

Politechnika Świętokrzyska

Kielce, 2023r.

Przedmiot:

Technologie obiektowe

Tytuł projektu:

Parser Yaml w Python

Grupa projektowa:

Jakub Francuz

Grupa dziekańska:

1IZ21

1) Założenia systemu

System ma za zadanie parsować (tłumaczyć) struktury zapisane w składni języka znacznikowego YAML do obiektów python. Nazwano go **yapy** (**Y**aml-to-**P**ython) oraz odwrotnie tj. Tłumaczenia struktur python do składni YAML. Całość została napisana zgodnie z paradygmatem obiektowym (OOP) - tak jak zostało to przedstawione w wytycznych projektowych przez prowadzącego zajęcia z przedmiotu *Technologie Obiektowe*.

2) Technologie

Do napisania oprogramowania wykorzystano:

- IDE – Visual Studio Code
- Język programowania Python wraz z bibliotek standardową
- Web framework Flask
- Powłoka bash shell
- System kontroli wersji Git

3) Funkcjonalności

System ma za zadanie przetworzyć struktury YAML'owe do struktur obiektowych języka Python. Całość interakcji z systemem opiera się na wpisywaniu/wklejaniu struktur YAML lub Python do okien tekstowych, wygenerowanych we frameworku webowym Flask oraz naciśnięciu przycisku pod oknem tekstowym (button) wykonującym akcję i przekierowującym użytkownika do strony z treścią wyjściową procesu parsowania wyświetlającą rezultat końcowy - oczekiwany.



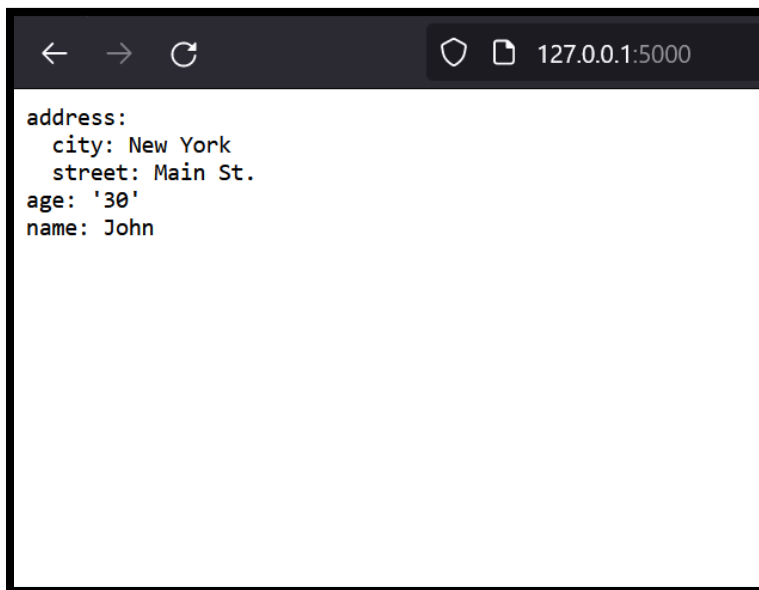
Rys. 3.1) Widok interfejsu webowego z poziomu przeglądarki na hoście lokalnym

Python -> Yaml

```
{'name': 'John', 'age': '30', 'address': {'street':  
'Main St.', 'city': 'New York'}}
```

Parse to YAML

Rys. 3.2) Wprowadzenie danych struktury Python do parsowania



```
address:  
  city: New York  
  street: Main St.  
age: '30'  
name: John
```

