Flask

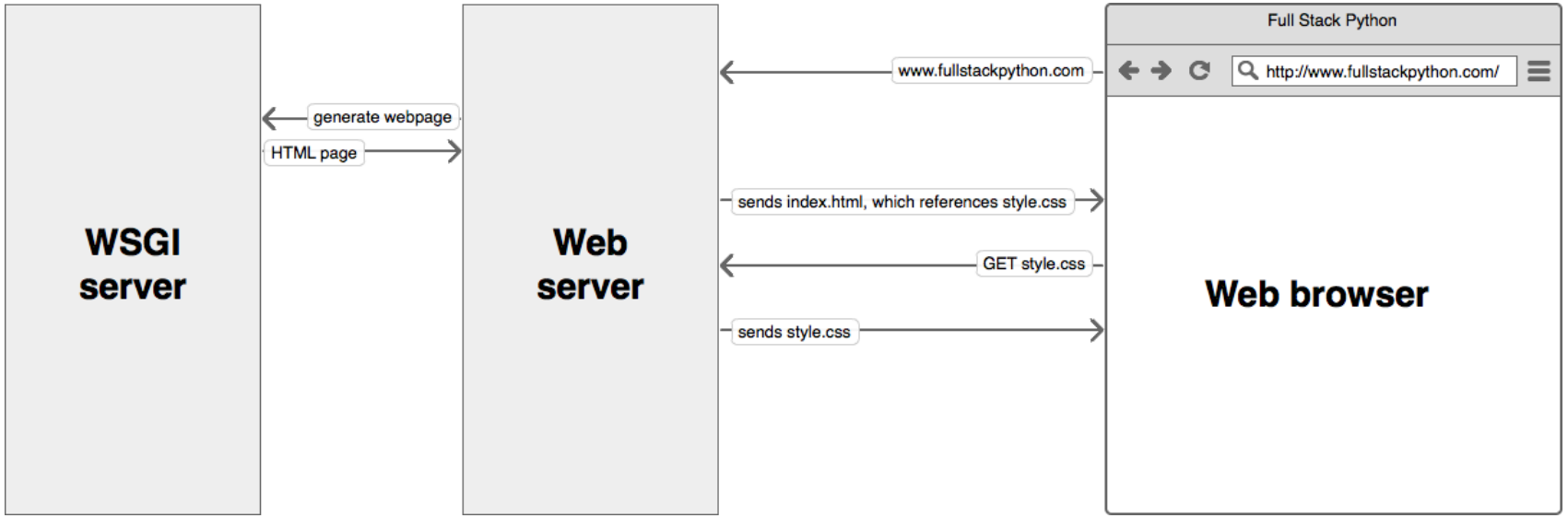
Paweł Gliwny

Flask - web framework

- **Web Framework**, czyli Framework internetowy, to zbiór bibliotek i modułów umożliwiających tworzenie aplikacji internetowych bez konieczności martwienia się o szczegóły na niskim poziomie, takie jak protokoły czy zarządzanie wątkami.
- **Flask** to framework do tworzenia aplikacji internetowych napisany w języku Python.
- **Flask** opiera się na narzędziu WSGI Werkzeug oraz silniku szablonów **Jinja2**.

WSGI oraz Werkzeug

- **WSGI**, czyli Web Server Gateway Interface, został przyjęty jako standard dla tworzenia aplikacji internetowych w Pythonie. WSGI jest specyfikacją uniwersalnego interfejsu między serwerem internetowym a aplikacjami internetowymi.
- **Werkzeug** to zestaw narzędzi WSGI, który implementuje obiekty żądania, odpowiedzi oraz inne funkcje narzędziowe. Można budować framework internetowy na jego podstawie. Flask korzysta z Werkzeug jako jednej z jego podstaw.



Szablony

- **Jinja2** to popularny silnik szablonów dla języka Python. System szablonowania internetowego łączy szablon z określonym źródłem danych, aby renderować dynamiczne strony internetowe.
- **Flask** jest często określany jako mikro-framework. Ma na celu zachowanie prostoty rdzenia aplikacji, ale jednocześnie zapewnia możliwość jego rozszerzania.
- **Flask** nie posiada wbudowanej warstwy abstrakcji do obsługi bazy danych, ani nie posiada wsparcia dla walidacji formularzy. Zamiast tego, **Flask** obsługuje rozszerzenia, które dodają takie funkcjonalności do aplikacji.

