

DESENVOLVIMENTO DE SISTEMAS WEB COM PHP FRAMEWORKS (SYMPHONY)

Martin Pfeifer
martin.pfeifer20@gmail.com

Sobre mim

- *Especialista em Engenharia e qualidade de software*
- *Arquiteto de software e sub-gerente na PublicSoft*
- *Programador sênior na LeaderWeb*



Introdução

- Novidades e benefícios do php 7+
- Boas práticas de programação com object calisthenics php
- DataTransferObject (DTO) no php
- Introdução a DDD (Domain Driven Design)
- Serialização de objetos com JMS Serializer
- Gerenciamento de dependência com Composer
- Introdução ao Symfony
- Comparação de performance com outros frameworks;
- Versões do Symfony 3+
- Symfony e suas funcionalidades
- Criar um projeto para gerenciar currículos
- Tópicos especiais

Novidades e benefícios PHP 7+

- Erros fatais e exceções
- Indução de Tipos: Scalar Types
- Tipo de Retorno de Funções e Métodos
- Novo Operador Spaceship (<=>)
- Null Coalesce Operator (operador ??)
- Nullable Types 7.1
- Capturando múltiplas exceções
- Void return
- Pseudo-tipo iterable

Erros fatais e exceções

```
ereg('^[a-z]$', 'php7');  
echo "FIM";
```

Fatal error: Uncaught Error: Call to undefined function ereg()...

Já no php 7+

```
try {  
    ereg('^[a-z]$', 'php7');  
} catch (Error $e) {  
    echo "Ocorreu um erro: " . $e->getMessage(); }  
echo "FIM";
```

Ocorreu um erro: Call to undefined function ereg()
FIM

Indução de tipos: Scalar types

- PHP é uma linguagem NÃO tipada.
- Aos poucos ela vem ganhando alguns recursos que a torna fracamente tipada.

- **Tipo de Retorno de Funções e Métodos**

```
function soma(int $x, int $y) : float
{
    return $x + $y + 1.5;
}
```

```
function funcaoX(string $nome) : string
{
    return "Aluno"
}
```

Novo operador Spaceship (<=>)

- Esse operador recebe o nome de “Spaceship” em algumas outras linguagens, e é usado para comparação numérica.
- Se você já usou a função strcmp(), com certeza vai entender esse operador sem dificuldades.

- **Exemplos**

```
var_dump(2 <=> 3); // retorna -1
```

```
var_dump(2 <=> 2); // retorna 0
```

```
var_dump(2 <=> 1); // retorna 1
```

Null Coalesce Operator (operador ??)

- Ele é útil para verificar a existência de variáveis, como fazemos com valores de `$_GET` ou `$_POST`, usando `isset`.

- **Exemplos**

No php 7+:

```
$email = $_POST['email'] ?? 'valor padrão';
```

No php inferior ao 7:

```
$email = isset($_POST['email']) ? $_POST['email'] : 'valor padrão';
```


Nullable types

```
function bar(int $value) {  
    if($value) {  
        // todo  
    }  
}
```

```
bar();
```

Uncaught TypeError: Argument 1 passed to bar() must be of the type integer, none given

```
bar(1); // ok
```

```
bar(0); // ok
```

```
bar(); // Erro! É preciso enviar null ou um valor do tipo que a função espera receber
```

```
bar("1"); // Erro! O tipo enviado não é inteiro.
```

Capturando múltiplas exceções

```
try {  
    // todo  
} catch (MyException1 $e) {  
    // todo  
} catch (MyException2 $e) {  
    // todo  
} catch (Exception $e) {  
    // todo  
}  
  
try {  
    // todo  
} catch (MyException1 | MyException2 $e) {  
    // todo  
} catch (Exception $e) {  
    // todo  
}
```

Void return

```
public function run() : void  
{  
    // todo  
}
```

Pseudo-tipo iterable

```
public function change(iterable $meta) : void
{
    foreach($meta as $key => $value) {
        // todo
    }
}
```

```
public function getMeta() : iterable
{
    return [1, 2, 3, 4, 5];
}
```



