ZSUT

Zakład Sieci i Usług Teleinformatycznych

Podstawy komutacji cyfrowej

Instrukcja

Ćwiczenie: Klient REST

ZSUT. Zakład Sieci i Usług Teleinformatycznych
Instytut Telekomunikacji
Wydział Elektroniki i Technik Informacyjnych
Politechnika Warszawska, Listopad 2020

Cel ćwiczenia

Celem ćwiczenia jest implementacja prostego klienta będącego w stanie porozumieć się z serwerem REST.

Wprowadzenie

1. Internet

Standardowy użytkownik wykorzystuje sieć Internet do połączenia się ze źródłem, z którego dane pobiera. Następnie korzysta ze standardowego narzędzia, które mu te dane zaprezentuje. W ostatnim kroku użytkownik samodzielnie te dane konsumuje. Przykładem może być tu wizyta na stronie elka.pw.edu.pl, czyli połączenie się z serwerem o adresie 194.29.160.81, ściągnięcie treści strony na swoje urządzenie, zaprezentowanie jej przez przeglądarkę w formacie, który twórca strony uznał za najlepszy, a na samym końcu przeczytanie treści strony.

Załóżmy, że nasz standardowy użytkownik, nazwany tutaj studentem, zdecydował się inwestować na giełdzie. Przeczytał książkę o aktywnym inwestowaniu, opracował jakąś skomplikowaną regułę, na podstawie której zamierza akcje kupować, usiadł do komputera, otworzył stronę giełdy https://www.gpw.pl/akcje, zobaczył, że notowanych jest paręset spółek i zorientował się, że na kartce tego wszystkiego nie policzy. Szczęśliwie student umie już programować i wie, że komputer zrobi to za niego błyskawicznie, a on musi tylko dostarczyć dane.

Student ochoczo bierze się do pracy. Potrafi już napisać program, który czyta z pliku, więc zaczyna tworzyć plik tekstowy z nazwami spółek, kursami akcji i pozostałymi danymi wymaganymi przez jego skomplikowaną formułę inwestycyjną. Po kwadransie orientuje się, że proces pozyskiwania danych lepiej zautomatyzować, więc zaczyna rozgryzać komunikację swojego urządzenia z serwerem informacyjnym rozpoczynając swoją drogę do stania się ponadstandardowym użytkownikiem.

2. HTTP/HTTPS

Student zauważył, że adresy wszystkich stron, które do tej pory odwiedził, rozpoczynają się w pasku przeglądarki od https://. HTTP to podstawowy protokół w świecie WWW (World Wide Web). Służy on do organizacji komunikacji pomiędzy klientem a serwerem, a jego skrót rozwija się do Hypertext Transfer Protocol. HTTPS to wersja „bezpieczna” HTTP, gdzie S rozwija się do Secure. Student poczytał o HTTP (np. [1,2]) i zorientował się, że komendy GET i POST protokołu HTTP całkowicie mu wystarczą.

Student poeksperymentował trochę z aplikacją ncat.exe [3]. Wysłał do serwera Elki zapytanie o stronę elka.pw.edu.pl/index.html. To samo uczynić dla strony gpw.pl/akcje. O ile ściągnięcie strony Elka powiodło się, o tyle dostanie się do zasobów GPW okazało się kłopotliwe. Po otrzymaniu zapytania GET, strona GPW poprosiła studenta o wykorzystanie protokołu HTTPS, który jest znacznie bardziej skomplikowany od HTTP. Podobna sytuacja spotkała studenta na wielu innych stronach. Student już wiedział, że do obsługi HTTP/HTTPS użyje bibliotek.

3. HTML

Po ściągnięciu strony giełdy student zaczął się zastanawiać, jak wyciągnąć z niej potrzebne informacje. Szybko orientuje się, że HTML (Hypertext Markup Language) nie jest idealnym językiem w komunikacji pomiędzy maszynami, a jego głównym celem jest ułatwienie konsumowania danych przez człowieka.

4. RESTful API

Po krótkiej analizie sytuacji student natrafia na koncepcję REST (REpresentational State Transfer). REST został zaprojektowany w celu ułatwienia komunikacji pomiędzy maszynami. Zbudowany jest na protokole HTTP/HTTPS, który organizuje komunikację. Do reprezentacji danych używa się m.in. JSON czy XML. Student znalazł dostawcę danych giełdowych wystawiającego API (Application Programming Interface), słono zapłacił za dostęp w czasie rzeczywistym i wziął się do pracy.

5. Postman

Aby zaznajomić się z zakupionym API student wykorzystał aplikację Postman [4], która pozwala w łatwy sposób wysyłać zapytania HTTP i testować API. Po przetestowaniu i zrozumieniu zakupionego API, student wziął się za wybór narzędzi programistycznych.

6. Python

Nie drążąc zbytnio motywacji studenta, zdecydował się on na język Pythona i bibliotekę http.client.

7. Liczenie zysków i strat

Po udanym zaimplementowaniu klienta REST okazało się, że rzeczywistość brutalnie zweryfikowała skuteczność wymyślonej przez studenta strategii inwestycyjnej. Całe szczęście, że przy okazji nauczył się tworzyć klientów REST, po jakimś czasie serwery REST, potem jeszcze wiele innych ciekawych rzeczy. Pozostało tylko jedno zmartwienie, jak skutecznie zainwestować zgromadzony na nowo kapitał.

