



Jekyll

Geração de sites estáticos

Linguagens e Tecnologias Digitais

2 semestre

Daniel da Silva

@danielfdsilva

Introdução

O que é Jekyll?

Jekyll é um gerador de sites estáticos especialmente orientado para sites com coleções de dados (ex: blog).

A partir de ficheiros *markdown**, templates e pedaços de código reutilizáveis produz um site completo.

**Linguagem de formatação*

Lista de outros geradores: <https://www.staticgen.com/>



Processo geração



Processo geração



Vantagens

- Rápido (ficheiros todos pré-processados)
- Funciona em qualquer servidor html e conseqüentemente tem custos de alojamento reduzidos
- Seguro (não usa tecnologias de servidor)

Desvantagens

- Não permite usar tecnologias dinâmicas para comentários, perfis de utilizador, etc (mas existem alternativas)
- Requer alguma configuração

Estrutura do Jekyll

Estrutura de pastas

Existem 4 pastas principais:

`_includes` - pedaços de código reutilizáveis

`_layouts` - diferentes tipos de páginas (templates) que existem no site

`_posts` - pasta onde são colocados os artigos do site em formato markdown (no caso do blog)

`_site` - apenas aparece depois de executar o jekyll a primeira vez e é onde se encontra o resultado do processamento.

```
├── _includes
│   ├── footer.html
│   ├── head.html
│   ├── header.html
│   └── social.html
├── _layouts
│   ├── default.html
│   ├── home.html
│   ├── page.html
│   └── post.html
├── _posts
│   ├── 2016-05-19-super-short-article.md
│   ├── 2016-05-20-my-example-post.md
│   ├── 2016-05-20-super-long-article.md
│   ├── 2016-05-20-this-post-demonstrates-post-content-styles.md
│   └── 2016-05-20-welcome-to-jekyll.md
└── assets
    ├── _config.yml
    ├── 404.html
    └── index.md
```

Dica

Pastas começadas por **underscore _** não são copiadas para a pasta `_site`.

Estrutura de pastas

As restantes pastas (ex: `assets`) serão copiadas para a pasta final (`_site`).

Os conteúdos dessas pastas poderão ser processados pelo jekyll (se possuírem *front matter*) ou simplesmente copiados como no caso de imagens.

```
├─ _includes
│  ├── footer.html
│  ├── head.html
│  ├── header.html
│  └── social.html
├─ _layouts
│  ├── default.html
│  ├── home.html
│  ├── page.html
│  └── post.html
├─ _posts
│  ├── 2016-05-19-super-short-article.md
│  ├── 2016-05-20-my-example-post.md
│  ├── 2016-05-20-super-long-article.md
│  ├── 2016-05-20-this-post-demonstrates-post-content-styles.md
│  └── 2016-05-20-welcome-to-jekyll.md
└─ assets
   ├── _config.yml
   ├── 404.html
   └── index.md
```

Tipos de documentos

Existem 2 tipos de documentos de conteúdo:

posts - são ficheiros que fazem parte de uma coleção, têm uma nomenclatura específica (YYYY-MM-DD-titulo.md) e em ordem cronológica.

pages - páginas soltas (sobre, contacto, ajuda) sem relação direta entre si.

```
├─ _includes
│  ├── footer.html
│  ├── head.html
│  ├── header.html
│  └── social.html
├─ _layouts
│  ├── default.html
│  ├── home.html
│  ├── page.html
│  └── post.html
├─ _posts
│  ├── 2016-05-19-super-short-article.md
│  ├── 2016-05-20-my-example-post.md
│  ├── 2016-05-20-super-long-article.md
│  ├── 2016-05-20-this-post-demonstrates-post-content-styles.md
│  └── 2016-05-20-welcome-to-jekyll.md
└─ assets
   └─ _config.yml
      ├── 404.html
      └── index.md
```

Nota

Os documentos escritos em markdown serão convertidos em html automaticamente pelo jekyll antes de serem usados.

Ficheiros de trabalho

<https://bit.ly/2YSsfEt>

(Clone or Download > Download ZIP)



Comandos importantes

```
$ jekyll build
```

Gera o site estático e coloca os ficheiros resultantes na pasta `_site`.

```
$ jekyll serve --watch
```

Gera o site estático e coloca os ficheiros resultantes na pasta `_site`. Cria um servidor web local acessível em `http://localhost:4000`. Regenera o site a cada alteração efetuada num ficheiro.

Configuração

A configuração do jekyll é feita no ficheiro `_config.yml` usando a linguagem YAML.

Além de definições que o jekyll precisa para funcionar como `exclude` (ficheiros a ignorar) ou `future` (opção de publicar posts com data no futuro), este ficheiro pode ser usado para guardar as vossas próprias definições.

