Testowanie backendu narzędziami Postman, SoapUI, JMeter - wprowadzenie

Szkolenie przeprowadzone będą w formie warsztatów, gdzie będzie położony bardzo duży nacisk na praktykę. Poza praktyką związaną z poznawaniem podstawowych funkcji narzędzi takich jak Postman SoapUI oraz JMeter uczestnik powinien być przygotowany na dużą dawkę samodzielnego programowania w 2 językach:

- JavaScript – wykorzystywany w narzędziu Postman
- Groovy – wykorzystywany w narzędziu SoapUI

Program warsztatów obejmuje zakres zapoznania się z podstawowymi funkcjami programów Postman, SoapUI wersja free, JMeter.

Aktualnie najpopularniejszym sposobem komunikacji frontendu z backendem jest restowe API w formie komunikatów w formacie JSON i takiej też komunikacji będziemy używać podczas warsztatów.

Zatem poznasz zarówno najpopularniejsze narzędzia do testowania backendu jak i najpopularniejszy sposób komunikacji z backendem 😊

W trakcie warsztatów dowiesz się:

1. Jak tworzyć test case'y wywołujące restowe requesty GET, POST, PUT
2. Jak zarządzać zmiennymi wykorzystywanymi w requestach.
3. Jak wykonywać wiele kroków w ramach jednego test case'a
4. Jak tworzyć asercje sprawdzające zawartość JSONa w responsie
5. Jak tworzyć asercje sprawdzające oczekiwany stan w bazie danych [SoapUI]
6. Jak testować requesty wykonujące upload plików binarnych [SoapUI, Postman, JMeter?]
7. Jak tworzyć test casy, które wczytują dane wejściowe w plików CSV [SoapUI]
8. Jak tworzyć mocki, które symulują część jeszcze niegotowego systemu [SoapUI, Postman]
9. Jak wykonywać zapytania do bazy danych za pomocą konstrukcji SQL [SoapUI]
10. Jak czytać dokumentacje API wygenerowaną za pomocą narzędzia Swagger
11. Jak pisać proste asercje i jak manipulować zmiennymi w JavaScript [Postman]
12. Jak pisać proste i nieco bardziej złożone asercje oraz jak manipulować zmiennymi w Groovy [SoapUI]

W celu sprawnego przeprowadzenia warsztatów są następujące wymagania:

- Bardzo podstawowa wiedza na temat zapytań SQL
- Styczność z programowaniem i znajomość podstawowych pojęć programistycznych – zmienna, pętla, funkcja, klasa, biblioteka, przypisanie, mapa
- Chęć programowanie w JavaScript w celu pisania asercji w Postmanie
- Chęć programowania w Groovie w celu pisania asercji w SoapUI

Zadanie 1 [Postman, SoapUI, JMeter] - Skonfiguruj narzędzia

1. [SoapUI] Ściągnij SoapUI Open Source ze strony: https://www.soapui.org/downloads/soapui.html, aktualnie najnowsza wersja to SoapUI 5.5.0
2. [SoapUI] Z otrzymanych materiałów skopiuj plik ***postgresql-42.2.5.jar*** do katalogu ***SoapUI-5.5.0/bin/ext***. Plik ***postgresql-42.2.5.jar*** jest biblioteką pozwalającą wykonać połączenie do bazy danych.
3. [Postman] Ściągnij Postmana ze strony https://www.getpostman.com/downloads/
4. [JMeter] Ściągnij JMeter ze strony https://jmeter.apache.org/download_jmeter.cgi lub skopiuj z materiałów plik apache-jmeter-5.1.1.zip i rozpakuj w wybranym miejscu na komputerze.
5. [Baza danych] Ściągnij ***DBeaver Community Edition*** ze strony https://dbeaver.io/download/ - jest to narzędzie pozwalające przeglądać dane na bazie danych.
6. [Baza danych] Skonfiguruj połączenie i zobacz czy działa

   Namiary do połączenia się z bazą danych:

Host	51.38.129.181
Port	8200
Database	pkko_norad
User	pkko
Password	pkko

(czas 30min)

Zadanie 2 [Postman, SoapUI, JMeter] - Dokumentacja API

Zapoznaj się z dokumentacją API dostępną na stronie http://51.38.129.181:8100/norad/swagger-ui.html

(czas: 10min)

Zadanie 3 [SoapUI] – Import Swagger do SoapUI

1. Uruchom SoapUI
2. Utwórz nowy projekt w SoapUI o nazwie ***qa-training***
3. Pobierz dokumentację API do Norad w postaci struktury JSON, dostępna na stronie http://51.38.129.181:8100/norad/admin/swagger
4. Zapisz ją na dysku.
5. W SoapUI zaimportuj plik z dokumentacją.
6. Utwórz Test Suite o nazwie ***"Norad Test Suite"***, w którym będziesz tworzyć testy w trakcie warsztatów.

