3.06.2022

Dziś zapoznamy się ze specyfikacją Common Gateway Interface (CGI) oraz z jej pythonowym rozwinięciem Web Server Gateway Interface (WSGI) i zobaczymy jak w stworzyć serwer REST w Pythonie.

- W Pythonie 3 dostępny jest standardowy moduł `http.server` który poza tworzeniem skryptów działających jako serwer HTTP pozwala też łatwo stworzyć domyślny serwer http:

  ```
  python -m http.server
  ```

  Zapoznaj się z działaniem i dokumentacją tego modułu. Uruchom serwer w wybranym katalogu i otwórz w przeglądarce adres `localhost:8000`.

- Zapoznaj się z opcją `--cgi` modułu `http.server`. Utwórz w wybranym katalogu podkatalog `cgi-bin` a w nim prosty sktypt w Pythonie (albo skrypt/program dowolnym innym języku) wypisujący dowolny tekst na standardowe wyjście. Uwaga: aby program zadziałał poprawnie z CGI, wypisywany tekst musi być poprzedzony przynajmniej następującą linią (zawierającą nagłówek HTTP):

  ```
  "Content-type: text/html\r\n\r\n"
  ```

- Uruchom serwer HTTP z opcją `--cgi` w katalogu zawierającym `cgi-bin` i otwórz w przeglądarce adres `localhost:8000/cgi-bin/nazwa_skryptu.py`

- Zamiast w Pythonie, stwórz program obsługujący zapytania w C (skompiluj go i umieść plik wykonywalny w `cgi-bin/`). Sprawdź, czy CGI działa z binarnym plikiem wykonywalnym.

- Zapoznaj się z listą zmiennych środowiskowych ustawianych przez serwer HTTP przy wywołaniu CGI. Spróbuj odczytać niektóre z nich z poziomu swojego programu i wypisać je w odpowiedzi HTTP.

- Uważnie przeanalizuj i uruchom przykłady demonstrujące implementowanie REST-owych usług w środowisku WSGI: hello_webapp.py, rest_webapp.py, rest_webapp.sh.

- Zwróć uwagę na to, w jaki sposób metody tej klasy nawzajem się wywołują w zależności od tego, jakie zapytanie przyszło od klienta. Część z nich, np. sql_select, jest wywoływana z kilku różnych miejsc. Jeśli się w tym zgubisz spróbuj rozrysować na kartce papieru graf wywołań, to powinno pomóc.

- Uruchom rest_webapp.py bezpośrednio, bez pomocy załączonego skryptu. Wyślij do uruchomionego serwera kilka zapytań i sprawdź, czy dostajesz takie odpowiedzi, jakich się spodziewałaś(-eś).

- Spróbuj znaleźć w powyższym kodzie rzeczy, które serwer przyjmuje na wiarę, bez weryfikacji. Zastanów się, czy napastnik mógłby je wykorzystać aby włamać się do serwera lub w inny sposób zakłócić jego pracę.

- Rozszerz aplikację tak, aby można było wyszukiwać osoby o zadanym imieniu i/lub nazwisku (tzn. zaimplementuj obsługę URL-i postaci `/osoby/search?imie=Adam&nazwisko=Nowak`).

- (alternatywa dla powyższego zadania, jeśli ktoś nie lubi Pythona) Sprawdź, czy Twój ulubiony język programowania wspiera tworzenie aplikacji REST. Jeśli tak, odszukaj i przejrzyj przykład takiego serwera, albo i dwa przykłady. Jak przekazywane są informacje o szczegółach zapytania, i jak zwracana jest wygenerowana odpowiedź? Zaimplementuj w tym środowisku odpowiednik hello_webapp.py.

- Przejrzyj dokumentację pythonowych modułów xmlrpc.client i xmlrpc.server. Zorientuj się (w ogólnych zarysach) jak w Pythonie pisze się programy korzystające z XML-RPC.

- Zastanów się, w jaki sposób można byłoby przepisać naszą przykładową bazę osób z REST na XML-RPC. Jak musiałby zmienić się jej interfejs, czym zastąpić przesyłanie dokumentów w formacie TSV?
- (nieobowiązkowe) Spróbuj dokonać takiej reimplementacji. Nie musi być pełna, wystarczy zaimplementować obsługę jednej-dwóch zdalnych metod; już to powinno Państwu pozwolić opanować podstawy korzystania z tych nowych bibliotek.