Boas práticas de programação - Object calisthenics php

- Um nível de indentação por método;
- Não use ELSE;

Um nível de indentação por método

- class Customer

```
{  
    public function getPromoCode(string $promoName)  
    {  
        // 1.  
        if ($this->promoCode) {  
            // 2.  
            if (false === $this->promoCodeExpired()) {  
                // 3.  
                if ($this->promoName == $promoname) {  
                    return $this->promoCode;  
                } else {  
                    throw new Exception('Promoção não existe mais');  
                }  
            } else {  
                throw new Exception('Promoção Expirada');  
            }  
        } else {  
            throw new Exception('Cliente sem código de promoção');  
        }  
    }  
}
```

Um nível de indentação por método

- class Customer

```
{
    public function getPromoCode(string $promoName)
    {
        if ($this->promoCode) {
            return $this->getValidPromoCode($promoName);
        }
        throw new Exception('Cliente sem código de promoção');
    }

    public function getValidPromoCode(string $promoName)
    {
        if (false === $this->promoCodeExpired()) {
            return $this->getPromoExists($promoName);
        }
        throw new Exception('Promoção Expirada');
    }

    public function getPromoExists(string $promoName)
    {
        if ($this->promoName == $promoName) {
            return $this->promoCode;
        }
        throw new Exception('Promoção não existe mais');
    }
}
```

Não use ELSE

- Essa regra parece ser muito estranha, mas ela funciona muito bem com o conceito early return, que emprega o uso do "retorne seu valor o quanto antes", ação que só é facilmente implementada dentro de funções, métodos ou loops.

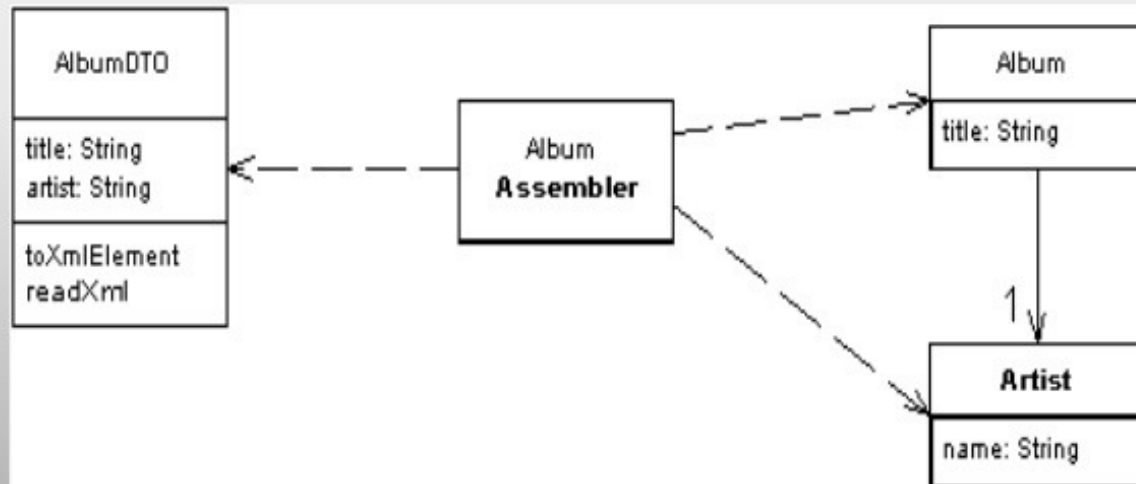
```
// Exibir a lista de membros
// que pagaram a mensalidade
foreach ($members as $member) {
    if ($member->paid()) {
        $report[] = [$member->name => 'Paid'];
    } else {
        $report[] = [$member->name => 'Not Paid'];
    }
}
```

```
// Sem Else
foreach ($members as $member) {
    if ($member->paid()) {
        $report[] = [$member->name => 'Paid'];
        continue;
    }
    $report[] = [$member->name => 'Not Paid'];
}
```