(czas: 10min - SoapUI)

SOAPUI – ZADANIA

Zadanie 4 Pobierz listę ras

Utwórz TestCase o nazwie ***"Exercise 4 – Get all races"***, a w nim przygotuj request, który będzie pobierał listę wszystkich ras dostępnych pod zasobami "***dictionaries***"

Asercje:

A. Dodaj asercję sprawdzającą czy http code = 200
B. dodaj asercję sprawdzającą czy ilość elementów na liście w JSONie zwrotnym **jest równa 10**
C. dodaj asercję sprawdzającą czy na 5 pozycji nazwa kota to "Kot egzotyczny"

Hint: Stronka wspierająca w czytaniu JsonPath: https://jsonpath.com/

(czas: 15min)

Zadanie 5 Zarejestruj nowego użytkownika

Utwórz nowy TestCase o nazwie ***"Exercise 5 – Register new user"***, który zawierać będzie request wywołujący utworzenie nowego użytkownika.

Hint: Przykład zawartości body requestu tworzącego użytkownika odnajdziesz w dokumentacji Swagger na stronie http://51.38.129.181:8100/norad/swagger-ui.html

A) Zmień TestCase w taki sposób, ażeby można było wywołać request utworzenia **wielokrotnie.**

Czyli e-mail nie może się powtarzać, za każdym wywołaniem powinien generować się nowy unikalny ze stałym prefixem - przyjmijmy Twoja pierwsza litera imienia i nazwisko. Na przykład: ***mszpiler***

Hint: Przeszukaj projekt ***postman-soapui-jmeter-qa-training*** w katalogu groovy znajdują się przykłady, które mogą Tobie pomóc rozwiązać zadanie.

B) Dodaj asercję sprawdzającą, że w odpowiedzi będzie w strukturze json w polu **email** istniała wartość oczekiwana wygenerowana przed wywołaniem requesta tworzącego nowego użytkownika.

C) Dodaj asercję sprawdzającą za pomocą skryptu Groovie czy rekord w bazie danych istnieje

- Połącz się z bazą danych
- Pobierz parametr email z kroku pierwszego, w którym email jest generowany i wykorzystany podczas rejestracji konta
- Wykonaj zapytanie SQL
- Zastosuj polecenie assert

D) Zapisz UUID użytkownika w propertiesach w kontekście Test Suite pod zmienną **userUUID**.

(czas: 45min)

Zadanie 6 Dodaj nowe zwierzę do nowego użytkownika

Utwórz nowy TestCase ***"Exercise 6 – Add new animal for new user"***, który zawierać będzie następujące kroki:

Krok 1. Rejestracja nowego użytkownika.

Krok 2. Utworzenie losowej nazwy zwierzęcia ze stałym prefixem na przykład pierwsza litera imienia i nazwisko np. **mszpiler .**

Prefix pozwoli lepiej identyfikować Twoje dane w bazie danych, która jest wspólna dla wszystkich uczestników szkolenia.

Krok 3. Pobierz wybrany losowo element z listy ras i zastosuj go przy tworzeniu zwierzęcia. Zapisz go w kontekście Test Case'a w propertiesach pod zmienną **raceId**

Krok 4. Dodaj nowe zwierzę dla użytkownika z kroku 1, dodane zwierzę powinno zawierać 3 zdjęcia oraz rasę ustaloną w kroku 3.

Krok 5. Zapisz uuid zwierzęcia w propertiesach w kontekście **Test Suite** pod zmienną **animalUUID**.

Hint: Zobacz jaka relacja jest między encjami user, animal i image. Zobacz w dokumentacji zawartość DTO dla utworzenia nowego zwierzęcia.