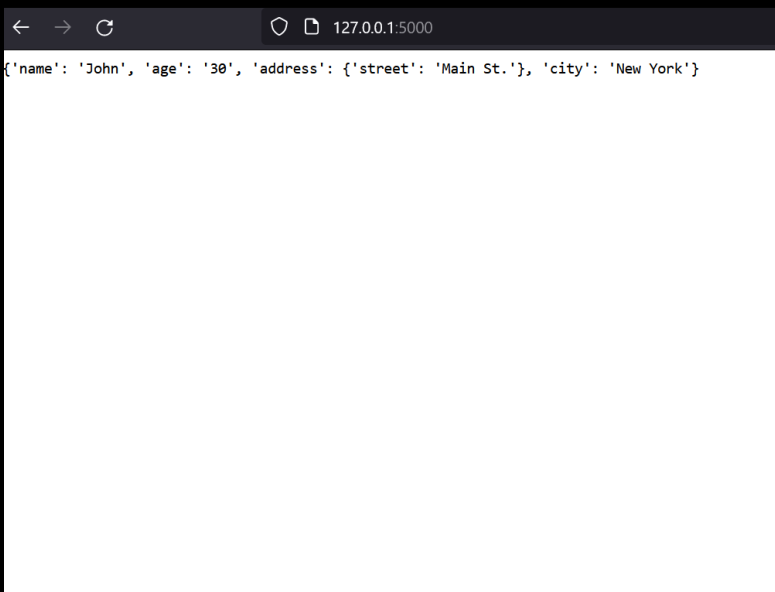
Rys. 3.3) Efekt parsowania struktury Python do YAML

Yaml -> Python

```
name: John  
age: 30  
address:  
  street: Main St.  
  city: New York|
```

Parse to Python

Rys. 3.4) Wprowadzenie danych struktury YAML do parsowania



The screenshot shows a web browser window with a dark header bar. The address bar displays a shield icon, a document icon, and the URL '127.0.0.1:5000'. The main content area of the browser shows a single line of Python code representing a dictionary: `{'name': 'John', 'age': '30', 'address': {'street': 'Main St.', 'city': 'New York'}}`.

Rys. 3.5) Efekt parsowania struktury YAML do obiektu Python

4) Uruchamianie aplikacji

Projekt jest obecny w repozytorium Github pod adresem: <https://github.com/jqbFrnz/yapy.git>

Można go pobrać np. Za pomocą polecenia *git clone*

Będąc w głównym folderze z plikami źródłowymi przechodzimy do folderu *.venv/Scripts/*

A następnie z poziomu powłoki bash uruchamiamy skrypt *activate*, są w tym folderze dostępne również skrypty umożliwiające aktywację z poziomu windowsowego *cmd* lub *powershell*. Teraz uruchomione jest wirtualne środowisko python z wszystkimi zależnościami potrzebnymi do uruchomienia web servera *Flask* na środowisku *localhost*.

```
minds@DESKTOP-K7GTHLI MINGW64 ~/Desktop/yappy (master)
$ pwd
/c/Users/minds/Desktop/yappy
(.venv)
minds@DESKTOP-K7GTHLI MINGW64 ~/Desktop/yappy (master)
$ cd .venv/Scripts/
(.venv)
minds@DESKTOP-K7GTHLI MINGW64 ~/Desktop/yappy/.venv/Scripts (master)
$ activate
(.venv)
minds@DESKTOP-K7GTHLI MINGW64 ~/Desktop/yappy/.venv/Scripts (master)
$
```

Rys. 4.1) Uruchomienie wirtualnego środowiska projektowego Python w bash

Następnie wracamy do folderu z plikami kodu źródłowego, tam gdzie obecny jest plik *hello.py* i uruchamiamy projekt poleceniem:

```
flask --app hello run
```