Data Transfer Object (DTO)

Data Transfer Object

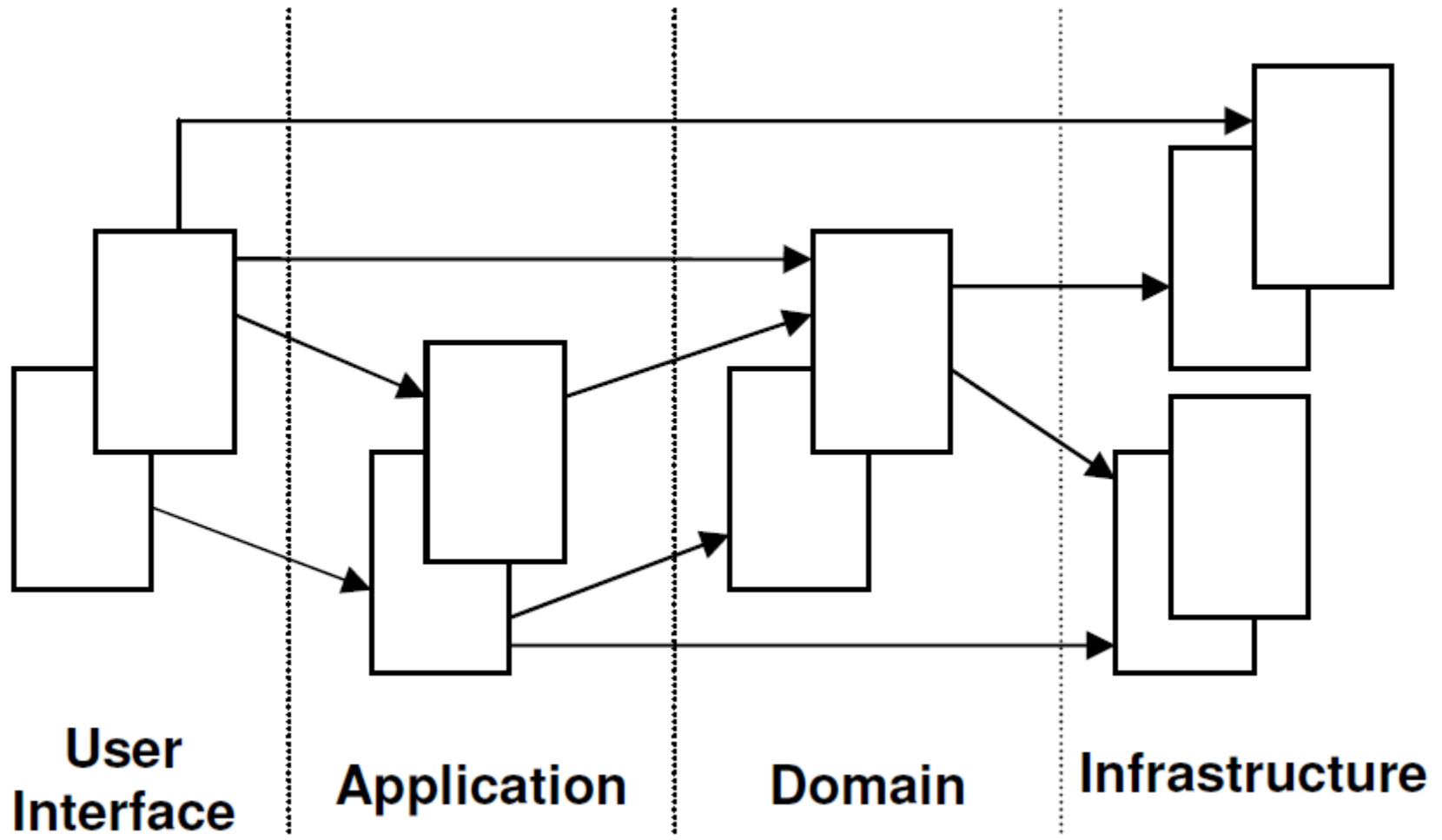
- *An object that carries data between processes in order to reduce the number of method calls*



```
function findAlbum(AlbumDTO $albumDTO) {  
    //todo  
}
```