Treść zadania

W czasie laboratorium wcielimy się w rolę opisanego studenta rozpoczynając o punktu 4, tj. Zakup API. W ćwiczeniu korzystać będziemy z uproszczonego, przykładowego serwera giełdowego ZSUT znajdującego się pod adresem:

https://stockserver.azurewebsites.net/

Dokumentacja API serwera znajduje się pod adresem:

https://documenter.getpostman.com/view/12940619/TVRkb8Kg

Celem ćwiczenia jest napisanie klienta REST, który w przemyślany sposób wykona zestaw transakcji, który skutkować będzie zredukowaniem stanu konta ze 100000 do 90000. Wynikiem ćwiczenia jest kod klienta oraz ściągnięty plik JSON zawierający ocenę wraz z komentarzem oraz listę dokonanych transakcji (api/history). Kod oraz plik JSON należy umieścić na wydziałowym gitlabie:

https://gitlab-stud.elka.pw.edu.pl

Projekt należy zatytułować **2020Z_PKC_REST**. Proszę nie zapomnieć o dodaniu do projektu prowadzącego laboratorium, tj. **mzotkiew.** Laboratorium składa się z 4 kroków. Wszystkie oprócz pierwszego powinny zakończyć się commitem.

Opis i dokumentacja przebiegu ćwiczenia są zbędne.

Przebieg ćwiczenia

Ćwiczenie można wykonywać samodzielnie lub w grupach dwuosobowych. W przypadku grup wymagany jest jeden projekt w gitlabie do którego wkład wnoszą wszyscy członkowie zespołu.

UWAGA: Nawet w przypadku grup, ćwiczenie oceniane jest indywidualnie, tj. wymagane transakcje muszą być zrealizowane na kontach wszystkich członków grupy.

1. Przygotowanie środowiska pracy

Zainstaluj oprogramowanie niezbędne do wykonania ćwiczenia: Python, aplikację Postman.

Zadanie można wykonać również korzystając z innego języka programowania. Sugerujemy jednak język Python, gdyż w kolejnych ćwiczeniach będziemy korzystali właśnie z tego języka.

2. Wykorzystanie aplikacji Postman

2.1. Rejestracja na serwerze giełdowym

Wykorzystaj aplikację Postman w celu zarejestrowania się na serwerze giełdowym.

2.2. Przetestowania funkcjonalności serwera

Korzystając z aplikacji Postman wykonaj następujące operacje:

a. Sprawdź stan swojego konta (Client data) i dostępne giełdy

b. Sprawdź udziały notowane na jednej z giełd

c. Sprawdź aktualny kurs udziałów jednej ze spółek

d. Złóż ofertę na udziały tej spółki na rozważanej giełdzie

e. Sprawdź historię swoich transakcji i aktualny stan konta

Uwaga: Oferty są zmienne w czasie, a inni użytkownicy mogą być od Was szybsi w generowaniu ofert.

3. Implementacja klienta REST

3.1. Implementacja podstawowej wersji klienta

Napisz klienta REST, który dokona co najmniej jednej transakcji.

Wskazówka: przykładowy kod ściągający listę dostępnych parkietów znajduje się poniżej.

```
import http.client
import mimetypes

conn = http.client.HTTPSConnection("stockserver.azurewebsites.net")
payload = ''
headers = {}
conn.request("GET", "/api/stockexchanges", payload, headers)
res = conn.getresponse()
data = res.read()
print(data.decode("utf-8"))
```

Warto też korzystać z dokumentacji “zakupionego” API.

3.2. Implementacja rozbudowanej wersji klienta

W tej części należy w przemyślany sposób zapętlić kod, aby liczba wykonanych transakcji przekroczyła 39.

3.3. Implementacja ostatecznej wersji klienta

O ile do realizacji poprzednich punktów nie były potrzebne wszystkie funkcje dostępne w API, o tyle w ostatecznej wersji klienta wykorzystanie do tej pory nieużywanych funkcjonalności może okazać się koniecznością. W tej części należy tak zaprojektować agenta, aby wykonał on takie transakcje, które będą skutkowały stanem konta w granicach 89999-90001.

UWAGA: do stanu konta zaliczamy tylko wolne środki.

Punktacja

25% - skuteczna rejestracja na serwerze

25% - napisanie klienta REST, który wykona co najmniej jedną operację na serwerze (commit; punkt 3.1)

25% - napisanie klienta REST, który dokona co najmniej 40 transakcji (commit; punkt 3.2)

25% - napisanie klienta REST, który zakończy działanie, gdy stan konta będzie równy 90000.0 (+- 1.0); do stanu konta zaliczamy tylko wolne środki (commit; punkt 3.3)

Czas realizacji: 8.12 20:00

Literatura

[1] https://developer.mozilla.org/en-US/docs/Web/HTTP

[2] https://www.samouczekprogramisty.pl/protokol-http/

[3] https://nmap.org/ncat/

[4] https://www.postman.com/product/rest-client/