Testing – API Automation Framework Guidelines

Enterprise Technology Group

Version – 1.0

Revision History

Date	Version	Comments	Created By
July 12th 2022	1.0	Initial	Sreenivasulu Boyapati

1. Introduction

This document provides details about QA API automation framework and the complete flow of project structure. In addition, it contains all the naming conventions used in a project, Script writing best practices and some other additional references.

All QA automation testers must follow the guidelines defined in this document to avoid any deviations from the standard processes.

2. Naming conventions

2.1 Project name

Project name must follow given naming convention.

Syntax: OrgName.Automation.AutomationType.ProjectName

Example: Mirra.Automation.API.CA

2.2 Feature files

Feature file names must begin with upper case letters and the remaining words (if Applicable) starting letters should have even upper-case letters followed by ".feature".

Example: GetApi.feature

2.3 Methods

Method names must begin with lower case letters and the remaining words starting letters should have upper-case letters i.e., **Camel Casing**.

Example: randomString()

2.4 Variables

Follow same as "Methods naming convention".

Examples: generatedString

2.5 Properties file variables

Follow same as "Methods naming convention".

Examples: baseUrl, tokenId

2.6 Classes

Class names must begin with upper case letters and the remaining words starting letters should have even upper-case letters i.e., **Pascal Casing**.

Examples: JavaUtils

3. Script writing best practices

3.1 API response comparison

While working on GET methods the entire API response can be compared with target response if both the data sets are mirror.

3.2 Java methods

Testers can create custom defined java methods to work with complex scenarios in karate.
Note – Refer framework template "JavaUtils" class and "RandomString" feature for better understanding

3.3 Assertions

- Karate offers different types of assertions to validate responses. Refer Karate's official web site for all types of assertions.
  https://github.com/karatelabs/karate
- Use the appropriate assertion type based on requirement

3.4 JSON request body (request payload)

- **DO NOT** hard code the request body in feature files itself. Always store the data in a "Json" file and pass it from "Payloads" section.
- Generate test data dynamically (Wherever possible) using **randomString()** method from "Commons.feature" file or you can even use **randomString()** from "JavaUtils" class

4. Technologies

Editor: IntelliJ IDEA 2021.3.3 (Community edition)
Programming language: Java JDK-18
Framework: Karate BDD with Junit5 (version - 1.2.0)
Design Pattern: Data Driven Testing
Project Type: Maven (Apache Maven 3.8.5 version)
Loggers: logback classic & core
Reports: HTML & JSON