Introdução a DDD (Domain Driven Design)

- Alinhamento do código com o negócio
- Favorecer reutilização
- Mínimo de acoplamento
- Independência da Tecnologia
- Linguagem Ubíqua



Serialização de objetos com JMS Serializer

- Suporta XML e JSON.
- Recursos internos incluem:
- (De-) serializar dados de qualquer complexidade; Referências circulares são manipuladas graciosamente.
- Suporta muitos tipos de PHP integrados (como datas)
- Integra-se com o Doctrine ORM, et. al.
- Suporta versionamento, por exemplo, para APIs
- Configurável via Anotações XML, YAML ou Doctrine

```
$serializer = JMS\Serializer\SerializerBuilder::create()->build();  
$jsonContent = $serializer->serialize($data, 'json');  
echo $jsonContent; // or return it in a Response
```

Gerenciamento de dependência com Composer

Arquivo composer.json

```
{
  "name": "Nome do projeto",
  "description": "Breve descrição do que a aplicação se propoe a fazer",
  "authors": [
    {
      "name": "Seu nome",
      "email": "seu-email@seu-dominio.com"
    }
  ],
  "require": {
    "php": ">=5.2.8"
  }
}
```

O “name” é o nome de sua aplicação. Esta marcação é opcional mas recomendada.

O “description” é uma breve descrição do que sua aplicação se propõe a fazer. Também opcional.

Em “authors” aparecem os créditos de desenvolvedores que contribuíram com o projeto.

Introdução do Symfony

- História
- Flexibilidade ilimitada
- Expansível
- Estável e sustentável
- Inversão de Controle (Ioc) vs Injeção de Dependência
- Comparação de performance com outros frameworks
- Estrutura do symfony
- Bundles
- Serviços
- Repositórios
- Doctrine ORM
- Command

História

- Por trás do Symfony existe uma empresa: SensioLabs. Criada há mais de 18 anos, a SensioLabs é uma agência da web que possui muitas das principais referências entre suas referências. Concebido para suas próprias necessidades, o framework Symfony é ainda hoje a ferramenta diária usada por suas próprias equipes para desenvolver projetos de clientes. Projetado por profissionais para profissionais, o Symfony é, antes de tudo, uma ferramenta pragmática, cujas características atendem aos requisitos do mundo real.

Flexibilidade Ilimitada

- Quaisquer que sejam suas necessidades, o Symfony será adaptável. Seu injetor de dependência e o Event Dispatcher o tornam totalmente configurável, com cada um dos tijolos sendo totalmente independentes. Um quadro 3 em 1, de tipos:
 1. Full Stack (versão completa): você quer desenvolver um aplicativo complexo e precisa de muitas funcionalidades.
 2. Tijolo por tijolo: você constrói sua estrutura de acordo com as funcionalidades que você precisará.
 3. Microframework: como um standalone, o Symfony também pode ser usado para desenvolver uma funcionalidade específica em um de seus projetos. Sem ter que redesenvolver tudo e sem instalar todo o framework, mas apenas o brick específico que você precisa.

Expansível

- Do menor bloco ao próprio núcleo completo, tudo é apresentado como um “pacote” (ou plug-in na linguagem do Symfony) no Symfony. Cada pacote tem a finalidade de adicionar funcionalidade à estrutura, é claro, e cada pacote também pode ser reutilizado em outro projeto ou compartilhado com o restante da comunidade.

Em qualquer caso, o sistema de pacotes permite que tudo mude dentro do Symfony, incluindo o próprio núcleo. Usando os contratos de interface do sistema entre tijolos, o comportamento da estrutura pode ser alterado à vontade, sem exigir uma reconfiguração completa.

Estável e Sustentável

- Desenvolvido pela SensioLabs, as principais versões do Symfony são suportadas por 3 anos pela empresa. E até mesmo para a vida, no que diz respeito a questões relacionadas à segurança.

