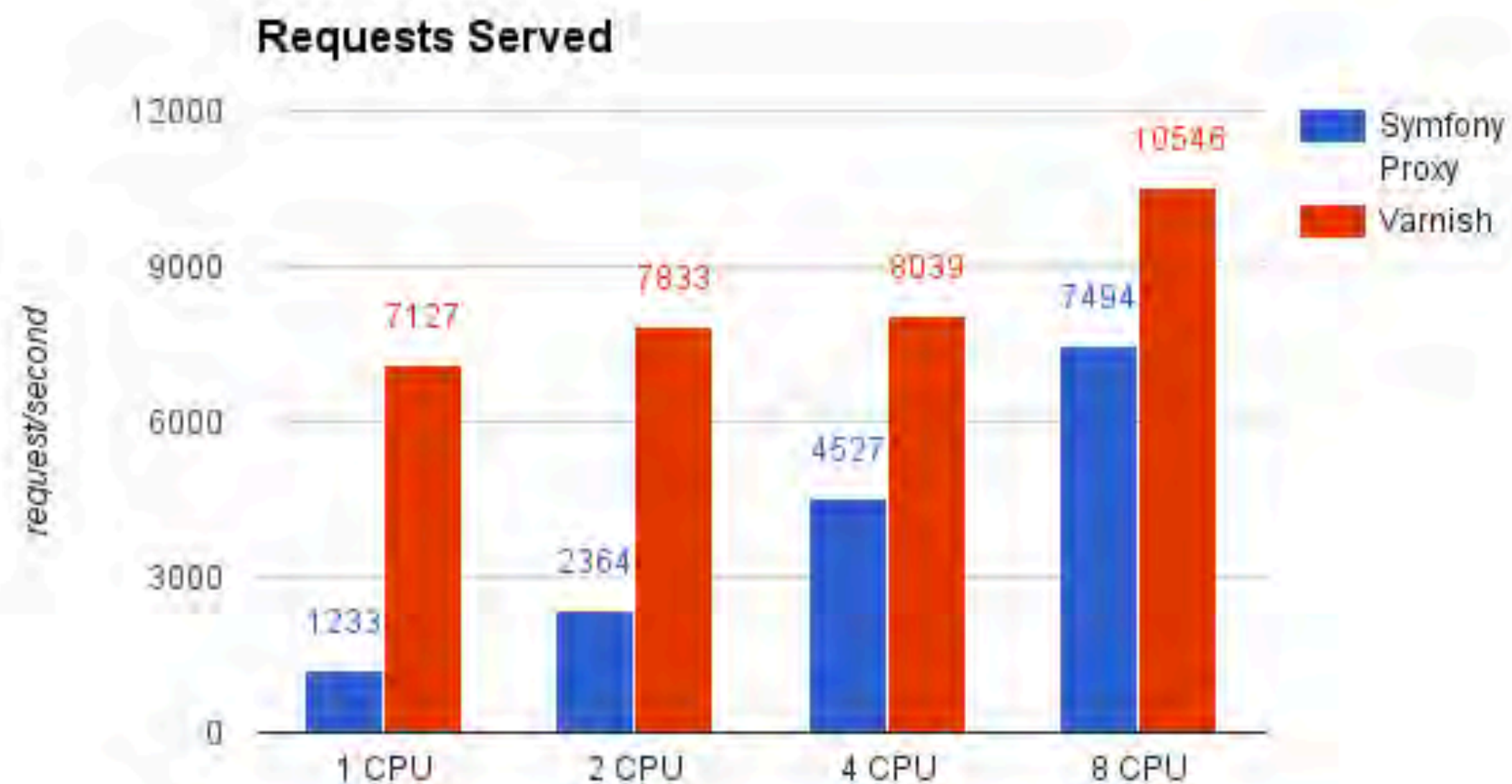
GET /esi/xxxx

# Web Server

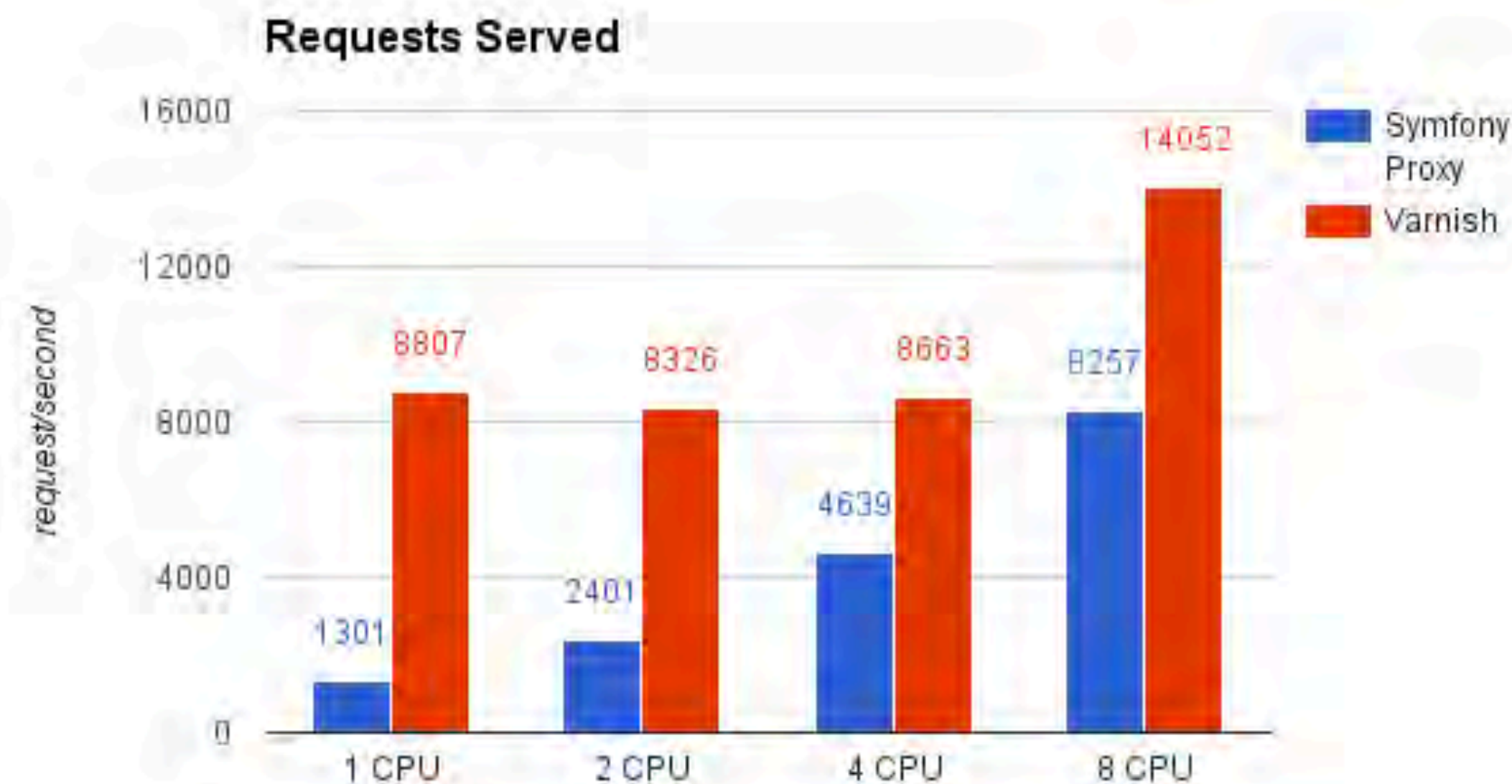
```
<html>
<esi:include src= ""/>
<esi:include src= ""/>
</html>
```