Estes valores podem depois ser usados através de `site.[nome]` (ex: `site.title`)

Valores por omissão em:

<https://jekyllrb.com/docs/configuration/default/>

```
title: My Jekyll DXD Blog
author: Daniel da Silva
email: email@example.com
description: > # this means to ignore newlines until "exclude"
  Write an awesome description for your new site here. This could
  for example be used for SEO purposes.
```

```
exclude:
  - Gemfile
  - Gemfile.lock
  - README.md
  - .gitignore
```

```
# Date format for posts
# refer to http://shopify.github.io/liquid/filters/date/ if you
# want to customize this
date_format: "%b %-d, %Y"
```

```
# generate social links in footer
social_links:
  twitter: daniel fdsilva
  github: daniel fdsilva
# facebook:
```

Dica

Coloquem neste ficheiro definições que utilizem várias vezes no site.

Exercício

(configuração do jekyll)





Conteúdo

Markdown

Linguagem de formatação que permite de forma simples estruturar documentos. É convertida automaticamente pelo jekyll em html aquando da geração do site.

Documentação:

<https://daringfireball.net/projects/markdown/syntax>

Markdown

```
# Título de nível 1
## Título de nível 2
### Título de nível 3
#### Título de nível 4
```

- *Lista não ordenada*
- *Mais um item na lista*
- *E outro*

Este é um pequeno parágrafo que demonstra algumas opções de markdown. Permite colocar **palavras em itálico**, usando apenas um asterisco antes e depois (também funciona com underscore). Também podemos destacar ****palavras a negrito**** usando dois asteriscos. Ahah! [Links](<https://google.pt>) também existem!

1. *E agora uma lista*
2. *que tem uma ordem*
3. *específica*

HTML

```
<h1><a id="Ttulo_de_nvel_1_0"></a>Título de nível 1</h1>
<h2><a id="Ttulo_de_nvel_2_1"></a>Título de nível 2</h2>
<h3><a id="Ttulo_de_nvel_3_2"></a>Título de nível 3</h3>
<h4><a id="Ttulo_de_nvel_4_3"></a>Título de nível 4</h4>
```

```
<ul>
  <li>Lista não ordenada</li>
  <li>Mais um item na lista</li>
  <li>E outro</li>
</ul>
```

```
<p>
  Este é um pequeno parágrafo que demonstra algumas opções de markdown. Permite
  colocar <em>palavras em itálico</em>, usando apenas um asterisco antes e
  depois (também funciona com underscore). Também podemos destacar
  <strong>palavras a negrito</strong> usando dois asteriscos. Ahah!
  <a href="https://google.pt">Links</a> também existem!
</p>
```

```
<ol>
  <li>E agora uma lista</li>
  <li>que tem uma ordem</li>
  <li>específica</li>
</ol>
```

Markdown

Título de nível 1

Título de nível 2

Título de nível 3

Título de nível 4

- *Lista não ordenada*
- *Mais um item na lista*
- *E outro*

Este é um pequeno parágrafo que demonstra algumas opções de markdown. Permite colocar **palavras em itálico**, usando apenas um asterisco antes e depois (também funciona com underscore). Também podemos destacar ****palavras a negrito**** usando dois asteriscos. Ahah! [Links](<https://google.pt>) também existem!

1. *E agora uma lista*
2. *que tem uma ordem*
3. *específica*

Versão renderizada

Título de nível 1

Título de nível 2

Título de nível 3

Título de nível 4

- Lista não ordenada
- Mais um item na lista
- E outro

Este é um pequeno parágrafo que demonstra algumas opções de markdown. Permite colocar *palavras em itálico*, usando apenas um asterisco antes e depois (também funciona com underscore). Também podemos destacar **palavras a negrito** usando dois asteriscos. Ahah! [Links](#) também existem!

1. E agora uma lista
2. que tem uma ordem
3. específica

Criação de posts

Os posts são criados dentro da pasta `_posts`.

O nome do post deve conter a data e o título no seguinte formato: `YYYY-MM-DD-titulo.md`

A extensão `.md` indica ao jekyll que o ficheiro contém markdown que necessita de processamento.

É a partir do nome que é construído o url para o post.

`2019-01-01-ola-mundo.md`

resulta em

`http://localhost:4000/2019/01/01/ola-mundo.html`

Criação de páginas

As páginas não têm de ser criadas dentro de uma pasta específica mas o url vai respeitar a pasta onde elas estiverem.

Exemplo:

Uma página `about.html` criada na raíz do site terá o url:

`http://localhost:4000/about.html`

Uma página `daniel.html` criada dentro da pasta `autores/` terá o url:

`http://localhost:4000/autores/daniel.html`



Estrutura

Qualquer post ou página de conteúdo vai ter duas secções obrigatórias:

- YAML Front Matter (delimitado por três hífenes `---`)
- Conteúdo (normalmente markdown, mas pode ser em html)

```
1 ---
2 layout: post
3 title: Olá Mundo
4 author: Daniel da Silva
5 ---
6
7 O meu primeiro post deste site e um pato.
8 ![imagem pato](/assets/images/pato.png)
9
```



Estrutura: YAML Front Matter

O YAML Front Matter é usado para configurar o post de modo a que o jekyll o saiba processar.

