Flask

Jan Kałucki
Michał Jankowski

Czym jest Flask?

- Framework webowy
- Dostępny od 2010 roku
- Obecnie w wersji 3.0.2
- Dostępny dla pythona>=3.8
- Korzysta z Werkzeug, Jinja, MarkupSafe, ItsDangerous, Click oraz Blinker

Główne cechy

- Flask to microframework
- Flask w celu routingu używa dekoratorów (np. app.route(‘/<variable>’))
- Flask korzysta z konceptu blueprintów (schematów)
- Flask nie oczekuje użycia konkretnego systemu bazy danych - nie posiada wbudowanego ORM (Object Relational Mapping)
- Flask korzysta z Jinja2 i Werkzeug

Szablony

Szablony

```
<ul>
    {% for item in items %}
        <li>{{ item.attr1 }} some other text {{ item.other_param }}</li>
    {% endfor %}
</ul>
```

- Flask korzysta z Jinja2
- Charakterystycznym elementem Jinja są {}. W domyślnej konfiguracji:
- {{ variable }} wypisuje wartość variable
- {# komentarz #} jest komentarzem
- {% kod %} jest kodem do wykonania
- Jinja oferuje wyrażenia takiej jak:
  - for*
  - if* / elif / else
  - block*
  - macro*
  - extends
  - include
  - * - obszar działania do endfor, endif, …

```
items: list[dict[str, str]] = [
    {"attr1": "AAAAA", "other_param": "11111"},
    {"attr1": "BBBBB", "other_param": "22222"},
    {"attr1": "CCCCC", "other_param": "33333"}
]

@app.route('/')
def home() -> str:
    return render_template('index.html', items=items)
```

Szablony

- Block można określić jako wymagany (required) - niedokonanie podmiany zakończy się błędem
- Jinja wspiera możliwość importowania macro z innych plików niczym python - nazwę źródła podaje się tak jak w przypadku extends
- Include jest przydatnym tagiem - można określić jakiś element jako opcjonalny (ignore missing), lub dostarczyć listę do wstawienia - jeśli nie ma pierwszego użyty jest drugi, jeśli nie ma go to trzeci, i tak dalej

```
{% block content %}
    Base
{% endblock %}
```

```
{% extends "base.html" %}

{% block content %}
{{ super() }}
<br/>
Added by child
{% endblock %}
```

Base
Added by child

```
<h1>TITLE</h1>
```

```
{% include "title.html" %}
<p>Content</p>
```

TITLE

Content

Szablony

- Pewne zadania można uprościć poprzez użycie filtrów lub makr - jednakże nie zawsze są najlepszym rozwiązaniem

```
{% macro hello(name) %}
Hello, {{ name }} <br/>
{% endmacro %}
{{ hello("Alice") }}
{{ hello("Bob") }}
```

Hello, Alice
Hello, Bob

```
{% set x = "heLLO wORLd!" %}
{{ x|capitalize }} <br/>
{{ x|lower }} <br/>
{{ x|upper }} <br/>
{{ x|random }} <br/>
{{ x|sort }} <br/>
{{ x|truncate(10, True, leeway=0) }} <br/>
{# 10 characters including ..., cut where indicated, cut without leeway#}
```

Hello world!
hello world!
HELLO WORLD!
w
[' ', '!', 'd', 'e', 'h', 'L', 'L', 'L', 'O', 'O', 'R', 'w']
heLLO w...

Najciekawsze moduły i ich cechy

Najciekawsze moduły i ich cechy

- Flask-WTF (WTForms):
  - biblioteka do tworzenia i obsługi formularzy
  - posiada wsparcie dla reCAPTCHA oraz zabezpieczenia przed CSRF
- Flask-Login
  - ułatwia wprowadzenie zalogowanych sesji
- Flask-Security-Too
  - Fork Flask-Security, obecnie bardziej rozwinięty od oryginału
  - Scala kilka modułów
  - Zawiera wiele zasobów pomagających dodać zabezpieczenia
- Flask-Caching

Najciekawsze moduły i ich cechy

- Flask-Babel
  - Pozwala ułatwić lokalizację i internacjonalizację
- Flask-Mailman
  - Bazowany na module z Django
- Flask-Admin
  - Przydatny w tworzeniu interfejsu do zarządzania aplikacją
- Flask-RESTful
  - Ułatwia stworzenie REST API
- Flask-Talisman - o tym więcej pod koniec