# Symfony Proxy vs Varnish



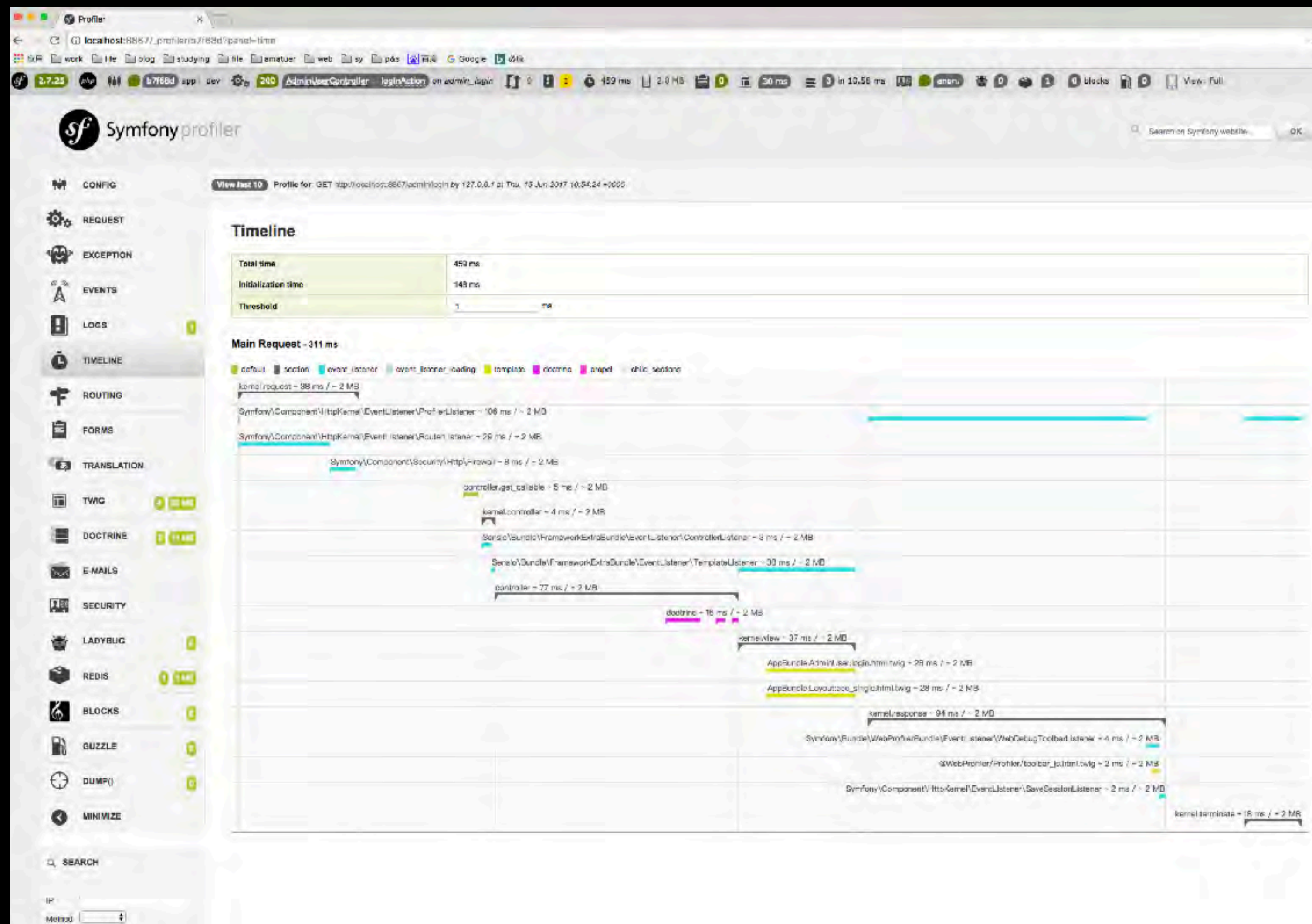
某电商项目Html页



某电商项目API接口



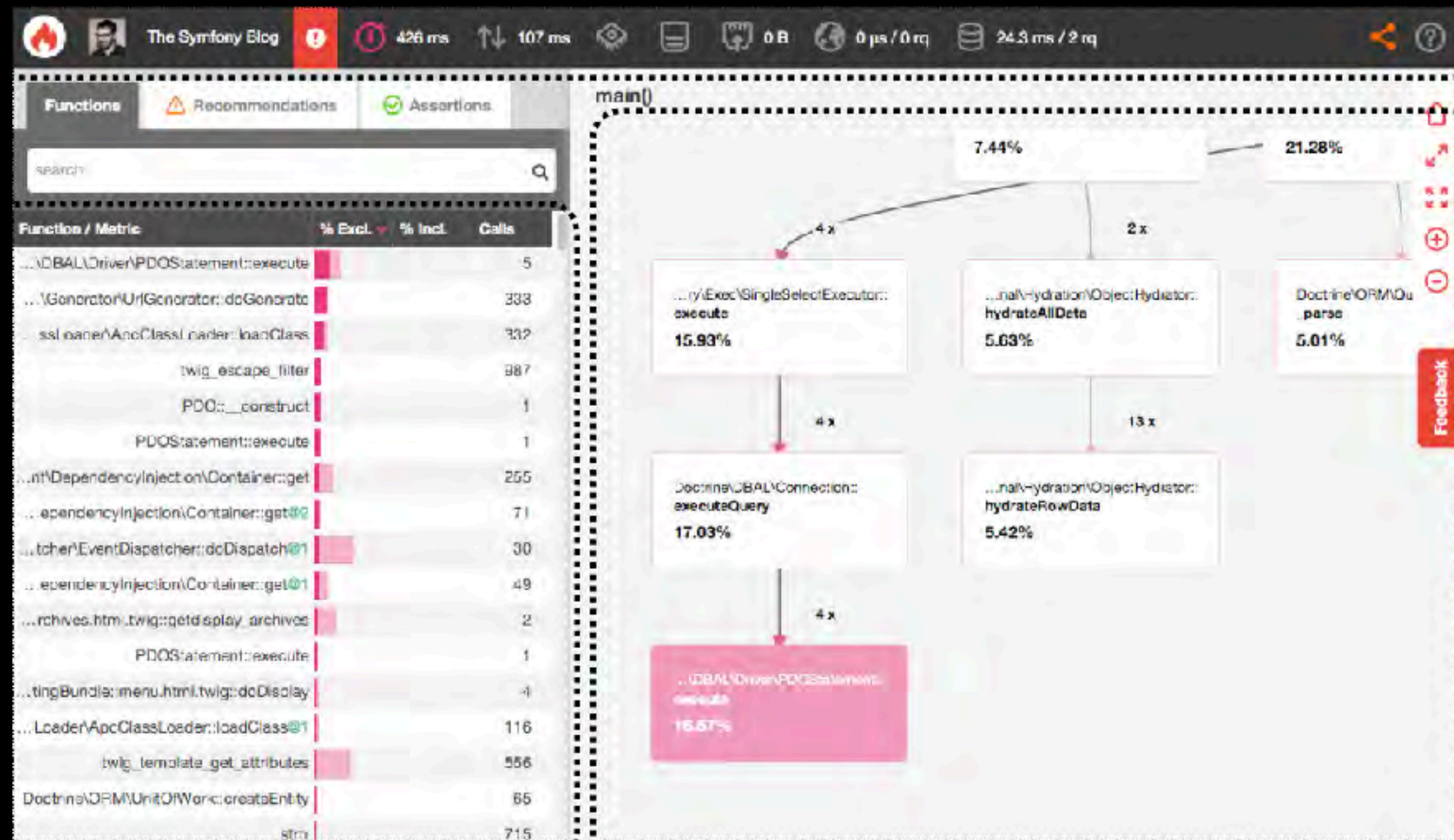
# Symfony Debug Bar — Timeline