5. Framework demo

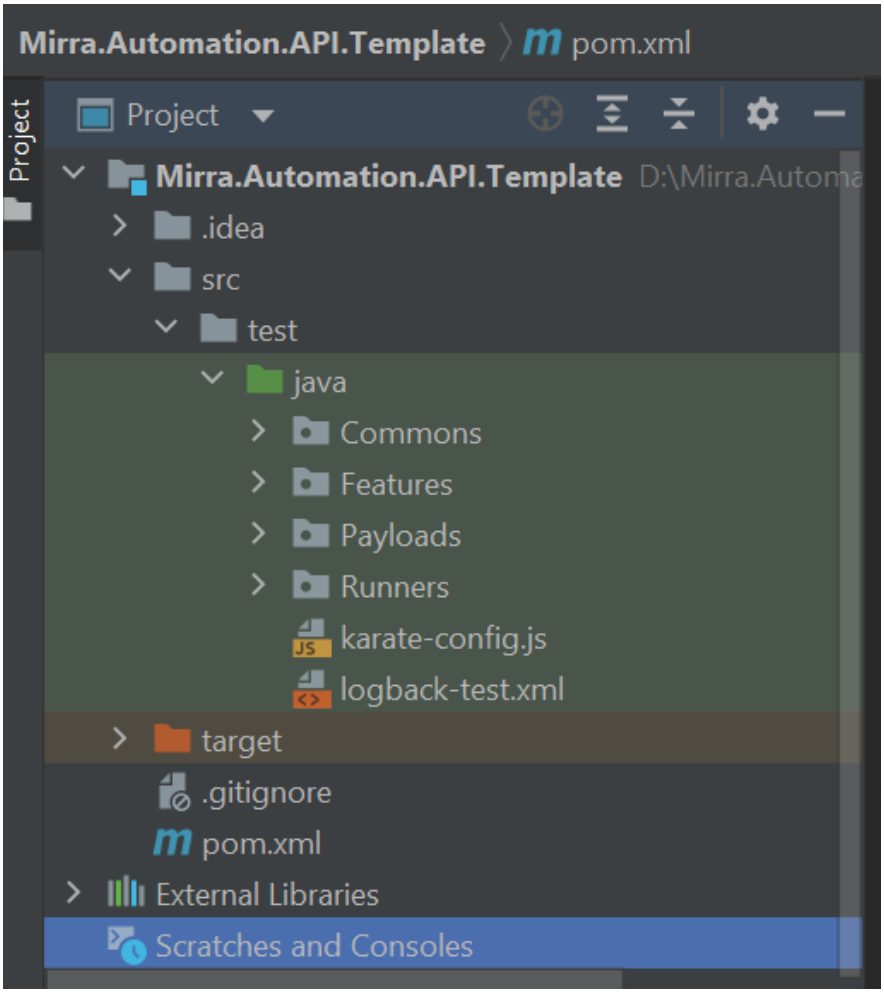
Refer given link for framework demonstration.

