Narzędzia deweloperskie przeglądarki

Jekyll - narzędzie do tworzenia statycznych stron www

Instalacja

1. Wymagania wstępne:

- **Ruby:** Jekyll jest napisany w Ruby, więc musisz go mieć zainstalowanego. Zalecana wersja to Ruby 2.7 lub nowsza. Możesz sprawdzić wersję Ruby poleceniem `ruby -v` w terminalu. Jeśli nie masz Ruby, lub masz starszą wersję, zainstaluj go. Dla Windowsa, RubyInstaller jest dobrym wyborem. Dla macOS i Linuxa, użyj menedżera pakietów (np. `apt`, `brew`, `yum`). Na przykład, dla macOS z Homebrew: `brew install ruby`.
- **RubyGems:** RubyGems jest menedżerem pakietów dla Ruby i jest zazwyczaj instalowany razem z Ruby. Sprawdź, czy masz go zainstalowanego poleceniem `gem -v`. Jeśli nie, zainstaluj Ruby.
- **Bundler (opcjonalnie, ale zalecane):** Bundler pomaga zarządzać zależnościami projektu Jekyll. Zainstaluj go poleceniem `gem install bundler`.

2. Instalacja Jekylla:

Otwórz terminal (wiersz poleceń) i wpisz następujące polecenie: `gem install jekyll`. Zainstaluje ono Jekylla i jego podstawowe zależności. Na students.mimuw.edu.pl użyj polaceneń

```
export GEM_HOME=$HOME/gems

export PATH=$HOME/gems/bin:$PATH

gem install jekyll bundler
```

3. Utworzenie nowego projektu Jekyll:

Przejdź do katalogu, w którym chcesz utworzyć projekt (np. `cd ~/projekty`) i użyj polecenia: `jekyll new moja-strona`. Zamiast `moja-strona` możesz podać dowolną nazwę. To polecenie utworzy nowy katalog z podstawową strukturą projektu Jekyll.

4. Wejście do katalogu projektu:

```
cd moja-strona
```

5. Uruchomienie serwera Jekylla:

```
bundle exec jekyll serve
```

Lub, jeśli nie używasz Bundlera:

```
jekyll serve
```

To polecenie uruchomi lokalny serwer, który pozwoli Ci zobaczyć Twoją stronę w przeglądarce. Zazwyczaj strona będzie dostępna pod adresem `http://localhost:4000`.

6. Zatrzymanie serwera:

Aby zatrzymać serwer, naciśnij `Ctrl+C` w terminalu.

Tworzenie stron

Najprostszym sposobem utworzenia strony jest dodanie pliku HTML w katalogu głównym z odpowiednią nazwą. Można również napisać stronę w formacie Markdown, używając rozszerzenia `.md` i tak zwanym front matter, które podczas budowania konwertuje się na HTML. Dla witryny ze stroną główną, stroną "o mnie" i stroną kontaktową, tak mógłby wyglądać katalog główny i powiązane adresy URL: Markdown to prosty język znaczników, który pozwala na formatowanie tekstu w czytelny sposób.

Tworzenie postów

Podobnie jak zwykłe strony post mogą być tworzone w formacie HTML lub markdown.

1. Lokalizacja postów

- Posty w Jekyllu przechowywane są w folderze `_posts`.
- Każdy post musi mieć nazwę formacie `RRRR-MM-DD-tytuł-postu.md` (lub `.html`). Data w nazwie pliku jest bardzo ważna, ponieważ Jekyll używa jej do generowania adresów URL postów oraz do ich sortowania.

2. Front matter

- Każdy post w Jekyllu może zawierać tzw. front matter, czyli nagłówek w formacie YAML, który zawiera metadane o poście.
- Front matter umieszcza się na początku pliku, pomiędzy trzema myślnikami (`---`).
- W front matter możesz zdefiniować tytuł postu, datę publikacji (jeśli chcesz inną niż w nazwie pliku), kategorię, tagi, autora i inne niestandardowe pola.

