**Write-Up**

**GitHub:** <https://github.com/olomansaas/Non-Functional-Testing-_Phase-3.git>

**PostMan Assignments:**

**1. CreatePetID**

Method: POST

URL: https://petstore.swagger.io/v2/pet.

Tests: Checks if the status code is 200 and if the response body includes the string "available".

**2. GetPetID**

Method: GET

URL: https://petstore.swagger.io/v2/pet/{{petID}}

Purpose: Retrieves pet information based on a given pet ID From the Provided CsvFile.

Tests: Validates the status code is 200 and the response includes "available".

**3. DeletePet**

Method: DELETE

URL: https://petstore.swagger.io/v2/pet/{{petID}}

Purpose: Deletes a pet from the pet store based on the given ID From the Provided CsvFile.

Tests: Checks for a 200 status code and if the response includes "unknown".

**4. Assignment002**

Method: PUT

URL: Derived from a variable {{testUrl}}

Purpose: Test Url is Defined on Global Level as Prescribed in the question.

Tests: Check for the environment Provided As the output.

**5. Assignment003**

Method: GET

URL: https://petstore.swagger.io/v2/user/{{UserName}}

Purpose: Retrieves user information based on a username.

Tests: Validates the status code, username, email, and user status from the response.

**6. Assignment004**

Method: GET

URL: https://petstore.swagger.io/v2/pet/findByStatus?status= Sold

Purpose: Finds pets by status.

Tests: Checks if the status code is 200 and if all pets in the response have the status "available",”Sold”,”Pending”.

**7. Assignment005**

Method: GET

URL: https://petstore.swagger.io/v2/user/logout

Purpose: Logs out a user.

Tests: Validates the status code and message in the response.

**Conclusion**

1.The collection focuses on testing various CRUD (Create, Read, Update, Delete) operations related to pets and user management in the Swagger Pet Store API.

2. Each request is accompanied by tests to validate the response, ensuring the API behaves as expected.

3.This collection is a useful tool for automated testing of the Pet Store API, ensuring that its endpoints function correctly and return expected results.

**RestAssured Assignments:**

**#001**

RestAssure\_001, is designed to test a REST API using REST Assured, a Java library for testing RESTful web services. The class contains three test methods, each annotated with @Test, for performing POST, GET, and DELETE requests to a pet store API.

**Class:** RestAssure\_0011

**Purpose:** To perform API testing on a pet store service.

**Library Used:** REST Assured for API testing.

**Dependencies and Imports**

java.io.File: For file handling.

org.apache.logging.log4j: For logging.

org.hamcrest.Matchers: For assertions.

org.testng.annotations.Test: For defining test methods.

Logger logger: An instance of Logger for logging information and traces.

**Test Methods**

Assignment001Post()

**Priority:** 1

**Purpose**: Tests the POST request functionality of the pet store API.

**Process**:

Logs the start of the test.

Reads a JSON file containing the request body.

Sends a POST request to the API.

Asserts the status code and response body.

Logs the status code validation and ID extraction.

assignment001Get()

**Priority:** 2

Depends On: Assignment001Post

**Purpose:** Tests the GET request functionality of the pet store API.

**Process**:

Logs the start of the test.

Sends a GET request to retrieve the pet information.

Asserts the status code and response body.

Logs the retrieval and validation of the pet ID and status.

assignment001Delete()

**Priority:** 3

Depends On: assignment001Get

Purpose: Tests the DELETE request functionality of the pet store API.

Process:Logs the start of the test.

Sends a DELETE request to remove the pet information.

Asserts the status code and response body.

Logs the completion of the delete request.

**#002**

Test Class: RestAssured\_002

**Test Methods:**

**Method:** assignment002Post

Test Priority: 1

**Description**: This method sends a POST request to the pet store API to add a new pet using data from a JSON file.

**Steps:**

Reads a JSON file containing pet data.

Sends a POST request to the API with the JSON data.

Validates the response status code, logs the response, and extracts the pet's ID.

**Dependencies**: None

Logger Usage: Logs information about the request, status code validation, and ID extraction.