A única variável realmente obrigatória é `layout`. Através desta define-se qual o template a utilizar para gerar este post.

Qualquer outra variável é considerada parte dos metadados do post que são utilizadas nos templates. É o programador quem define quais os metadados necessários.

Estes valores podem depois ser usados através de `page.[nome]` (ex: `page.title`, `page.author`)

```
1 ---
2 layout: post
3 title: Olá Mundo
4 author: Daniel da Silva
5 ---
```

Exercício

(criação de um post)



Sistema de templates

Pode ser entendido como uma Matrioska em que um ficheiro de conteúdo usa um template que por sua vez pode usar outro template. É sempre gerado do mais pequeno para o maior.



Sistema de templates

2019-01-01-ola-mundo.md
(usa layout `post.html`)

O meu primeiro
post deste site
é um pato.



`post.html`
(usa layout `base.html`)

```
<posts relacionados>
{{content}}
<caixa comentários>
```

`base.html`
(não usa layout)

```
<html>
  <body>
    {{content}}
  </body>
</html>
```

Sistema de templates

1. Usando o primeiro ficheiro da cadeia perceber qual o template (`layout`) que usa.
2. Coloca o conteúdo desse ficheiro na zona de conteúdo do template (`{{content}}`).
3. Repetir até chegar ao fim da cadeia. (I.e. Ficheiro que não usa template)

Sistema de templates

2019-01-01-ola-mundo.md
(usa layout `post.html`)

O meu primeiro
post deste site
é um pato.



`post.html`
(usa layout `base.html`)

```
<posts relacionados>
{{content}}
<caixa comentários>
```

resultado

resultado intermédio
(usa layout `base.html`)

```
<posts relacionados>
```

```
O meu primeiro
post deste site
é um pato.
```



```
<caixa comentários>
```

Sistema de templates

resultado intermédio
(usa layout `base.html`)


```
<posts relacionados>  
  
O meu primeiro  
post deste site  
é um pato.  
  
  
  
<caixa comentários>
```

base.html
(não usa layout)

```
<html>  
<body>  
  
    {{content}}  
  
</body>  
</html>
```



ola-mundo.html
(gerado na pasta `_site`)

```
<html>  
<body>  
  
    <posts relacionados>  
  
        O meu primeiro  
        post deste site  
        é um pato.  
  
          
  
    <caixa comentários>  
  
</body>  
</html>
```

Liquid

Liquid é a linguagem de templating utilizada pelo jekyll na construção dos vários layouts.

Permite imprimir valores usando duas chavetas

`{{page.title}}`, efetuar verificações lógicas `{% if page.author %}`, ciclos `{% for post in site.posts %}`, etc...

Documentação:

<https://shopify.github.io/liquid/>

<https://github.com/Shopify/liquid/wiki/Liquid-for-Designers>

layout: default

```
<article class="post h-entry">
```

```
  <header class="post-header">
```

```
    <h1 class="post-title p-name">{{ page.title | escape }}</h1>
```

```
    <p class="post-meta">
```

```
      <time class="dt-published" datetime="{{ page.date | date_to_xmlschema }}">
```

```
        {%- assign date_format = site.date_format | default: "%b %-d, %Y" -%}
```

```
        {{ page.date | date: date_format }}
```

```
      </time>
```

```
      {%- if page.author -%}
```

```
        • <span class="p-author h-card">{{ page.author | escape }}</span>
```

```
      {%- endif -%}</p>
```

```
  </header>
```

```
  <div class="post-content e-content">
```

```
    {{ content }}
```

```
  </div>
```

```
</article>
```

Exercício

(listar posts na homepage)



Collections e Themes

Collections (<https://jekyllrb.com/docs/collections/>)


O jekyll permite criar outros “grupos” de dados além dos posts. Torna-se útil quando têm de gerir um site com uma estrutura mais complexa (Ex: membros de uma equipa, lista de projetos...)

Themes (<https://jekyllrb.com/docs/themes/>)

Permite de forma muito simples utilizar temas desenvolvidos pela comunidades. Também é possível criar os próprios temas, abstraíndo-os do conteúdo para que possam ser reutilizados.

Exercícios



- 
1. Criar uma página “*about*” com informações sobre o projeto.
 2. Criar uma variável no *YAML Front Matter* dos posts que permita definir uma cor para o título daquele post.
 3. Modificar a lista de posts da homepage para que apenas sejam listados posts com a variável de *YAML Front Matter* `featured: true`.