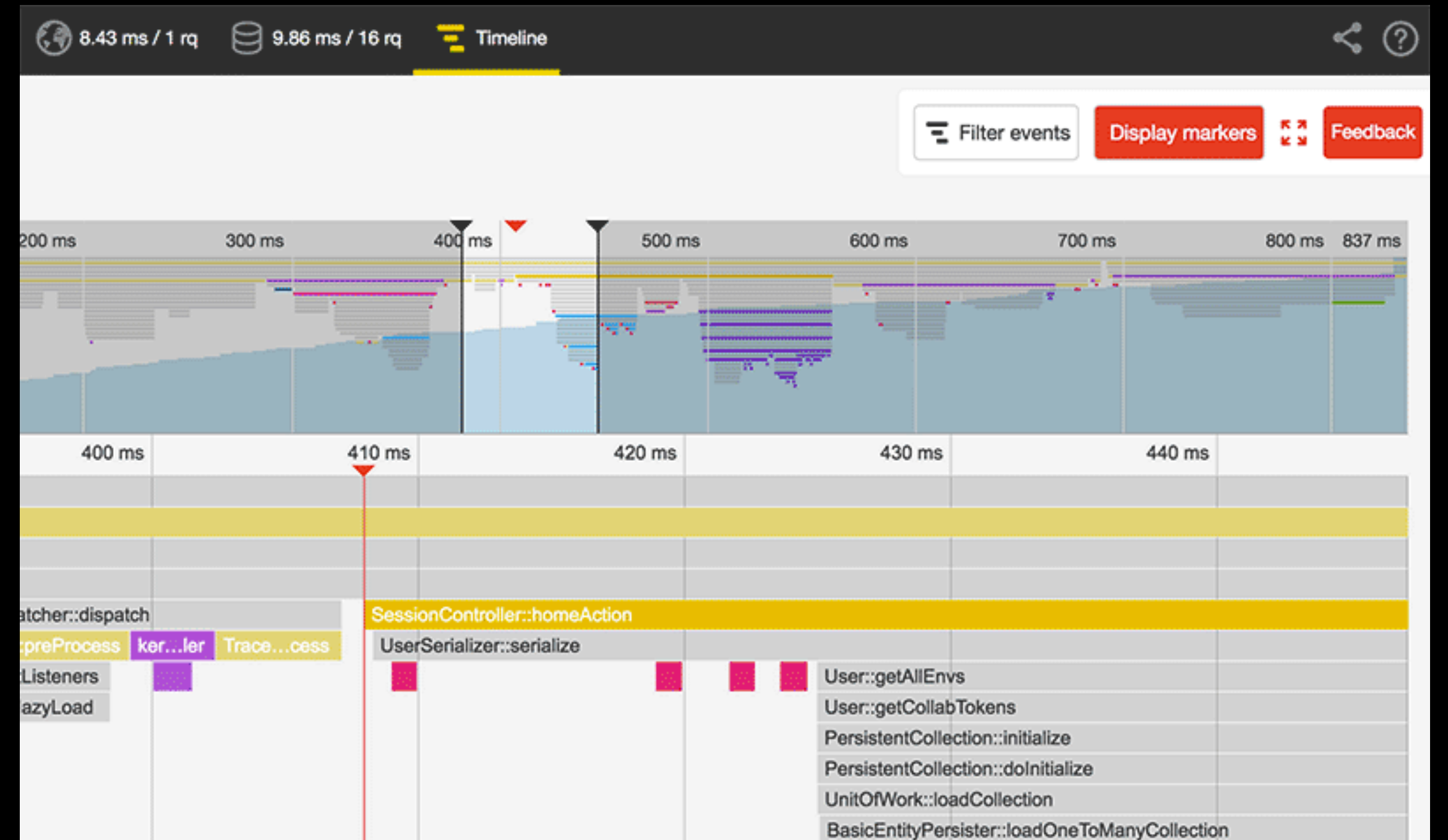


# Blackfire

## — 专业的在线性能分析工具



程序执行树状图



程序执行Timeline



# 3. 如何开始使用 Symfony

# Symfony有好多版本，该怎么选？

- 2.6/2.7/2.8/3.0/3.1/3.2/3.3/4.0
- 为什么有那么多版本？
- 追求稳定性：2.8 LTS
- 希望使用新功能：3.2

# 现在做项目用什么版本号?

- 我推荐使用：2.8 LTS，因为：
  - 2.8 稳定版本，已经上线2年，社区支持足够广
  - 兼容第三方Bundle比较多
- 明年年中左右开始逐步迁移到3.4



