API Test Automation
Foundation Level Hands-On Exercises
Using REST Assured

About:

1. These hands-on exercises will enable you to achieve Foundation level capability on REST API automation, using REST Assured.
2. As a guidance, the complexity and duration that you would typically take to complete each exercise is mentioned.
3. You may use IBM Learning, online medium, or SME interaction to understand how to technically solve these exercises.
4. You must successfully execute the REST Assured automated scripts prior to marking the exercises as complete.

Installation and Set Up

Download and install the latest open-source versions of the following software:

Download Java Development Kit (JDK) and set up Java Environment variables under System variables	JDK download link: https://www.oracle.com/in/java/technologies/javase-downloads.html Define Environment variables: i. JAVA_HOME = C:\Program Files\Java\jdk-16.0.1 ii. Add path 'C:\Program Files\Java\jdk-16.0.1\bin' within the Path parameter under the System variables Note: Download the latest JDK version, based on which the Environment variables values may vary from the values mentioned below.
Eclipse	https://www.eclipse.org/downloads/
Download and import REST Assured JARs in Java Project	https://github.com/rest-assured/rest-assured/wiki/Downloads Note: Download all the zip files under the **Current Direct Downloads** section. Extract content from the zip files and then configure/Import REST Assured JARs under the Project within Eclipse IDE (as demonstrated below): i. Create a Java Project within Eclipse IDE. ii. Right-click the Java Project created within Eclipse IDE, select **Properties**, and then select **Add External JARs** under **Java Build Path**. iii. Import ALL the files with type **Executable Jar File** (indicated with a Java icon) from all the extracted folders (Note: Importing all JAR files may not be required though it is recommended that you import all JAR files before you start REST API Automation using REST Assured.).

Background Information:

For practitioners to **effectively understand REST API automation using REST Assured**, it is extremely important to gain meaningful experience interacting with **real-time APIs**. The following instructions demonstrate how you can CLONE the **Best Buy API Playground** to your local machine. The advantage of Best Buy API Playground is that it supports full CRUD (Create Read Update Delete) operations for all API endpoints and does not require any external services or databases.

Instructions for cloning Best Buy API Playground to your local machine:

1. Make sure you have NodeJS installed (version 4 or newer) on your system. If NodeJS isn't installed, then install it from: https://nodejs.org/en/
2. Create a folder on your local machine and name it **Best Buy APIs**. (Note: You can create the folder anywhere on your local machine, for example, under **Documents**.)
3. Open command prompt and navigate to the Best Buy APIs folder. For example, if you have created the folder under **Documents**, then type:
   `C:\Users\SACHINJIBHAKATE\Documents\Best Buy APIs`
4. Clone the project under the Best Buy APIs directory on your local machine using the command **`git clone`** **`https://github.com/bestbuy/api-playground/`**
5. Navigate to the API-playground using the command: **`cd api-playground`**
6. Install NPM (if it isn't already installed on your system) using the command: **`npm install`**
7. Then start the server using the command: **`npm start`**
8. Access the Best Buy APIs (via a browser) hosted on your local machine using **`http://localhost:3030`**

Use Case 1 - Create a New Service

Details: Use a single API request to create a new service through the POST REST API method. The objective of this exercise is to create a new service within Best Buy API Playground using the REST Assured automation script and validate the response code.

Transaction complexity	Automation complexity	Recommended duration
Low	Low	1 hour

The following MUST be considered in the automation script:

1. To create a new service using the POST REST API method, provide the following parameter in the POST REST API method:
   - Provide the service name in the POST request body.
2. Refer Swagger Best Buy API Playground (http://localhost:3030/docs/) **POST /services** section under **Services** for endpoint and body schema format.
3. To ensure the request is successful, extract the response code from the json response body and use REST Assured assertion to ensure the response code from the json response body matches the response code provided in the schema.
4. Extract the Json response body in a variable.
5. Parse the Json response using the JsonPath class.
6. Extract the ID (generated for the newly created service) from the json response body and store it in a variable.

Use Case 2 - Create a new service, update it with a new name and retrieve service details to validate if the new service name is present in the json response body

Details: Build an end-to-end automation script with REST Assured using GET, POST and PUT http methods, thus integrating multiple APIs with Jason response values. To build the end-to-end automation script, create a new service using the POST REST API method, update the service with a new name using the PUT REST API method, and retrieve the service details to validate if the new service name is present in the json response body, using the GET REST API method.

Transaction complexity	Automation complexity	Recommended duration
Medium	Medium	3 hours

The following MUST be considered in the automation script (follow the instructions in sequence under Parts 1, 2, 3, and 4):

Part 1:

1. Create a new service using the POST REST API method. Provide the following parameter in the POST REST API method:
   - Provide the service name in the POST request body
2. Refer Swagger Best Buy API Playground (http://localhost:3030/docs/) **POST /services** section under **Services** for endpoint and body schema format.
3. To ensure the request is successful, extract the response code from the json response body and use the REST Assured assertion to ensure the response code retrieved from the response body matches the response code provided at Best Buy API Playground.
4. Extract the Json response body in a variable.
5. Parse the Json response using the JsonPath class.
6. Extract the ID (generated for the newly created service) from the json response body and store it in a variable.

