# PHP com Silex no Google App Engine

Nanderson Castro

# PHP com Silex no Google App Engine

## Abordagem prática na criação de APIs com o micro framework Silex.

Nanderson Castro

Esse livro está à venda em http://leanpub.com/phpcomsilex-e-googleappengine

Essa versão foi publicada em 2017-09-14


Esse é um livro Leanpub. A Leanpub dá poderes aos autores e editores a partir do processo de Publicação Lean. Publicação Lean é a ação de publicar um ebook em desenvolvimento com ferramentas leves e muitas iterações para conseguir feedbacks dos leitores, pivotar até que você tenha o livro ideal e então conseguir tração.

© 2015 - 2017 Nanderson Castro

# Tweet Sobre Esse Livro!

Por favor ajude Nanderson Castro a divulgar esse livro no Twitter!

A hashtag sugerida para esse livro é #phpcomsilexnoappengine.

Descubra o que as outras pessoas estão falando sobre esse livro clicando nesse link para buscar a hashtag no Twitter:

#phpcomsilexnoappengine

*Agradeço a DEUS pela força concedida para criar esse material, agradeço em especial a minha esposa por me aguentar em minha carreira e em minhas saídas para eventos. Agradeço também aos meus amigos da comunidade PHP Maranhão, comunidade essa que participo ativamente e cresço bastante a cada dia através das interações existentes na mesma. Agradeço minha familia por sempre acreditar e incentivar o meu trabalho. Por fim, agradeço a minha bebê, é por ela também que tenho forças para continuar gerando conteúdos e divulgando conhecimento. Enfim, obrigado a você leitor, por ter adquirido este e-book. Espero contribuir com sua carreira profissional e com seu crescimento!*

# Conteúdo

# Introdução

Olá, seja bem-vindo ao nosso e-book sobre o micro framework Silex, neste e-book abordaremos todo o poder desse micro framework criado por Fabien Potencier. O Silex é um micro framework baseado nos componentes do Symfony e foi criado para ser focado em aplicações pequenas, extensíveis e facilmente testáveis!

Em nosso e-book abordaremos a utilização desse fw com prática do começo ao fim e vamos criar uma API de eventos como estudo de caso. Através dessa API pretendemos mostrar tudo o que o Silex nos disponibiliza para a criação de aplicações utilizando suas ferramentas. Veremos como ele pode ser facilmente estendido e utilizaremos testes em nossa API do começo ao fim, para torna-la mais consistente possível!

Por fim conheceremos o Google App Engine, um dos produtos do Google Cloud Platform[1]. Nesta sessão abordaremos todo o processo de deploy de nossa API e durante esse deploy mostraremos os conceitos necessários sobre o Google Cloud Platform e sobre o Google App Engine.

Embarque conosco nessa jornada! Esperamos que esse conhecimento possa te ajudar profissionalmente e não pouparemos esforços para oferecer o melhor durante sua leitura!!

[1] https://cloud.google.com

# Conhecendo e instalando o Silex

## Silex

O Silex é um micro framework baseado nos componentes do Symfony e foi feito pelo mesmo criador deste framework, Fabien Potencier. O Silex foi concebido para a criação de aplicações pequenas com foco na agilidade, extensibilidade e para ser facilmente testável. O Silex provê um sistema de rotas muito poderoso, inclusive rotas é a área que ele se propõe a resolver porém através dos Services e Providers, conceitos que veremos mais a frente, você perceberá que ele é facilmente extendido e suas funcionalidades recebem o plus através dessas integrações.

## Instalação

Para instalar o Silex em nossos projetos é muito simples, precisamos apenas do composer[2] para gerenciar nossas dependências. Mas afinal o que é o composer?! O composer é um gerenciador de dependências para aplicações PHP, ele é baseado nas GEMs do Ruby e no NPM do Node.JS. Com o composer você pode facilmente gerenciar a instalação de pacotes de terceiros, bem como preparar o seu pacote para que ele fique disponivel para os desenvolvedores que utilizam essa ferramenta. Tudo que precisaremos é de um arquivo composer.json na raiz de nosso projeto. Utilizaremos `api-events` como nome do nosso folder.

Na raiz desse folder crie um arquivo composer.json com o seguinte conteúdo:

```
1 {
2         "require" : {
3                 "silex/silex" : "^1.2"
4         }
5 }
```

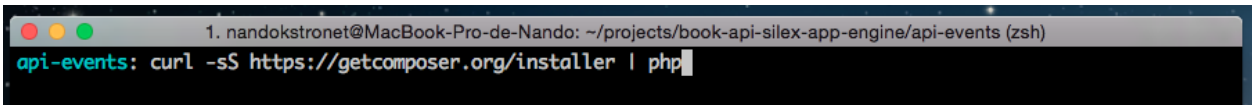
O composer.json é o arquivo que o Composer lê para poder realizar as tarefas de download e instalação dos pacotes, ali especificados, para você.

Agora precisamos instalar o composer em nosso projeto. O composer pode ser utilizado de duas maneiras, de forma local e de forma global, abordarei aqui a forma local. Para instalá-lo em sistemas Unix like, você precisará da lib curl disponível. Se você utiliza o Windows, baixe o executável aqui[3].