Pierwsza aplikacja

app = Flask(__name__) - Tworzy instancję aplikacji Flask.

- **Dekorator @app.route():**
  - Określa, że funkcja poniżej dekoratora jest funkcją widoku, która zostanie wywołana, gdy klient odwiedzi daną ścieżkę (URL).
  - @app.route('/') - Główny URL aplikacji wywołuje funkcję hello_world.
- **Funkcja Widoku:**
  - def hello_world(): - Definicja funkcji, która jest przypisana do określonego URL.
  - return 'Hello, World!' - Odpowiedź, którą przeglądarka otrzyma po wejściu na główną stronę.
- **Uruchamianie Aplikacji:**
  - app.run() - Uruchamia lokalny serwer rozwojowy.
  - Odwiedzając URL http://127.0.0.1:5000/, zobaczymy wiadomość 'Hello, World!'.

```
from flask import Flask

app = Flask(__name__)


@app.route('/')

def hello_world():

  return 'Hello World’


if __name__ == '__main__':

  app.run()
```

Parametry Metody app.run():

- host='127.0.0.1': Domyślny adres hosta; ustawienie na '0.0.0.0' sprawia, że serwer jest dostępny z zewnątrz.
- port=5000: Domyślny port serwera.

Tryb Debugowania:

- Aby ułatwić rozwój, włącz tryb debugowania, który pozwala na auto-ładowanie serwera i dostęp do debugera błędów.
- Aktywacja: app.run(debug=True) lub app.debug = True; app.run().

Routing w Flask

- Dekorator **@route()** w Flasku jest używany do powiązania określonego adresu URL z funkcją.
- Jest to bardzo intuicyjny sposób na określenie, co powinno się stać, gdy użytkownik odwiedzi określony adres w aplikacji.

```
@app.route('/hello')
def hello_world():
    return 'hello world'
```

- W tym przypadku, odwiedzając URL http://localhost:5000/hello, zostanie wyświetlona odpowiedź funkcji hello_world(), czyli w przeglądarce pojawi się tekst "hello world".

Zmienne w adresach URL w Flask

- W **Flask** można dynamicznie budować adresy URL, dołączając zmienne do parametru reguły adresu URL.
- Część zmiennej jest oznaczona w nawiasach trójkątnych, np. <nazwa_zmiennej>, i przekazywana jako argument do funkcji.
- W **Flask** można określać typ zmiennej w adresie URL przy pomocy konwerterów:
  - **int** - akceptuje liczby całkowite
  - **float** - dla wartości zmiennoprzecinkowych
  - **path** - akceptuje ukośniki, traktując je jako separator katalogów

```
from flask import Flask

app = Flask(__name__)


@app.route('/blog/<int:postID>')

def show_blog(postID):

  return 'Blog Number %d' % postID


@app.route('/rev/<float:revNo>')

def revision(revNo):

  return 'Revision Number %f' % revNo


if __name__ == '__main__':

  app.run()
```

- Wprowadzenie http://localhost:5000/blog/11 wyświetli w przeglądarce "Blog Number 11",

Budowanie URLi w Flask przy użyciu funkcji url_for()

- Funkcja **url_for()** w Flasku pozwala dynamicznie budować URL-e, które prowadzą do określonych funkcji obsługujących trasy w aplikacji.
- Aby skorzystać z funkcji, należy przekazać nazwę funkcji jako pierwszy argument. Kolejne argumenty to zmienne, które odpowiadają dynamicznym częściom adresu URL.
- Funkcja **url_for()** pozwala na elastyczne budowanie i zarządzanie trasami w aplikacji webowej.

URL http://localhost:5000/user/admin: Użytkownik zostaje przekierowany do trasy /admin, która odpowiedzi "Hello Admin".

URL http://localhost:5000/user/mvl: Użytkownik zostaje przekierowany do trasy /guest/mvl, która odpowiedzi "Hello mvl as Guest".

```python
@app.route('/admin')
def hello_admin():
   return 'Hello Admin'

@app.route('/guest/<guest>')
def hello_guest(guest):
   return 'Hello %s as Guest' % guest

@app.route('/user/<name>')
def hello_user(name):
   if name =='admin':
      return redirect(url_for('hello_admin'))
   else:
      return redirect(url_for('hello_guest',guest = name))
```

Źródła

- tutorialspoint flask