Asercje

A. Sprawdź czy w bazie zwierzę posiada oczekiwaną wartość określającą rasę.
B. Sprawdź czy w bazie zwierzę powiązane jest z użytkownikiem utworzonym w pierwszym kroku.
C. Sprawdź czy w odpowiedzi lista zdjęć jest oczekiwana i wynosi 3
D. Sprawdź czy każde ze zdjęć posiada niepuste pole uuid

(czas: 1,5h)

Zadanie 7 - Wykonaj upload zdjęcia - załącznik wskazany ręcznie

Przygotuj Test Case, który będzie wykonywał test uploadu wybranego contentu zdjęcia na serwer. Wykorzystaj encję zdjęcia utworzonego w zadaniu 6

Krok 1. Dodaj step "Upload file", który wywołuje endpoint odpowiedzialny za upload plików na serwer.

Krok 2. Pobierz UUID pierwszego zdjęcia oraz UUID zwierzęcia z zadania numer 6 i określ propertiesy **animalUUID** oraz **imageUUID** w kontekście stepa ***"Upload file***".

Krok 3. Dodaj załącznik ręcznie i wykonaj request, który wykona upload obrazka na serwer.

Asercje

A. Sprawdź czy odpowiedź http = 200
B. Utwórz request pobrania obrazka i sprawdź czy uzyskany obrazek posiada oczekiwany rozmiar - czyli zgodny z tym rozmiarem pliku jaki został użyty podczas uploadu zdjęcia.

(czas: 30min)

Zadanie 8 - Wykonaj upload zdjęcia - załącznik wskazany za pomocą skryptu Groovy

Przygotuj Test Case, który będzie wykonywał test uploadu wybranego contentu zdjęcia na serwer. Wykorzystaj encję zdjęcia utworzonego w zadaniu 6.

Krok 1. Dodaj step "Upload file", który wywołuje endpoint odpowiedzialny za upload plików na serwer.

Krok 2. Pobierz UUID pierwszego zdjęcia oraz UUID zwierzęcia z zadania numer 6 i określ propertiesy **animalUUID** oraz **imageUUID** w kontekście stepa ***"Upload file***".

Krok 3. Określ dwa custom properties na poziomie projektu:

- ***imageFilePath*** - ścieżka do pliku ze zdjęciem
- ***imageFileName*** **-** nazwa pliku ze zdjęciem

Krok 4. Dodaj załącznik za pomocą skryptu Groovy i wykonaj request, który wykona upload obrazka na serwer.

Asercje

A. Sprawdź czy odpowiedź http = 200
B. Utwórz request pobrania obrazka i sprawdź czy uzyskany obrazek posiada oczekiwany rozmiar - czyli zgodny z tym rozmiarem pliku jaki został użyty podczas uploadu zdjęcia.

(czas: 30 min)

Zadanie 9 - Dodaj zwierzęta pobierając dane z pliku CSV

Utwórz nowy TestCase, który utworzy wiele zwierząt do konkretnego użytkownika.

Wykorzystaj metodę gotoStep oraz korzystanie z pętli w Groovy.

Lista zwierząt znajduje się w pliku CSV. Dane na temat zwierzęcia w pliku, to nazwa zwierzęcia oraz rasa.

Krok 1. Utwórz step rejestrujący nowego użytkownika.

Krok 2. Utwórz step tworzący jedno zwierzę - ciało requesta ma sparametryzowane wartości **animalName**, **raceId**, **userUUID.** Parametr **userUUID** pobieramy z kontekstu TestSuite, natomiast parametry **animalName** i **raceId** będą określane w skrypcie groovy, który w pętli będzie czytać zawartość pliku CSV.

Krok 3. Utwórz step ze skryptem Groovy, który odczytuje plik CSV i w pętli przechodzi po wierszach pliku. Dla każdego wiersza - za wyjątkiem pierwszego który stanowi nagłówek danych – skrypt tworzy zwierzę powiązane z użytkownikiem utworzonym w kroku pierwszym.

Asercje

A. W pętli odczytaj zawartość pliku CSV i sprawdź czy istnieją dla każdego wiersza rekordy w bazie danych. Zastosuj połączenie do bazy danych i przygotuj zapytanie SQL, które wiąże użytkownika ze zwierzęciem, parametrami zapytania będą
    a. nazwa zwierzęcia
    b. Identyfikator rasy
    c. UUID zarejestrowanego użytkownika z kroku pierwszego

(czas: 45min)

Zadanie 10 - Wiele kroków

Utwórz nowy TestCase ***"Exercise 10 – Many steps",*** który wykona następujący scenariusz:

Krok 1. Zarejestruj nowego użytkownika.

