**Karate Installation and Testing**

Important reference link: <https://github.com/karatelabs/karate#5>

This link is an important link : <https://www.manual2automation.com/>

Important keywords:

* DSL: Domain-Specific Language (ex: XML, JSON)
* Payload Assertion: The request payload contains a user object that contains the data used to fill the form

Diagram

Description automatically generated

What is Glue in Cucumber?

Glue Option helps Cucumber to locate the step definition file. We specify the package to load glue code (step definitions or hooks) in the glue option. When no glue is provided, Cucumber will use the package of the annotated class.

Create a Spring boot web project and add following dependencies in pom file

A picture containing text

Description automatically generated

Timeline

Description automatically generated

Folder structure is important in Test.

Graphical user interface

Description automatically generated with low confidence

Create TestRunner class

Graphical user interface, text, application, email

Description automatically generated

Create a feature file as sample.feature , it should be in the features folder

A picture containing graphical user interface

Description automatically generated

Then run the TestRunner class.

Later TestRunner put inside separate package to organize the file structure (Tests and features separately)

Graphical user interface, text, application

Description automatically generated

After doing that make sure to change the classpath of testSample()

Graphical user interface, text, application

Description automatically generated

You can run tests separately by selecting test function -> right click ->run test case

For that import following plugins (maven compiler plugin and configuration source,target)

A picture containing text

Description automatically generated

For Testing use this site: <https://reqres.in/>

Use <https://tools.knowledgewalls.com/> to convert json to single line

userDeatils.feature file code

Graphical user interface, text, application, email

Description automatically generated

Create json file to store results

Graphical user interface, text, application

Description automatically generated

Make sure to give”../ ” when going out of the package

Graphical user interface, text, application

Description automatically generated

Ex: Currently I am in TestRunner

@Karate.Test  
Karate testFull() {  
 **return** Karate.*run*(**"../features/sample2.feature"**).relativeTo(getClass());  
}

How to maintain one output.json 🡪 put the json results in an array

Graphical user interface, text, application

Description automatically generated

Then change the feature file as following

Graphical user interface, text, application, email

Description automatically generated

**You can segregate the feature files as following**

* userDetails1.feature
* userDetails2.feature

You can call feature file from another feature file and get/use the data of the other feature file as following. (ex: result variable takes scenario 2 data)

Graphical user interface, text, application

Description automatically generated

Junit4---------------------------------------------

Important links:

<https://jsonpathfinder.com/>

pom file

*<?***xml version="1.0" encoding="UTF-8"***?>*<**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
 <**parent**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-parent</**artifactId**>  
 <**version**>2.7.5</**version**>  
 <**relativePath**/> *<!-- lookup parent from repository -->* </**parent**>  
 <**groupId**>com.ab</**groupId**>  
 <**artifactId**>testing-project</**artifactId**>  
 <**version**>0.0.1-SNAPSHOT</**version**>  
 <**name**>testing-project</**name**>  
 <**description**>Demo project for Spring Boot</**description**>  
 <**properties**>  
 <**java.version**>1.8</**java.version**>  
 </**properties**>  
 <**dependencies**>  
 <**dependency**>  
 <**groupId**>com.intuit.karate</**groupId**>  
 <**artifactId**>karate-core</**artifactId**>  
 <**version**>1.3.0</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.intuit.karate</**groupId**>  
 <**artifactId**>karate-apache</**artifactId**>  
 <**version**>0.9.6</**version**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>com.intuit.karate</**groupId**>  
 <**artifactId**>karate-junit4</**artifactId**>  
 <**version**>1.3.0</**version**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter</**artifactId**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-test</**artifactId**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 </**dependencies**>  
  
 <**build**>  
 <**testResources**>  
 <**testResource**>  
 <**directory**>src/test/java</**directory**>  
 <**excludes**>  
 <**exclude**>\*\*/\*.java</**exclude**>  
 </**excludes**>  
 </**testResource**>  
 </**testResources**>  
 <**plugins**>  
 <**plugin**>  
 <**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-maven-plugin</**artifactId**>  
 </**plugin**>  
 </**plugins**>  
 </**build**>  
  
</**project**>

File Structure

Graphical user interface, application

Description automatically generated