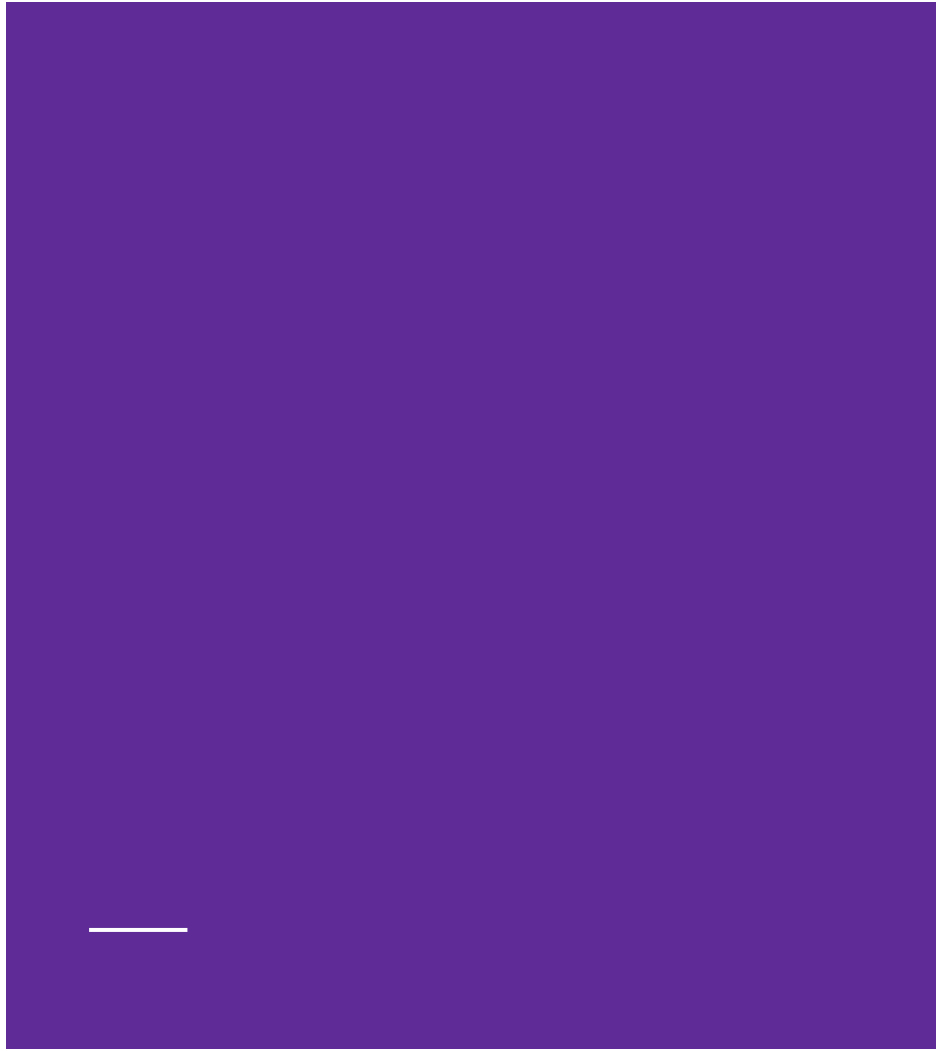
Zalety i wady

Zalety i wady

- Lekki
- Wspiera WSGI (poprzez Werkzeug)
- Idealny dla małych projektów
- Wiele rozszerzeń
- (W miarę) łatwy w użyciu

- Brak wbudowanej obsługi zabezpieczeń
- Słaba skalowalność dla większych projektów

Flask vs Django



Flask vs Django

- (Mikro)framework
- Elastyczny pod kątem stylów aplikacji
- Nie posiada domyślnego modelu bazy danych
- Posiada debugger i wbudowany serwer rozwojowy
- Idealny dla małych projektów
- Daje więcej kontroli

- Rozbudowany framework
- Posiada wbudowane systemy bezpieczeństwa
- Współpracuje z popularnymi systemami baz danych (np. MySQL)
- Zdolny do obsługi dużych ilości ruchu
- Wiele funkcjonalności
- Dobry do większych projektów
- Prostszy w użyciu przy dużych projektach

Bezpieczeństwo we Flasku

Bezpieczeństwo we Flasku - XSS i CSRF

- Domyślnie Flask, poprzez Jinja, zabezpiecza się przed XSS w większości przypadków
- Nie dotyczy to jednak:
  - wywołań Markup
  - wysyłania plików zawierających HTML - w pewnym stopniu zwalczalne poprzez Content-Disposition: attachment
  - atrybutów tagów HTML - ich wartości trzeba wstawiać poprzez atrybut=“{{wartość}}”
  - atrybutu href tagu a
- Tag a można zabezpieczyć poprzez Content Security Policy (CSP)
- Sam Flask nie zabezpiecza przed Cross-Site Request Forgery (CSRF)
  - Flask-WTForms oraz Flask-Security-Too są pomocne w walce z CSRF

Bezpieczeństwo we Flasku - Nagłówki

Flask-Talisman jest przydatny w postawieniu pewnych zabezpieczeń

Domyślnie zastosuje pewne zabezpieczenia, w tym:

- HTTP Strict Transport Security (HSTS) - przeciwko MITM, wymusza HTTPS
- X-Frame-Options - blokuje osadzanie strony w iframe
- X-Content-Options - przeciwko XSS
- Content Security Policy - przeciwko XSS
- Dostęp do ciasteczek zostanie ograniczony tylko do HTTPS - przeciwko CSRF

Źródła

- https://flask.palletsprojects.com/en/3.0.x/
- https://flask.palletsprojects.com/en/3.0.x/blueprints/
- https://flask-wtf.readthedocs.io/en/1.2.x/
- https://wtforms.readthedocs.io/en/3.1.x/
- https://flask-login.readthedocs.io/en/latest/
- https://flask-caching.readthedocs.io/en/latest/
- https://python-babel.github.io/flask-babel/
- https://waynerv.github.io/flask-mailman/
- https://flask-admin.readthedocs.io/en/latest/
- https://flask-restful.readthedocs.io/en/latest/
- https://flask-security-too.readthedocs.io/en/stable/index.html
- https://www.simplilearn.com/flask-vs-django-article
- https://github.com/GoogleCloudPlatform/flask-talisman
- https://jinja.palletsprojects.com/en/3.1.x/templates/