Symfony 3.0 was boring, a cleaned-up version of the Symfony 2.8 version:

| *Symfony 3.0 = Symfony 2.8—deprecated features*

Symfony 4.0 will be different:

| *Symfony 4.0 = Symfony 3.4—deprecated features + a new way to develop applications*

There is another way to think about a new major version though:

| *Symfony 4.0 = Symfony 3.0 + all features added in 3.x—deprecated features + a new way to develop applications*

# 一些学习资料

- 官网： <http://symfony.com/>  
官方Github： <https://github.com/symfony>  
Bundles仓库： <http://knpbundles.com/>
- 慕课网视频教程《洪大师带你解读Symfony2框架》：  
<http://www.imooc.com/learn/244>  
国外视频教程： <http://knpuniversity.com/>
- 国内QQ交流群： 230078413

# Symfony内的大量组件 是可以分拆使用的

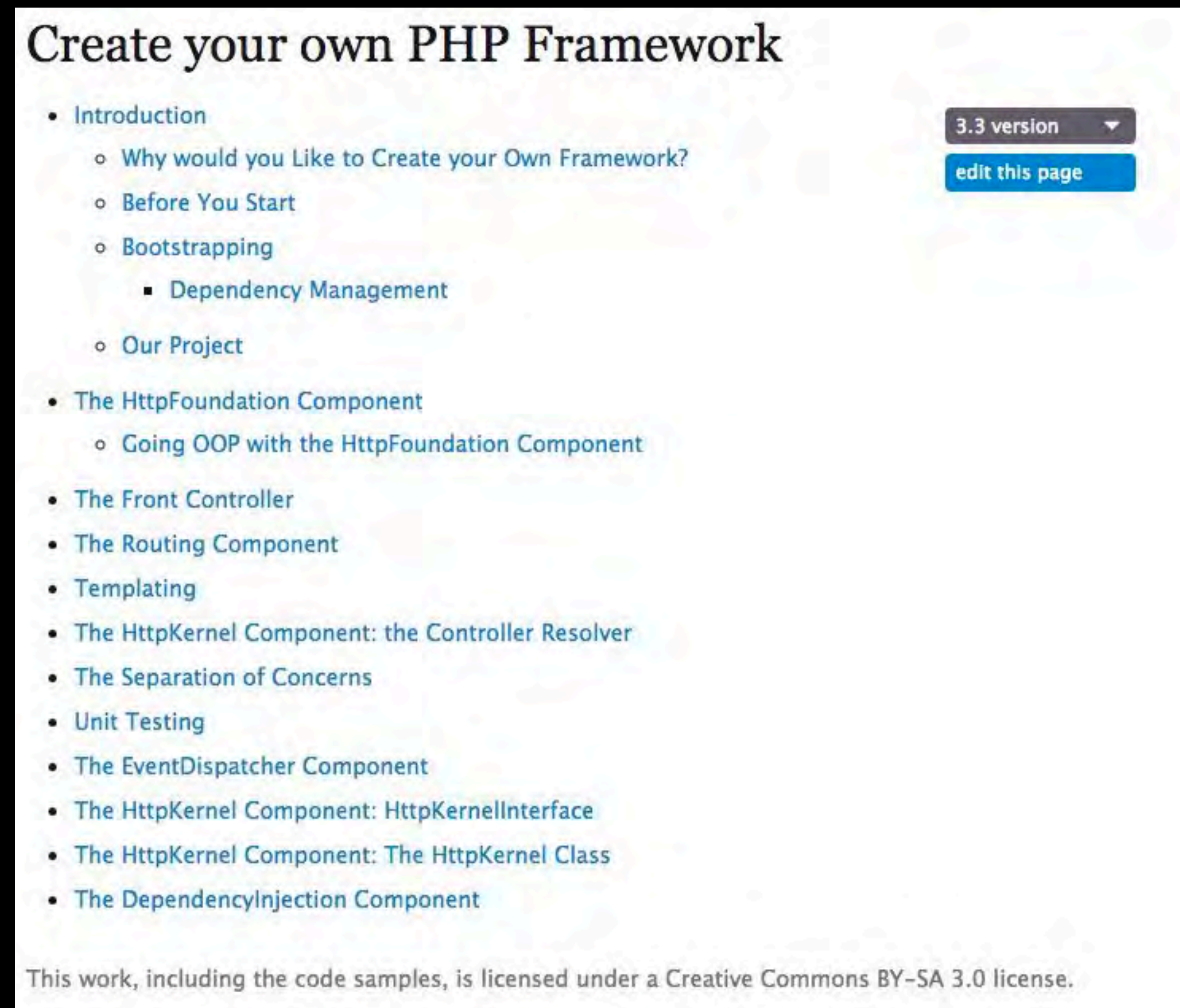
- 基于Symfony核心组件的框架—Silex/Laravel
- 数据库ORM—Doctrine
- HTTP请求处理核心类—HttpKernel
- 模板引擎—Twig
- 路由引擎—Symfony\Component\Routing
- Service引擎
- 等等.....

```
"require": {  
    "php": ">=5.6.4",  
    "ext-mbstring": "*",  
    "ext-openssl": "*",  
    "doctrine/inflector": "~1.0",  
    "erusev/parsedown": "~1.6",  
    "league/flysystem": "~1.0",  
    "monolog/monolog": "~1.11",  
    "mtdowling/cron-expression": "~1.0",  
    "nesbot/carbon": "~1.20",  
    "paragonie/random_compat": "~1.4|~2.0",  
    "ramsey/uuid": "~3.0",  
    "swiftmailer/swiftmailer": "~5.4",  
    "symfony/console": "~3.2",  
    "symfony/debug": "~3.2",  
    "symfony/finder": "~3.2",  
    "symfony/http-foundation": "~3.2",  
    "symfony/http-kernel": "~3.2",  
    "symfony/process": "~3.2",  
    "symfony/routing": "~3.2",  
    "symfony/var-dumper": "~3.2",  
    "tijsverkoyen/css-to-inline-styles": "~2.2",  
    "vlucas/phpdotenv": "~2.2"  
},
```