Krok 2. Utwórz step tworzący jedno zwierzę - ciało requesta ma sparametryzowane wartości **animalName**, **raceId**, **userUUID.** Parametr **userUUID** pobieramy z kontekstu TestSuite, natomiast parametry **animalName** i **raceId** będą określane w skrypcie groovy, który w pętli będzie czytać zawartość pliku CSV.

Krok 3. Utwórz step ze skryptem Groovy, który odczytuje plik CSV i w pętli przechodzi po wierszach pliku. Dla każdego wiersza - za wyjątkiem pierwszego, który stanowi nagłówek danych – skrypt tworzy zwierzę powiązane z użytkownikiem utworzonym w kroku pierwszym.

Dodatkowo w pętli, po każdym wywołaniu requesta z utworzeniem zwierzęcia odczytaj w responsie uuid utworzonego zwierzęcia i dodaj uuid do listy, która przechowywana będzie w kontekście TestCase'a.

Krok 4. Wejdź w szczegóły każdego zwierzęcia i sprawdź czy ma oczekiwaną rasę - nie korzystaj ze sprawdzenia bazy danych lecz wykorzystaj restowe API.

Hint: Przygotuj step ***REST Request*** wyciągający szczegóły jednego zwierzęcia. Przygotuj step ***Groovy Script*****,** który w pętli wywoła step ***REST Request***. Pętla będzie dotyczyć listy uuid zwierząt utworzonych w kroku 3.

(czas: 45min)

POSTMAN – ZADANIA

Zadanie 4 - Pobierz listę ras

Utwórz folder o nazwie ***"Exercise 4 – Get all races"***, a w nim przygotuj request, który będzie pobierał listę wszystkich ras dostępnych pod zasobami "***dictionaries***"

Asercje:

A. Dodaj asercję sprawdzającą czy http code = 200
B. dodaj asercję sprawdzającą czy ilość elementów na liście w JSONie zwrotnym **jest równa 10**
C. dodaj asercję sprawdzającą czy na 5 pozycji nazwa kota to "Kot egzotyczny"

Hint: Stronka wspierająca w czytaniu JsonPath: https://jsonpath.com/

(czas: 15 min)

Zadanie 5 - Zarejestruj nowego użytkownika

Utwórz nowy folder o nazwie ***"Exercise 5 – Register new user"***, który zawierać będzie request wywołujący utworzenie nowego użytkownika.

Hint: Przykład zawartości body requestu tworzącego użytkownika odnajdziesz w dokumentacji Swagger na stronie http://51.38.129.181:8100/norad/swagger-ui.html

A) Zmień request w taki sposób, ażeby można było wywołać request utworzenia **wielokrotnie.**

Czyli e-mail nie może się powtarzać, za każdym wywołaniem powinien generować się nowy unikalny ze stałym prefixem - przyjmijmy Twoja pierwsza litera imienia i nazwisko. Na przykład: ***mszpiler***

Hint: Przeszukaj projekt ***postman-soapui-jmeter-qa-training*** w katalogu ***java-script*** znajdują się przykłady, które mogą Tobie pomóc rozwiązać zadanie.

B) Dodaj asercję sprawdzającą, że w odpowiedzi będzie w strukturze json w polu **email** istniała wartość oczekiwana wygenerowana przed wywołaniem requesta tworzącego nowego użytkownika.

C) Dodaj request sprawdzający istnienie rekordu użytkownika w bazie danych

- Napisz test do tego requesta, który potwierdzi istnienie encji użytkownika w backendowej bazie danych

(czas: 30min)

Zadanie 6 - Dodaj nowe zwierzę do nowego użytkownika

Utwórz nowy folder ***"Exercise 6 – Add new animal for new user"***, który zawierać będzie następujące kroki:

Krok 1. Rejestracja nowego użytkownika.