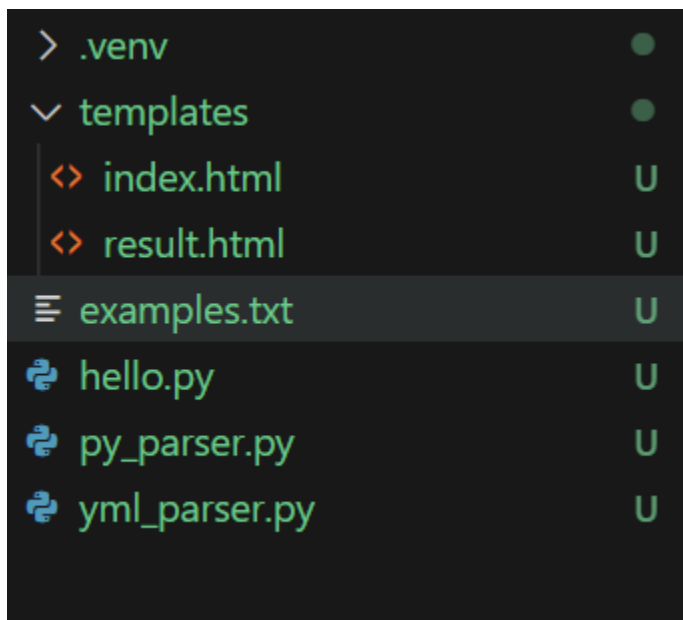
Uruchomiony serwer wskazuje nam adres i port na którym mamy dostęp do aplikacji webowej z systemem do parsowania. (127.0.0.1:5000). W konsoli będą widoczne zapytania *http/https* wysyłane do serwera oraz kody *http* zwracane w ramach interakcji z serwerem (przydatne do debugowania).

```
minds@DESKTOP-K7GTHLI MINGW64 ~/Desktop/yappy (master)
$ flask --app hello run
* Serving Flask app 'hello'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [16/Jun/2023 21:15:37] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [16/Jun/2023 21:15:44] "POST / HTTP/1.1" 200 -
```

Rys. 4.2) Widok z konsoli po uruchomieniu serwera z widocznymi rezultatami zapytań GET I POST

5) Implementacja

Struktura projektu prezentuje się w następujący sposób:



Rys. 5.1) Struktura plików projektowych

Opis funkcjonalności:

W pliku **py_parser.py** znajduje się klasa *PythonParser* odpowiedzialna za funkcjonalności związane z parsowaniem struktur języka Python do struktur YAML. Jej główna metoda to *parse_to_yaml()* - analizuje ona input struktury Python wprowadzonej w oknie tekstowym i na jego podstawie transformuje go do formatu YAML.

W pliku **yaml_parser.py** znajduje się klasa *YamlParser* odpowiedzialna za funkcjonalności związane z parsowaniem struktur języka Python do struktur YAML. Jej główna metoda to *parse_to_python()* - analizuje ona input struktury YAML wprowadzonej w oknie tekstowym i na jego podstawie transformuje go do formatu obiektu języka docelowego, w tym przypadku do Pythona.

Plik **hello.py** stanowi główny entry point programu, definiuje on ścieżkę (route) dla web frameworka python, która odpowiada za obsługę zapytań HTTP/HTTPS Get oraz Post dla root katalogu (/) aplikacji. Tworzone są w nim 2 obiekty klas Parsera YAML jak i Python. W zależności od tego które okna tekstowe zostaną wypełnione i który przycisk kliknięty przez użytkownika - dokonują one obsłużenia (handlingu) logiki serializacji lub deserializacji wprowadzonych struktur.

Plik **index.html** to główny plik, w którym renderowany jest dynamicznie przez Flaska główny interfejs programu. Posiada on formularze html, na których podstawie, w przypadku wypełnienia strukturami przez użytkownika obsługiwane są dynamicznie przez Flask za pomocą sprecyzowanych wyrażeń warunkowych.

Plik **output.html** zostaje wypełniony danymi wyjściowymi dla parsowania w zależności od wybrania typu parsowania. Użytkownikowi zostaje automatycznie zwrócona ta strona przez aplikację po kliknięciu przycisku dla wybranego typu parsowania. W pliku znajduje się jeden znacznik html `<pre>{{ yaml_output }}</pre>` - przechowuje on wartość renderowaną przez silnik JINJA2 dostępny w frameworku Flask.

5) Wnioski

Projekt pozwolił zapoznać się z wzorcami projektowymi oraz rozwinąć świadomość odnośnie obiektowego paradygmatu w kontekście programowania. Całość projektu udało się zrealizować zgodnie z zadaną tematyką oraz wskazanymi wytycznymi.