**Method:** assignment002Put

Test Priority: 2

**Description:** This method sends a PUT request to update information about a pet using data from a JSON file.

**Steps**:

Reads a JSON file containing updated pet data.

Sends a PUT request to the API with the JSON data.

Validates the response status code, logs the response, and extracts the updated pet's ID.

**Dependencies**: Depends on the successful execution of assignment002Post.

Logger Usage: Logs information about the request, status code validation, and ID extraction.

**REST Assured Usage**:

Base URI: The base URI for all requests is set to "[https://petstore.swagger.io/v2/pet](https://petstore.swagger.io/v2/pet" \t "https://chat.openai.com/c/_new)".

**POST Request:**

The POST request sends JSON data from a file to add a new pet.

The response is validated for a status code of 200, and the pet's ID is extracted for further use.

**PUT Request:**

The PUT request updates information about a pet using JSON data.

The response is validated for a status code of 200, and the updated pet's ID is extracted for further use.

**5. Conclusion:**

This test suite expands on the previous one by including a PUT request to update pet information. It demonstrates the use of REST Assured for testing CRUD operations on a pet store API. The logging mechanism enhances the test suite's maintainability and debugging capabilities.

**#003-4**

In this Java code comprises two methods, **assignment003User** and **assignment004login**, designed for testing user-related functionalities in a pet store API using REST Assured.

**1. Method: assignment003User**

**Purpose**:

This method aims to verify the details of a user with the username "Uname001" in the pet store system.

**Steps:**

Sends a GET request to retrieve user information.

Validates the response status code (expecting 200), logs the entire response, and performs body assertions.

Body assertions check that the retrieved user's username is "Uname001," email is "Positive@Attitude.com," and user status is 1.

**Usage:**

This method is useful for ensuring the correctness of user information retrieval from the API.

**Method: assignment004login**

**Purpose:**

This method is designed to simulate a user login operation by sending a GET request to the pet store API's login endpoint.

**Steps:**

Sends a GET request to the login endpoint with preemptive basic authentication using the username "Uname001" and password "@tt!tude."

Validates the response status code (expecting 200), logs the entire response, and performs a body assertion.

Body assertion checks that the response body contains a message (using Matchers.anything()).

**Usage:**

This method is essential for verifying the login functionality of the pet store API.

**#005-6**

**Method: assignment005FindByStatus**

**Purpose:**

This method is designed to find pets based on their status in the pet store system.

**Steps:**

Sends a GET request to the "findByStatus" endpoint with the query parameter "status" set to "sold."

Validates the response status code (expecting 200), logs the entire response.

**Usage:**

This method is useful for testing the functionality of retrieving pets based on their status.

**Method: assignment006Logout**

**Purpose:**

This method simulates a user logout operation by sending a GET request to the pet store API's logout endpoint.

**Steps:**

Sends a GET request to the logout endpoint.

Validates the response status code (expecting 200), logs the entire response, and performs body assertions.

Body assertions check specific fields in the response body, including "code," "type," and "message."

**Usage:**

This method is essential for verifying the correctness of the user logout mechanism

**J-Meter**

**Procedure/Methods:**

Add Listeners to collect and analyze the test results.

Common listeners include in this project :

**View Result Tree:**

A JMeter listener that displays detailed information about each sample, including request and response data, allowing in-depth inspection of individual HTTP transactions.

**Summary Report**:

A JMeter listener that provides a summary of key performance metrics such as average response time, throughput, and error rates for all executed samplers in the test.

**Aggregate Report**:

A JMeter listener that presents aggregate performance statistics, including average response time, throughput, and error rates, for each sampler in a tabular format.

**Graph Result**:

A JMeter listener that visualizes performance metrics over time, allowing users to analyze trends and patterns in response times, throughput, and other key indicators through graphical representations.

**Assertion Result**:

A JMeter listener that reports the results of assertions applied during a test, indicating whether each assertion passed or failed for individual samples, helping identify deviations from expected behavior.

**Conclusion:** The performance testing module provides a comprehensive framework for assessing the performance of a web portal using JMeter 5.1.1 and JDK 8. Regular execution of performance tests and analysis of results will ensure the continuous improvement of the web portal's performance.