- Więcej informacji o front matter na można znaleźć na stronie: https://jekyllrb.com/docs/front-matter/

3. Przykład postu w formacie Markdowns

```
---
title: Mój pierwszy post na Jekyllu
date: 2024-05-16
categories:
  - Jekyll
  - Blogowanie
tags:
  - Markdown
  - Tworzenie stron
author: Jan Kowalski
---

To jest treść mojego pierwszego postu na Jekyllu.

Używam Markdowna, aby formatować tekst.

# Nagłówek pierwszego stopnia

## Nagłówek drugiego stopnia

*   Lista punktowana
*   Lista punktowana

[Link do Google](https://www.google.com)

![Obrazek](https://www.google.com/images/branding/googlelogo/2x/google
logo_color_272x92dp.png)
```

4. Dodatkowe informacje

- Jekyll oferuje wiele możliwości konfiguracji postów, takich jak paginacja, archiwizacja, generowanie kanału RSS itp.
- Więcej informacji na temat tworzenia postów w Jekyllu znajdziesz w oficjalnej dokumentacji: https://jekyllrb.com/docs/posts/

Kolekcje

Kolekcje w Jekyllu to potężne narzędzie, które pozwala na grupowanie powiązanych treści, takich jak przepisy kulinarne, członkowie zespołu, produkty w sklepie internetowym czy galerie zdjęć. Dzięki kolekcjom możesz dynamicznie generować listy i strony, a także zarządzać metadanymi dla wielu elementów jednocześnie.

Jak działają kolekcje?

1. **Definicja kolekcji:** Kolekcje definiuje się w pliku `_config.yml` w głównym katalogu projektu Jekyll. Można tam określić nazwę kolekcji, folder, w którym będą przechowywane pliki, oraz inne opcje konfiguracyjne.

2. **Tworzenie plików:** Pliki należące do kolekcji umieszcza się w folderze określonym w konfiguracji. Każdy plik reprezentuje jeden element kolekcji. Pliki te mogą być w formacie Markdown lub HTML i zawierać front matter z metadanymi.

3. **Wyświetlanie kolekcji:** Aby wyświetlić listę elementów kolekcji na stronie, używa się pętli `for` w szablonach Liquid. Można iterować po elementach kolekcji, wyświetlać ich tytuły, opisy, zdjęcia i inne dane.

4. **Generowanie stron:** Jekyll może automatycznie generować strony dla poszczególnych elementów kolekcji. Wystarczy odpowiednio skonfigurować permalinki w pliku `_config.yml`.

Przykład użycia kolekcji:

Załóżmy, że chcemy stworzyć kolekcję przepisów kulinarnych. W pliku `_config.yml` dodajemy następującą konfigurację:

YAML
```yaml
collections:
  recipes:
    output: true
    permalink: /przepisy/:path/
```

Następnie tworzymy folder `_recipes` i umieszczamy w nim pliki z przepisami, np. `_recipes/ciasto-czekoladowe.md`:

Markdown

```
---
title: Ciasto czekoladowe
ingredients:
  - mąka
  - cukier
  - jajka
  - czekolada
---

Przepis na ciasto czekoladowe...
```

Aby wyświetlić listę przepisów na stronie, używamy pętli Liquid:

Code snippet

```
<ul>
  {% for recipe in site.recipes %}
    <li><a href="{{ recipe.url }}">{{ recipe.title }}</a></li>
  {% endfor %}
</ul>
```

Pliki z danymi

Pliki z danymi (Data Files) to wygodny sposób na przechowywanie danych, które chcesz wykorzystać na swojej stronie internetowej. Zamiast umieszczać dane bezpośrednio w kodzie HTML lub Markdown, możesz przechowywać je w oddzielnych plikach YAML, JSON lub CSV, a następnie dynamicznie wczytywać te dane do szablonów Liquid. To podejście ułatwia zarządzanie danymi, ich aktualizację i wielokrotne wykorzystywanie w różnych częściach witryny.

Jak działają pliki danych?

1. **Umieszczanie plików:** Pliki danych umieszcza się w folderze `_data` w głównym katalogu projektu Jekyll. Możesz tworzyć podfoldery wewnątrz `_data`, aby lepiej organizować dane.

2. **Formaty plików:** Jekyll obsługuje trzy formaty plików danych:

    - **YAML:** Najczęściej używany format, czytelny i łatwy do edycji.
    - **JSON:** Format popularny w aplikacjach internetowych, łatwy do parsowania przez JavaScript.