Para uma estabilidade ainda maior, as versões secundárias do contrato e da interface do Symfony também são garantidas e a compatibilidade entre todas as versões secundárias será assegurada na API definida pelas interfaces públicas.

Escolher o Symfony significa ter uma visão de longo prazo dos ativos de aplicativos. Isso também significa saber como facilitar muito o dia-a-dia dos desenvolvedores.

Inversão de Controle (Ioc) x Injeção de Dependência

IOC:

```
public class VendaDeProduto {  
    public void vendeProduto(Produto produto) {  
        //Todo o código para a venda do produto...  
        Log log = new Log("Arquivo.txt");  
        log.grava(produto);  
    }  
}
```

Pra você que desenvolve com TDD, como faríamos por exemplo para fazer um teste com a classe VendaDeProduto quando precisarmos "mocká-la"? Simplesmente não conseguimos, justamente por ela criar o objeto Log! E não vale você que conhece o PowerMock inventar algo! :)

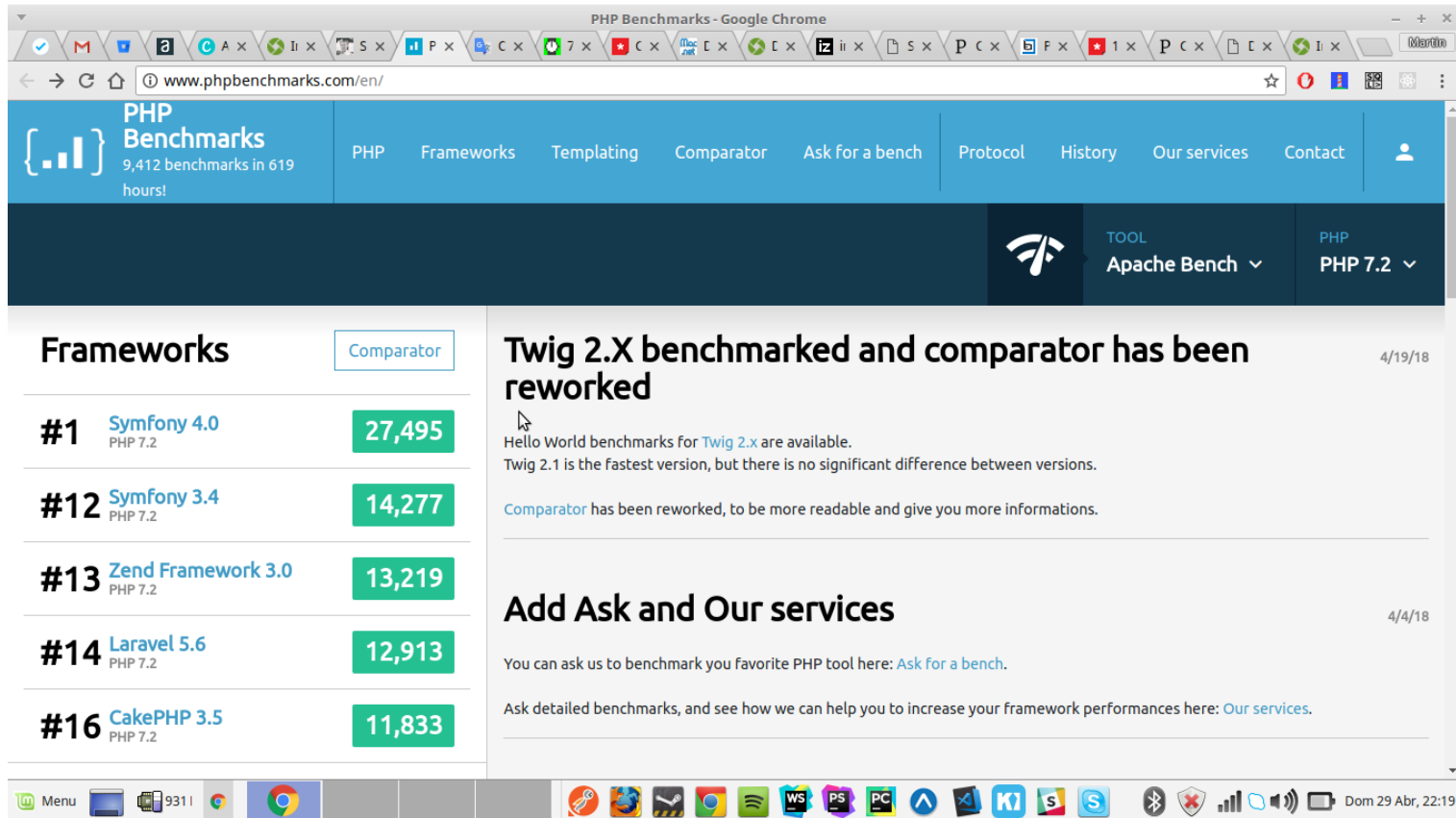
Resolução com Injeção de dependência

```
public class VendaDeProduto {  
    private Log log;  
    public void VendaDeProduto(Log logVenda) {  
        this.log = logVenda;  
    }  
    public void vendeProduto(Produto produto) {  
        //Todo o código para a venda do produto...  
        log.grava(produto);  
    }  
}
```