Krok 2. Pobierz wybrany losowo element z listy ras i zastosuj go przy tworzeniu zwierzęcia. Zapisz go w zmiennych środowiskowych pod nazwą **RACE_ID**

Krok 3. Dodaj nowe zwierzę dla użytkownika z kroku 1, dodane zwierzę powinno zawierać 3 zdjęcia oraz rasę ustaloną w kroku 2.

Dodatkowo, przed wywołaniem tworzenia nowego zwierzęcia przygotuj pre-script, który wygeneruje losową nazwę zwierzęcia ze stałym prefixem na przykład pierwsza litera imienia i nazwisko np. **mszpiler .**

Prefix pozwoli lepiej identyfikować Twoje dane w bazie danych, która jest wspólna dla wszystkich uczestników szkolenia.

Zapisz uuid zwierzęcia w zmiennych środowiskowych pod nazwą **ANIMAL_UUID**.

Asercje w kroku 3

A. Sprawdź czy w odpowiedzi lista zdjęć jest oczekiwana i wynosi 3
B. Sprawdź czy każde ze zdjęć posiada niepuste pole uuid

Krok 4. Utwórz request pobierający szczegóły zwierzęcia.

Asercje w kroku 4

A. Sprawdź czy rasa w odpowiedzi jest taka sama jak rasa wylosowana w kroku 2.

Krok 5. Utwórz request pobierający szczegóły użytkownika.

Asercje w kroku 5

A. Sprawdź czy zwierzę powiązane jest z użytkownikiem utworzonym w kroku 1.

(czas: 45min)

Zadanie 7 – Przetestuj upload zdjęcia - załącznik wskazany ręcznie

W kolekcji ***"Training Collection"*** utwórz folder ***"Exercise 7 – Upload file"***, który będzie wykonywał test uploadu wybranego contentu zdjęcia na serwer. Wykorzystaj encję zdjęcia utworzonego w zadaniu 6.

Krok 1. Popraw ***"Step 3 – Create new animal"*** w zadaniu ***"Exercise 6 – Add new animal for new user"*** w taki sposób ażeby z responsa zapisać w zmiennych środowiskowych UUID pierwszego zdjęcia. Zapisz tą wartość pod zmienną **FIRST_IMAGE_UUID**

Krok 2. Dodaj request o nazwie ***"Upload file"***, który wywołuje endpoint odpowiedzialny za upload plików na serwer.

Krok 3. Dodaj załącznik ręcznie i wykonaj request, który wykona upload obrazka na serwer.

Asercje

A. Sprawdź czy odpowiedź http = 200

(czas: 20min)

Zadanie 8 - Dodaj zwierzęta pobierając dane z pliku CSV

W kolekcji ***"Training Collection"*** utwórz folder ***"Exercise 8 – Add many animals from CSV file"***, który utworzy wiele zwierząt dla konkretnego użytkownika.

Krok 1. W folderze ***"Exercise 8 – Add many animals from CSV file"***, utwórz 2 dodatkowe foldery*:*

- ***Register new user*** – tutaj będzie request do utworzenia nowego użytkownika
- ***Create many animals -*** tutaj będzie request do utworzenia zwierzęcia, folder zostanie użyty do wywołania Runnera z listą zwierząt znajdujących się wpliku CSV.

Krok 2. Utwórz step rejestrujący nowego użytkownika i zapisujący UUID użytkownika pod zmienną **EX8_USER_UUID**.

W kroku tym również podobnie jak w zadaniu 5, email użytkownika powinien być generowany losowo.

Krok 3. W folderze ***"Create many animals"*** utwórz request tworzący jedno zwierzę - ciało requesta ma sparametryzowane wartości **name**, **raceId**, **userUUID.** Parametr **userUUID** pobieramy ze zmiennej **EX8_USER_UUID**, natomiast parametry **name** i **raceId** będą pobrane z pliku CSV w trakcie wywołania testu za pomocą Runnera.

Krok 4. Utwórz asercję sprawdzającą poprawne wykonanie kroku 3 – http code = 200.

Lista zwierząt znajduje się w pliku CSV. Dane na temat zwierzęcia w pliku, to nazwa zwierzęcia oraz rasa.

Krok 4. Wywołaj request z folderu ***"Create many animals"*** za pomocą ***Runnera***. W Runnerze podaj środowisko oraz wskaż plik CSV z danymi do testów. Przeanalizuj wyniki testów.

(czas: 45min)

JMETER - ZADANIA