    - **CSV:** Format tabelaryczny, przydatny do przechowywania danych w formie tabeli.
3. **Struktura danych:** Dane w plikach mogą mieć dowolną strukturę. Możesz przechowywać listy, obiekty, wartości proste itp. Ważne jest, aby struktura była spójna i logiczna, aby łatwo było z niej korzystać w szablonach.

4. **Wczytywanie danych:** Dane z plików są automatycznie dostępne w szablonach Liquid pod nazwą folderu (jeśli taki jest) i nazwą pliku (bez rozszerzenia).

Do zawartości pliku `_data/users.yml`:

```
- name: Jan Kowalski
  age: 30
- name: Anna Nowak
  age: 25
```

można odwołać się w szablonie Liquid w następujący sposób:

```
{% for user in site.data.users %}
  <p>{{ user.name }} ({{ user.age }} lat)</p>
{% endfor %}
```

Folder `assets` w Jekyllu służy do przechowywania statycznych plików, takich jak pliki CSS, JavaScript, obrazy, czcionki i inne zasoby, które są wykorzystywane na Twojej stronie internetowej. Organizacja tych plików w folderze `assets` pomaga w utrzymaniu porządku w projekcie i ułatwia zarządzanie zasobami.

Jak działają assets w Jekyllu?

1. **Umiejscowienie:** Wszystkie pliki statyczne, które chcesz dołączyć do swojej strony, umieszczasz w folderze `assets`. Zaleca się tworzenie wewnątrz folderu `assets` podfolderów, takich jak `css`, `js`, `img` (lub `images`), `fonts` itp., aby jeszcze lepiej zorganizować pliki.

2. **Dostęp:** Do plików w folderze `assets` odwołujesz się za pomocą ścieżek relatywnych. Jekyll automatycznie przetwarza te ścieżki i generuje poprawne adresy URL do Twoich zasobów.

3. **Przetwarzanie (opcjonalne):** Jekyll może przetwarzać niektóre typy plików w folderze `assets`, np. pliki SCSS mogą być kompilowane do CSS, a pliki JavaScript mogą być minifikowane. Konfiguruje się to w pliku `_config.yml`.

Jak odwoływać się do assets w kodzie?

W Twoich plikach HTML, Markdown lub Liquid możesz odwoływać się do plików w folderze `assets` za pomocą ścieżek relatywnych.

Przykłady:

- **CSS:**

```
<link rel="stylesheet" href="/assets/css/style.css">
```

- **JavaScript:**

```
<script src="/assets/js/script.js"></script>
```

- **Obraz:**

```
<img src="/assets/img/logo.png" alt="Logo">
```

Korzystanie z tagu `asset` (zalecane):

Jekyll oferuje tag `asset`, który jest *zalecany* do odwoływania się do plików w folderze `assets`. Tag ten ma kilka zalet:

- **Wersjonowanie:** Dodaje do nazw plików hash, co ułatwia cache'owanie i unika problemów ze starymi wersjami plików.
- **Elastyczność:** Umożliwia łatwą zmianę lokalizacji folderu `assets` w przyszłości.

Przykłady użycia tagu `asset`:

- **CSS**

```
<link rel="stylesheet" href="{{ '/assets/css/style.css' | asset_url }}">
```

- **JavaScript:**

```
<script src="{{ '/assets/js/script.js' | asset_url }}"></script>
```

- **Obraz:**

Code snippet

```
<img src="{{ '/assets/img/logo.png' | asset_url }}" alt="Logo">
```

Struktura folderów

… powinna wyglądać mniej-więcej tak:

```
.
├── _config.yml
├── _data
│   └── members.yml
├── _drafts
│   ├── begin-with-the-crazy-ideas.md
│   └── on-simplicity-in-technology.md
├── _includes
│   ├── footer.html
│   └── header.html
├── _layouts
│   ├── default.html
│   └── post.html
├── _posts
│   ├── 2007-10-29-why-every-programmer-should-play-nethack.md
│   └── 2009-04-26-barcamp-boston-4-roundup.md
├── _sass
│   ├── _base.scss
│   └── _layout.scss
├── _site
├── .jekyll-cache
│   └── Jekyll
│       └── Cache
│           └── [...]
├── .jekyll-metadata
└── index.html # can also be an 'index.md' with valid front matter
```