[2] https://getcomposer.org

[3] https://getcomposer.org/Composer-Setup.exe

O seguinte comando, executado via terminal e na raiz de nosso projeto, deve instalar o composer para você!




O comando acima vai baixar e compilar o composer.phar, arquivos `.phar`[4] são extensões executáveis do PHP. Agora temos o arquivo de configuração e o Composer em nosso projeto, agora precisamos instalar nossas dependências, ou seja, no nosso caso o Silex. É muito simples realizar a instalação dos pacotes, na raiz do seu projeto execute o seguinte comando:

```
php composer.phar install
```

Pressuponho aqui que você tenha o php-cli disponivel em seu terminal. O camando acima irá verificar o arquivo `composer.json` e logo em seguida baixará o Silex para você, conforme requerido no arquivo ‘json, na versão 1.2. Após tudo concluído, você deve ter uma imagem semelhante a essa:

[4] http://php.net/phar

```
api-events: php composer.phar install
Loading composer repositories with package information
Installing dependencies (including require-dev)
  - Installing symfony/routing (v2.7.0)
    Downloading: 100%

  - Installing psr/log (1.0.0)
    Loading from cache

  - Installing symfony/debug (v2.7.0)
    Downloading: 100%

  - Installing symfony/http-foundation (v2.7.0)
    Downloading: 100%

  - Installing symfony/event-dispatcher (v2.7.0)
    Downloading: 100%

  - Installing symfony/http-kernel (v2.7.0)
    Downloading: 100%

  - Installing pimple/pimple (v1.1.1)
    Loading from cache

  - Installing silex/silex (v1.3.0)
    Downloading: 100%

symfony/routing suggests installing symfony/config (For using the all-in-one router or any loader)
symfony/routing suggests installing symfony/yaml (For using the YAML loader)
symfony/routing suggests installing symfony/expression-language (For using expression matching)
symfony/routing suggests installing doctrine/annotations (For using the annotation loader)
symfony/event-dispatcher suggests installing symfony/dependency-injection ()
symfony/http-kernel suggests installing symfony/browser-kit ()
symfony/http-kernel suggests installing symfony/class-loader ()
symfony/http-kernel suggests installing symfony/config ()
symfony/http-kernel suggests installing symfony/console ()
symfony/http-kernel suggests installing symfony/dependency-injection ()
symfony/http-kernel suggests installing symfony/finder ()
symfony/http-kernel suggests installing symfony/var-dumper ()
silex/silex suggests installing symfony/browser-kit (~2.3)
silex/silex suggests installing symfony/css-selector (~2.3)
silex/silex suggests installing symfony/dom-crawler (~2.3)
silex/silex suggests installing symfony/form (~2.3)
Writing lock file
Generating autoload files
```

O Composer instalou o Silex bem como as dependências utilizadas pelo mesmo. O Composer baixa e instala as dependências do nosso projeto dentro da pasta `vendor/`. Além de baixar, ele também cria/mapeia os namespaces dos pacotes e cria um autoload para nós, através desse autoload teremos acesso a todos os pacotes baixados até o momento.

## Silex: Hello World!

Bom! Agora que nossas dependências foram baixadas e instaladas, podemos começar a utilizar nosso micro fw. Crie um arquivo `index.php` na raiz do seu folder. Criarei o mesmo utilizando o seguinte

comando:

```
echo "<?php " > index.php
```

Acima crio o arquivo `index.php` e adiciono a abertura de nosso código PHP. Beleza, vamos continuar! Abaixo segue o código na integra do nosso index, em seguida comentarei o mesmo!

```
1  <?php
2
3  use Silex\Application;
4
5  require 'vendor/autoload.php';
6
7  $app = new Application();
8
9  $app->get('/', function(){
10         return 'Hello World';
11 });
12
13 $app->run();
```

Na linha 3 informo ao meu script que desejo utilizar o Silex, informando seu namespace `Silex\Application`. Para ter acesso aos namespaces dos pacotes baixados, como comentei anteriormente sobre o autoload, precisamos adicionar o mesmo em nosso index, para isso utilizamos o `require` na linha 5. Na linha 7 simplesmente instanciamos nosso micro fw. Como comentamos, o Silex possui um poderoso sistema de rotas e das linhas 9 a 11, utilizamos o método `get`. O método `get` manipula as requisições GET vindas do client, no nosso caso estamos fazendo o seguinte:

Quando o cliente realizar uma requisição do tipo GET em nossa rota raiz, referenciada através da `/`, nós queremos executar o que for passado dentro do callback, o segundo parâmetro do método `get` do `Silex\Application`. Como estamos querendo apenas realizar/printar um "Hello World", vamos retornar essa string em nosso callback para a rota raiz.

E por fim, para que as respostas emitidas pelo Silex sejam enviadas ao browser ou a quem as solicitou, utilizamos o método `run` em nossa linha 13. Ao rodar nosso app no browser, temos a seguinte resposta:

```
Hello World
```

Podemos perceber o quão simples é, utilizar esse micro framework, através dos processos aqui vistos. Em nossos próximos capítulos começaremos a aprofundar-nos mais e mais nesta ferramenta. Por hora, pratique os conhecimentos até aqui passados! Nos vemos no próximo capitulo!