API Automation framework (Karate) demonstration

6. Project structure

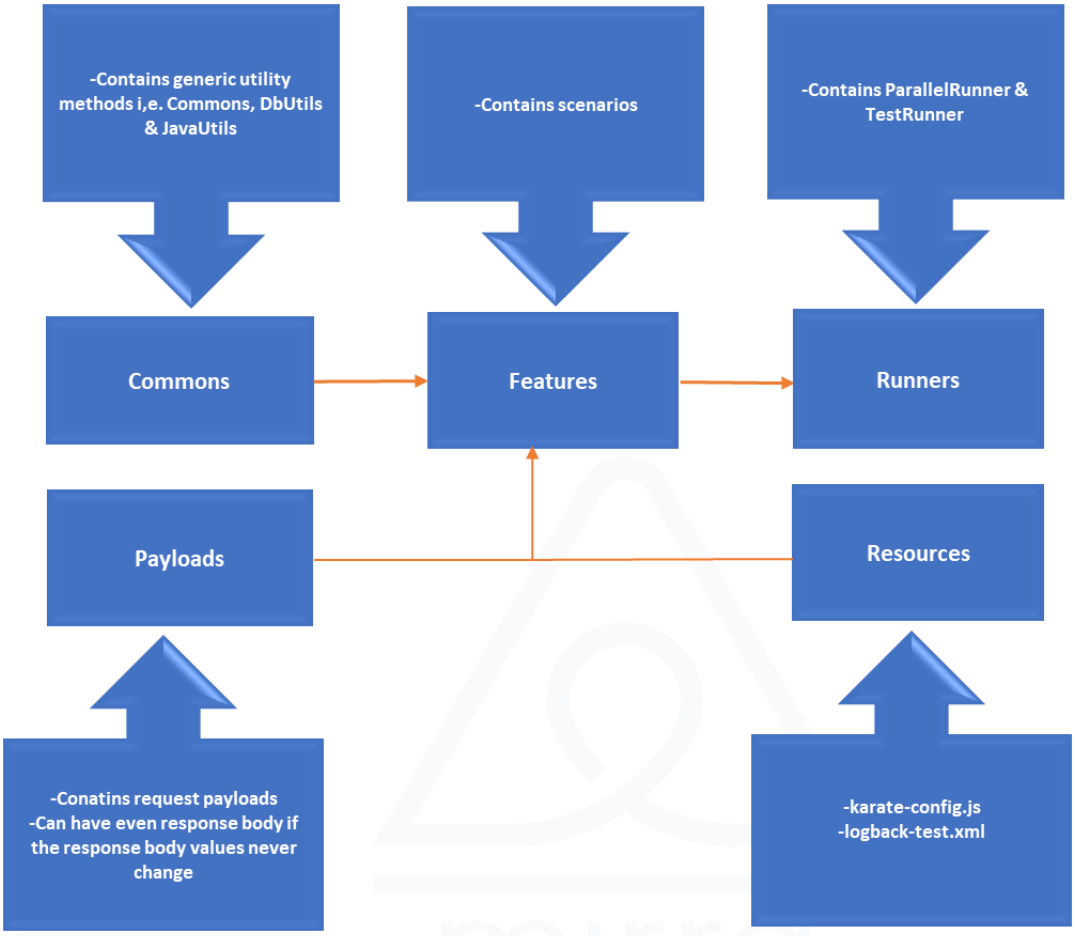
6.1 Sample project structure

Refer given snapshot to understand the overall folder structure.



6.2 Project structure

Refer given snapshot to understand how the files or classes are linked with each other from different folders.



7. Folder or Package level details

Go through the instructions provided in this section to understand project structure in detail.

7.1 POM.XML

- Contains list of dependencies & test resources path
- **Do not update this file. Must check with lead/Framework SPOC in case of any changes needed.**

7.2 Features

- Create feature files and add applicable scenarios

7.3 Commons

7.3.1 Commons.feature

- Contains generic java script methods
- **Do not update this file. Must check with lead/Framework SPOC in case of any changes needed**

7.3.2 DbUtils

- Contains generic methods related to database activities
- Use "DbUtils" methods under scenarios (If applicable) to get data from database in json format.
  - The query can be a table query or even it can be a stored procedure as well
- In case of multiple of DB connections, you may need to have multiple URLs representing each with one specific database
- **Do not update this class. Must check with lead/Framework SPOC in case of any changes needed**

7.3.3 JavaUtils

- Contains java generic methods
- Use "JavaUtils" methods under scenarios based on the requirements
- **Do not update this class. Must check with lead/Framework SPOC in case of any changes needed**

7.4 Payloads

- Add request payloads for POST, PUT and DELETE methods in ".json" format
- Store response body in ".json" format for requests where it's needed
- Payload files can be read under the scenarios using "**read**" keyword along with the class path

8. Resources

- Contains karate-config.js (properties) and logback-test.xml files
- Contains all the standard variables i.e., username, password, baseUrl, different environments, tokenId & timeouts etc.
- List of variables defined in "karate-config.js" file can be accessed anywhere in a project under features or scenarios

9. Logs

- Logs will be generated under "target" folder of the project main directory
- Refer "karate.log" file for log messages

10. Reports

- Reports will be generated in "html & JSON " format
- Refer "Target/karate-summary.html" location to view reports

11. Runners

- Use TestRunner to run test cases in sequential mode
- Test suits can be handled under the "tags" tab
- Use "ParallelRunner" to run test cases parallelly

12. Test cases execution

- Use test runner to run test cases in sequential mode
- Use "command prompt" to run test cases parallelly

  A few examples to run test cases from command prompt

  mvn clean test -Dkarate.env='uat'

  mvn test -DargLine="-Dkarate.env=uat"

  mvn clean test -Dtest=ParallelRunner

  mvn clean test -Dtest=ParallelRunner -Dkarate.env='uat'

13. Additional references

- **Karate official documentation:**
  https://github.com/karatelabs/karate
- **Karate sessions from internet:**
  https://www.youtube.com/watch?v=MUgG_n0WuvM&list=PLMd2VtYMV0OQkXQ5BrHlZwoTqp17ACsnG
  https://www.youtube.com/watch?v=xzq6JJZ0Oj8&list=PLFGoYjJG_fqpUgFYoKIMZJAblUbGHSQAb