Liquid

Liquid to język szablonów, który jest integralną częścią Jekylla. Pozwala on na dynamiczne wstawianie treści i danych podczas generowania statycznych stron

Podstawowe elementy Liquid:

- **Obiekty:** Obiekty reprezentują dane, które są dostępne w szablonie. Dostęp do nich uzyskuje się za pomocą podwójnych nawiasów klamrowych `{{ }}`. Na przykład `{{`

`page.title }}` wyświetli tytuł strony.

- **Tagi:** Tagi służą do wykonywania operacji logicznych, takich jak pętle, warunki, przypisania zmiennych itp. Tagi otacza się nawiasami klamrowymi z procentami `{% %}`. Na przykład `{% for post in site.posts %}` rozpoczyna pętlę, która iteruje po wszystkich postach na stronie.

- **Filtry:** Filtry modyfikują sposób wyświetlania obiektów. Stosuje się je po obiekcie, oddzielając je pionową kreską `|`. Na przykład `{{ post.date | date_to_string }}` sformatuje datę postu do czytelnego formatu,

Najczęściej używane tagi Liquid:

- `{% if warunek %}` ... `{% endif %}`: Warunek logiczny.
- `{% for element in kolekcja %}` ... `{% endfor %}`: Pętla iterująca po kolekcji.
- `{% assign zmienna = wartość %}`: Przypisanie wartości do zmiennej.
- `{% include 'nazwa-pliku.html' %}`: Włączenie innego pliku (partial).
- `{% capture zmienna %}` ... `{% endcapture %}`: Przechwycenie wyniku do zmiennej.

Najczęściej używane filtry Liquid:

- `date_to_string`: Formatuje datę do czytelnego formatu.
- `capitalize`: Zmienia pierwszy znak na wielką literę.
- `downcase`: Zmienia wszystkie znaki na małe litery.
- `upcase`: Zmienia wszystkie znaki na wielkie litery.
- `truncate`: Skraca tekst do określonej długości.
- `newline_to_br`: Zamienia znaki nowej linii na tagi `<br>`.

Przykład użycia Liquid w Jekyllu:

```
<h1>{{ page.title }}</h1>

<p>Opublikowano: {{ page.date | date_to_string }}</p>

<ul>
  {% for post in site.posts %}
    <li>
      <a href="{{ post.url }}">{{ post.title }}</a>
    </li>
  {% endfor %}
```

```
</ul>
```

W tym przykładzie:

- `{{ page.title }}` wyświetla tytuł strony.
- `{{ page.date | date_to_string }}` wyświetla datę publikacji strony w czytelnym formacie.
- Pętla `{% for post in site.posts %}` iteruje po wszystkich postach na stronie i dla każdego postu wyświetla tytuł i link.

Includes

`Includes` w Jekyllu to mechanizm, który pozwala na włączanie fragmentów kodu (tzw. partials) do szablonów i postów. To bardzo przydatne narzędzie, które pomaga w utrzymaniu porządku w kodzie, ułatwia jego ponowne wykorzystanie i zmniejsza redundancję.

Jak działają includes?

1. **Tworzenie partials:** Partiale to pliki, które zawierają fragment kodu HTML, Liquid lub Markdown. Zazwyczaj umieszcza się je w folderze `_includes` w głównym katalogu projektu Jekyll. Nazwa pliku partiala powinna być opisowa i sugerować jego przeznaczenie (np. `_includes/header.html`, `_includes/footer.html`, `_includes/sidebar.html`).

2. **Włączanie partiali:** Aby włączyć partial do szablonu lub postu, używa się tagu Liquid `{% include nazwa-pliku.html %}`. Jekyll w momencie generowania strony zastępuje ten tag zawartością pliku partiala.

3. **Przekazywanie parametrów (opcjonalne):** Do partiali można przekazywać parametry, co pozwala na ich dynamiczną konfigurację. Używa się do tego składni `{% include nazwa-pliku.html parametr1="wartość1" parametr2="wartość2" %}`. Wewnątrz partiala można odwoływać się do tych parametrów za pomocą `include.parametr1`, `include.parametr2` itd.