Comparação de performance com outros frameworks

<http://fabien.potencier.org/symfony4-performance.html>

<http://www.phpbenchmarks.com/en/>



The screenshot shows the PHP Benchmarks website in a Google Chrome browser. The page has a blue header with the site logo and navigation links. Below the header, there's a dark blue bar with a search icon and dropdown menus for 'TOOL' (set to 'Apache Bench') and 'PHP' (set to 'PHP 7.2'). The main content area is divided into two columns. The left column, titled 'Frameworks', contains a table of benchmark results. The right column features a news item about Twig 2.X benchmarks and a section for asking questions or services.

Frameworks		
#1	Symfony 4.0 PHP 7.2	27,495
#12	Symfony 3.4 PHP 7.2	14,277
#13	Zend Framework 3.0 PHP 7.2	13,219
#14	Laravel 5.6 PHP 7.2	12,913
#16	CakePHP 3.5 PHP 7.2	11,833

Twig 2.X benchmarked and comparator has been reworked 4/19/18

Hello World benchmarks for [Twig 2.x](#) are available.
Twig 2.1 is the fastest version, but there is no significant difference between versions.

[Comparator](#) has been reworked, to be more readable and give you more informations.

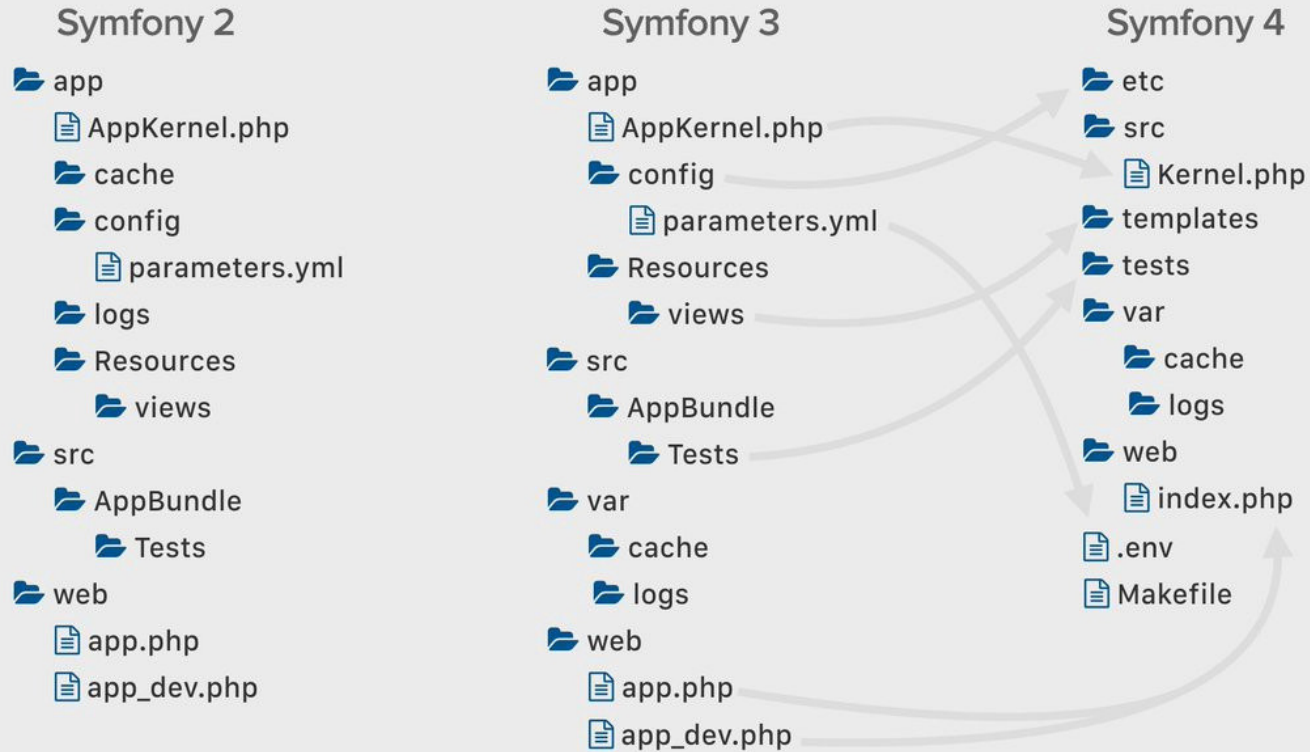
Add Ask and Our services 4/4/18

You can ask us to benchmark you favorite PHP tool here: [Ask for a bench](#).

