12 fatores em projetos PHP

Elton Minetto

http://eltonminetto.net
@eminetto

Criado pela equipe da Heroku é uma espécie de “manifesto” com os 12 fatores que uma aplicação deveria seguir para ser "cloud native".

I.Codebase

Uma única base de código, múltiplos deploys, gerenciado por controle de versões

→ Github

→ Gitlab

→ Bitbucket

II.Dependencies

Todas as dependências do projeto devem estar declaradas no próprio projeto.

→ Composer

→ Infrastructure as Code (Docker, Puppet, Chef, Ansible, Terraform, etc)

III.Config

Configurações armazenadas fora do código.

Variáveis de ambiente

```php
<?php
public static function build()
{
    $bernardSerializer = new BernardSerializer(new FQCNMessageFactory(), new NoOpMessageConverter());
    switch (getenv('QUEUE_DRIVER')) {
        case 'sqs':
            $connection = SqsClient::factory([
                'key'    => getenv('QUEUE_SQS_KEY'),
                'secret' => getenv('QUEUE_SQS_SECRET'),
                'region' => getenv('QUEUE_SQS_REGION')
            ]);

            $driver = new SqsDriver($connection);
            break;
        case 'flatfile':
            $driver = new FlatFileDriver(getenv('QUEUE_FLATFILE_PATH'));
            break;
    }
    $queue = new PersistentFactory($driver, $bernardSerializer);

    return $queue;
}
```

Arquivos distintos para ambientes distintos

```
ls -l config/
-rw-r--r--   1 eminetto  staff   646 Nov 27 11:51 local.php
-rw-r--r--   1 eminetto  staff  1237 Oct  2 09:06 staging.php
-rw-r--r--   1 eminetto  staff   357 Jul 31 11:21 testing.php
-rw-r--r--   1 eminetto  staff   549 Oct  2 09:06 production.php
```

IV.Backing Services

Serviços externos que o aplicativo consome.
Banco de dados, cache, podem estar tanto locais quanto remotos sem mudanças de código

→ Doctrine

→ Bernard

→ Zend Cache, Doctrine Cache, etc

V.Build, release, run

Três fases bem separadas e definidas facilita a criação de scripts e procedimentos a serem executados em cada uma delas.

→ Makefiles

→ Shell script

→ Deployer

VI.Processes

O aplicativo como um ou mais processos, que sejam “stateless” e “share-nothing”
Diminuir o acoplamento entre componentes do projeto para facilitar a escala

→ JWT

→ Filas, cache, S3, etc.

VII.Port binding

Não depender de um servidor externo para ser executado, poder ser auto-contido e executar em uma porta específica que seria acessado por outras partes do projeto.
Depender de uma estrutura de nomes e endereços que podem ser configurados em arquivos de configuração

```php
<?php

namespace Application\Service;

use Psr\Container\ContainerInterface;

class InstallFactory
{
    public function __invoke(ContainerInterface $container)
    {
        $entityManager = $container->get('EntityManager');
        $cache = $container->get('Cache');
        $config = $container->get('config');
        $tokenService = $container->get('TokenService');
        $providerService = $container->get('ProviderService');

        return new Install($entityManager, $cache, $config, $tokenService, $providerService);
    }
}
```

VIII. Concurrency

Pensar o projeto como processos que podem ser executados em paralelo.

→ RabbitMQ, Gearman, SQS (Bernard ajuda a deixar transparente)

IX. Disposability

Processos facilmente descartáveis, que podem ser iniciados ou parados a qualquer momento.
Facilitar este processo, permitindo início rápido, processo de finalização simplificado

X. Dev/prod parity

Ambientes de desenvolvimento, homologação e produção devem ser iguais

→ Vagrant

→ Docker

→ Puppet, Chef, Ansible, Terraform

XI. Logs

O código não deve se preocupar com o formato de armazenamento. Pode enviar as mensagens para a saída padrão e esta deve ser redirecionada para locais específicos de acordo com o ambiente onde o projeto está executando. Ou usar ferramentas específicas.

→ Monolog

→ Zend Log

→ Sentry

XII. Admin processes

Tarefas administrativas como limpar caches, carregar dados, atualizar bases de dados, devem ser tratadas de forma automatizada

→ Migrations (Laravel, Doctrine, etc)

→ Fixtures (Laravel, Doctrine, etc)

→ Makefiles

→ Shell Script

Resumindo

I.Codebase
II.Dependencies
III.Config
IV.Backing Services
V.Build, release, run
VI.Processes
VII.Port binding
VIII. Concurrency
IX. Disposability
X. Dev/prod parity
XI. Logs
XII. Admin processes

Perguntas

Contato

http://eltonminetto.net

http://coderockr.com

http://codenation.com.br

http://asemanaphp.com.br