laravel/composer.json



# Symfony甚至允许并鼓励你去基于它的组件去创造自己的框架



The screenshot shows a web page titled "Create your own PHP Framework". It features a table of contents with a list of links. A dropdown menu in the top right corner shows "3.3 version" and an "edit this page" button. At the bottom, a license notice states: "This work, including the code samples, is licensed under a Creative Commons BY-SA 3.0 license."

## Create your own PHP Framework

- Introduction
  - Why would you Like to Create your Own Framework?
  - Before You Start
  - Bootstrapping
    - Dependency Management
  - Our Project
- The HttpFoundation Component
  - Going OOP with the HttpFoundation Component
- The Front Controller
- The Routing Component
- Templating
- The HttpKernel Component: the Controller Resolver
- The Separation of Concerns
- Unit Testing
- The EventDispatcher Component
- The HttpKernel Component: HttpKernelInterface
- The HttpKernel Component: The HttpKernel Class
- The DependencyInjection Component

This work, including the code samples, is licensed under a Creative Commons BY-SA 3.0 license.

[http://symfony.com/doc/current/create\\_framework/index.html](http://symfony.com/doc/current/create_framework/index.html)

## 4. 我的一些感悟和理解

# 不同阶段的开发侧重点

- 初期
  - 关注开发速度，快速开发出『够用』的产品原型
- 中期
  - 快速响应业务变化
- 后期
  - 性能优化，『尽量』少改代码



“Distributions is the wrong abstraction. We don't need a fully bootstrapped project. We need a way to grow an application over time.”

现在许多『框架』的发展方向都错了。与其让开发者接受一个大而全的解决方案，不如为其提供一种能够稳步发展的开发方式。

# 很多网友经常会问我的问题

- Symfony太慢了
- 真像你说的那么好，怎么国内没人用？
- 我英文不好啊，Symfony的中文资料真的很少啊，我看的很累

谢谢