Ask detailed benchmarks, and see how we can help you to increase your framework performances here: [Our services](#).

Estrutura da aplicação

The default directory structure of Symfony applications

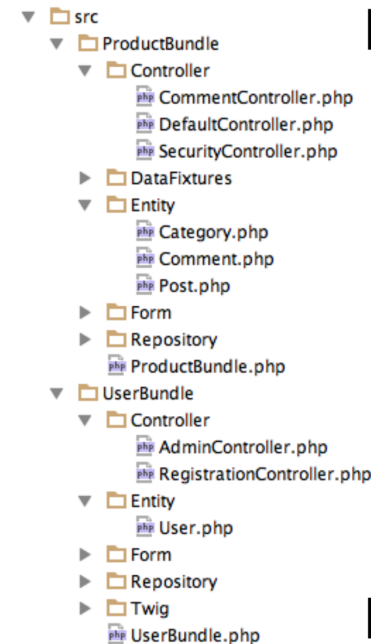


Bundles

```
// app/AppKernel.php
public function registerBundles ()
{
    $bundles = array (
        new Symfony\Bundle\FrameworkBundle\FrameworkBundle (),
        new Symfony\Bundle\SecurityBundle\SecurityBundle (),
        new Symfony\Bundle\TwigBundle\TwigBundle (),
        new Symfony\Bundle\MonologBundle\MonologBundle (),
        new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle (),
        new Symfony\Bundle\DoctrineBundle\DoctrineBundle (),
        new Symfony\Bundle\AsseticBundle\AsseticBundle (),
        new
Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundl
        e (),
        new AppBundle\AppBundle (),
    );

    if ( in_array ( $this -> getEnvironment (), array ( 'dev' , 'test' ))) {
        $bundles [] = new
        Symfony\Bundle\WebProfilerBundle\WebProfilerBundle ();
        $bundles [] = new
Sensio\Bundle\DistributionBundle\SensioDistributionBundle ();
        $bundles [] = new
Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle ();
    }

    return $bundles ;
}
```



Coupled

Serviços

arquivo services.yml

services:

App\Foo\Bar:

arguments: ['@baz', 'foo', '%qux%']