Part 2:

1. Update the service with a new name using the PUT REST API method. Provide following parameters in the PUT REST API method:
   a. Provide the variable name (where the ID generated for the newly created service in Part 1 is stored) in the PUT request body.
   b. Store the new desired service name in a variable and provide the variable name in the PUT request body.
2. Refer Swagger Best Buy API Playground (http://localhost:3030/docs/) **PUT /services** section under **Services** for endpoint and body schema format (skeleton)
3. To ensure request is successful, extract the response code from the json response body and use the REST Assured assertion to ensure the response code retrieved from the response body matches the response code provided at Best Buy API Playground.

Part 3:

Retrieve the Service details to validate if the new service name is present in the json response body using the GET REST API method.

1. Retrieve the Service (created in Part 1) using GET REST API method. Provide following parameter in the GET REST API method:
   a. Provide the variable name (where the ID generated for the newly created service in Part 1 is stored) in the GET request body.
2. Refer Swagger Best Buy API Playground (http://localhost:3030/docs/) **GET /services** section under **Services** for endpoint details.
3. To ensure request is successful, extract the response code from the json response body and use the REST Assured assertion to ensure the response code retrieved from the response body matches the response code provided at Best Buy API Playground.
4. Extract the json response body in a variable.
5. Parse the Json response using the JsonPath class.
6. Extract the name from the **Name** field in the json response body and store it in a variable.

Pre-requisite for Part 4: Download the TestNG JAR file from Maven Repository (URL: https://mvnrepository.com/) and import the TestNG JAR file within the project in Eclipse IDE.

PART 4:

1. Add a validation using TestNG Assertion to ensure the service name retrieved in the json response body (GET method) matches the service name stored in the name variable.

Use Case 3 - Retrieve a count of all the services and print IDs and the names of all the services

Details: Parse the nested Json response body having Array (Nested Json Parse) and retrieve a count of all the services.

Transaction complexity	Automation complexity	Recommended duration
Medium	Low	1 hour

The following MUST be considered in the automation script:

1. Retrieve all Services using the GET REST API method.
2. Refer Swagger Best Buy API Playground (http://localhost:3030/docs/) **GET /services** section under **Services** for endpoint details.
3. To ensure the request is successful, extract the response code from the json response body and use the REST Assured assertion to ensure the response code retrieved from the response body matches the response code provided at Best Buy API Playground.
   Note: Use JSON Editor to view json response body in the node format.
4. Extract the Json response body in a variable.
5. Parse the Json response using the JsonPath class.
6. Extract a count of all the services (use the Data array) and store it in a variable.
7. Extract service IDs and names for all the services (Hint: Iterate through services, that is, elements within the array using Array Indexing). Print service IDs and names for all the services in the console.

Use Case 4 - Basic framework implementation using Cucumber BDD, Maven, JUnit for REST Assured API Automation for creating a new service

Details: Create a basic automation framework using Cucumber BDD, Maven, JUnit for REST Assured API Automation and execute the creation of a new service test case (through the Create Service API) using the automation framework.
Note: It is imperative that Foundational-level practitioners are equipped with this basic framework knowledge, which will enable them to work on real time projects. Advanced framework topics will be covered in the exercises for Experienced-level practitioners.

Transaction complexity	Automation complexity	Recommended duration
Low	Medium	2 hours

The following MUST be considered in the automation script:

Pre-requisites:

- Install Maven - bin.zip file (https://maven.apache.org/download.cgi)
- Define the Environment variable for Maven, that is, Configure 'MAVEN_HOME' in System variables. MAVEN_HOME = C:\work\apache-maven-3.8.1-bin\apache-maven-3.8.1 (if your maven is located at C:\work).

1. Create Maven Project within Eclipse IDE.
2. Define Cucumber, REST Assured, and Cucumber Junit dependencies within the pom.xml file (Cucumber, REST Assured, and Cucumber Junit dependencies are available within the Maven Repository. Maven Repository url: https://mvnrepository.com/).
3. Create a package for feature files.
   - Create a Feature File within the Feature Files package for the Create Service API
   - Define content within the Feature File using 'Feature', 'Scenario', 'Given, 'When', and 'Then' keywords for the Add Place API.
4. Create a package for the Step Definition files.

- Create a Step Definition file within the Step Definition files package for the Create Service API.
- Define content (implementation for 'Create Service API' using '@Given, '@When', and '@Then' keywords) within the definition file.
  Note: Use the POST REST API method details from Exercise 1 (provide a different service name in the json request body).

5. Create a package for the JUnit Test Runner file.
   - Create a JUnit Test Runner file within the JUnit Test Runner file package and define the Feature File and Step Definition file path under **CucumberOptions**.
6. Run the JUnit Test Runner file as 'Junit Test'. Ensure the run is successful with the desired response, and that the service is created successfully.