Przykład użycia:

Załóżmy, że chcemy stworzyć partial z nagłówkiem strony. Tworzymy plik `_includes/header.html` z następującą zawartością:

```
<header>
  <h1>{{ site.title }}</h1>
```

```
  <nav>
    <ul>
      <li><a href="/">Strona główna</a></li>
      <li><a href="/o-mnie/">O mnie</a></li>
      <li><a href="/kontakt/">Kontakt</a></li>
    </ul>
  </nav>
</header>
```

Następnie, aby włączyć ten partial do szablonu, np. `_layouts/default.html`, dodajemy tag `{% include header.html %}` w odpowiednim miejscu:

```
<!DOCTYPE html>
<html>
<head>
  <title>{{ page.title }}</title>
</head>
<body>
  {% include header.html %}

  <main>
    {{ content }}
  </main>

  {% include footer.html %}
</body>
</html>
```

Przykład z parametrami:

Tworzymy partial `_includes/alert.html`:

```
<div class="alert alert-{{ include.type }}">
  {{ include.message }}
</div>
```

A następnie używamy go w szablonie:

```
{% include alert.html type="warning" message="To jest ostrzeżenie!" %}
```

```
{% include alert.html type="success" message="Operacja zakończona sukcesem!" %}
```

Layouts

Layouty (ang. layouts) w Jekyllu to szablony, które definiują strukturę i wygląd Twoich stron internetowych. Ułatwiają one zarządzanie wspólnymi elementami, takimi jak nagłówek, stopka, menu nawigacyjne itp.

Jak działają layouty?

1. **Tworzenie layoutów:** Layouty to pliki HTML, które zazwyczaj umieszcza się w folderze `_layouts` w głównym katalogu projektu Jekyll. Nazwa pliku layoutu powinna być opisowa (np. `default.html`, `post.html`, `page.html`).

2. **Struktura layoutu:** Layout zawiera podstawową strukturę HTML strony, w tym znaczniki `<html>`, `<head>`, `<body>` itp. Wewnątrz layoutu umieszcza się tag Liquid `{{ content }}`, który w momencie generowania strony jest zastępowany treścią konkretnej strony lub postu.

3. **Wykorzystanie layoutu:** Aby przypisać layout do strony lub postu, należy określić go w front matter danego pliku. Na przykład:

```
---

layout: default

title: Moja strona

---

Treść mojej strony.
```

Przykład layoutu `default.html`:

```
HTML
<!DOCTYPE html>
<html>
<head>
  <title>{{ page.title }}</title>
  <link rel="stylesheet" href="{{ '/assets/css/style.css' | asset_url }}">
```

```
</head>
<body>
  <header>
    {% include header.html %}
  </header>

  <main>
    {{ content }}
  </main>

  <footer>
    {% include footer.html %}
  </footer>
</body>
</html>
```

W tym przykładzie layout `default.html` zawiera nagłówek (wczytywany za pomocą include'a), treść strony (wstawianą za pomocą `{{ content }}`) oraz stopkę (również wczytywaną za pomocą include'a).

Generowanie strony

1. **Uruchom Jekyll:** Uruchom komendę `jekyll serve` w terminalu, aby wygenerować stronę i uruchomić lokalny serwer.
2. **Podgląd strony:** Otwórz przeglądarkę i wejdź na adres `http://localhost:4000`, aby zobaczyć swoją stronę.

Publikacja strony

1. **Przygotowanie do publikacji:** Po wygenerowaniu strony, zawartość katalogu `_site` zawiera gotowe pliki HTML, które można umieścić na serwerze.
2. **Wybór hostingu:** Wybierz hosting, który obsługuje statyczne strony internetowe, np. GitHub Pages. Można też umieścić zawartość strony na serwerze students w katalogu `public_html`. Opis jak to zrobić znajduje się na stronie laboratorium komputerowego: https://lk.mimuw.edu.pl/pl/studenci/wlasna-strona-www

3. **Wgranie plików:** Wgraj zawartość katalogu `_site` na serwer.