



Karate REST

Tests de non régression d'API REST
Geoffrey, le 18 juin 2019

© Ubik Ingénierie 2019

Sommaire

- Tests de non régression
- Pourquoi ?
- Stack technique
- Fonctionnalités
- Démo
- Questions / Réponses

Qu'est-ce qu'un test de non régression ?

Test automatisé qui permet de tester une fonctionnalité afin de vérifier qu'elle soit toujours opérationnelle.

Cela permet de détecter les régressions plus rapidement et de les corriger.

Pourquoi Karate ?

- Test de non régression d'API REST basé sur Cucumber et Gherkin
- Pas besoin de client lourd (contrairement à Postman)
- Conçu pour les tests d'API
- Tests des résultats avec des syntaxes Javascript
- Open source / gratuit

Stack technique

cucumber 

maven



Comment écrire un test avec Gherkin ?

Prérequis

* ...

Entrants :

Given ...

And ...

Actions :

When ...

And ...

Vérifications :

Then ...

And ...

Karate prédéfinit des étapes pour la manipulation d'API rest



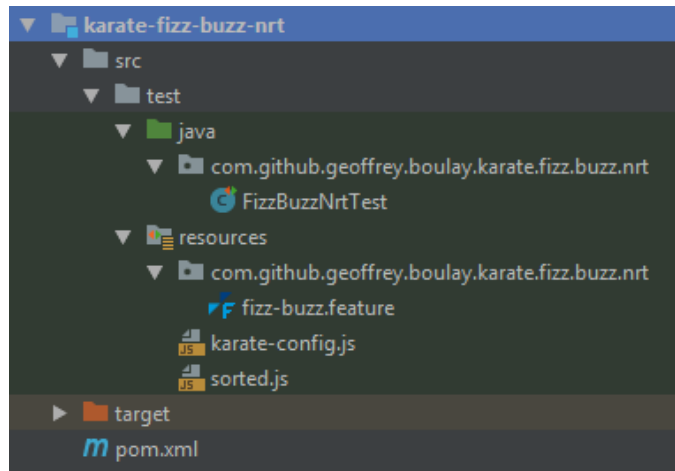
Pas besoin de définir d'étapes avec cucumber

```
Scenario: interval fizz buzz : 1 to 100
* def sorted = read('classpath:sorted.js')
* def byOne = function(one,two) {return one.input == two.input - 1;}
Given path "/fizzbuzz/standard/interval"
And param from = "1"
And param to = "100"
When method GET
Then status 200
And match response == '#[ === 100] '
And match each response contains
"""
  {
    input: "#number? __ >= 1 && __ <= 100",
    result: "#string"
  }
"""
And assert response[0].input === 1
And assert sorted(response, byOne)
```

Comment lancer karate ?

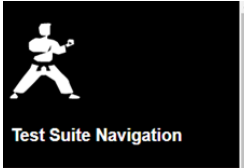
Via maven (ou Gradle) :

> mvn test -Dparam1=value1 -Dparam2=value2



```
<dependency>
  <groupId>com.intuit.karate</groupId>
  <artifactId>karate-apache</artifactId>
  <version>0.8.0</version>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>com.intuit.karate</groupId>
  <artifactId>karate-junit4</artifactId>
  <version>0.8.0</version>
  <scope>test</scope>
</dependency>
```


Résultats des tests



Test Suite Navigation

of failed tests: 0/85
(0.00%)

of skipped tests: 0/85
(0.00%)

of passed tests: 85/85
(100.00%)

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36
37	38	39	40	41	42
43	44	45	46	47	48
49	50	51	52	53	54
55	56	57	58	59	60
61	62	63	64	65	66
67	68	69	70	71	72
73	74	75	76	77	78

Scenario Outline: unit fizz buzz : 1 must return 1 (1)	
Test 1 : * configure connectTimeout = 3000	0.504377
Test 2 : * configure readTimeout = 3000	0.000563
Test 3 : * url serverUrl	0.000082
Test 4 : Given path "/fizzbuzz/standard/unit"	0.004522
Test 5 : And param input = "1"	0.002946
Test 6 : When method GET	0.345531
Test 7 : Then status 200	0.000386
Test 8 : And assert response.input === 1	0.017968
Test 9 : And assert response.result === "1"	0.004247
Scenario Outline: unit fizz buzz : 2 must return 2 (2)	
Test 10 : * configure connectTimeout = 3000	0.006348
Test 11 : * configure readTimeout = 3000	0.000201
Test 12 : * url serverUrl	0.000037
Test 13 : Given path "/fizzbuzz/standard/unit"	0.000117
Test 14 : And param input = "2"	0.002757
Test 15 : When method GET	0.005892
Test 16 : Then status 200	0.000068
Test 17 : And assert response.input === 2	0.003348
Test 18 : And assert response.result === "2"	0.00394
Scenario Outline: unit fizz buzz : 3 must return Fizz (3)	
Test 19 : * configure connectTimeout = 3000	0.004832
Test 20 : * configure readTimeout = 3000	0.000118
Test 21 : * url serverUrl	0.000028
Test 22 : Given path "/fizzbuzz/standard/unit"	0.000096
Test 23 : And param input = "3"	0.002031
Test 24 : When method GET	0.005025
Test 25 : Then status 200	0.000054
Test 26 : And assert response.input === 3	0.003671
Test 27 : And assert response.result === "Fizz"	0.00335
Scenario Outline: unit fizz buzz : 4 must return 4 (4)	
Test 28 : * configure connectTimeout = 3000	0.004855
Test 29 : * configure readTimeout = 3000	0.000136
Test 30 : * url serverUrl	0.000034

Comment factoriser des tests avec Karate

En utilisant la
fonctionnalité

Scenario Outline de
Cucumber

1 test par exemple

```
Scenario Outline: unit fizz buzz : <Case>
  Given path "/fizzbuzz/standard/unit"
  And param input = "<Input>"
  When method GET
  Then status 200
  And assert response.input === <Input>
  And assert response.result === "<Result>"
```

Examples:

Case	Input	Result
1 must return 1	1	1
2 must return 2	2	2
3 must return Fizz	3	Fizz
4 must return 4	4	4
5 must return Buzz	5	Buzz
6 must return Fizz	6	Fizz
7 must return 7	7	7
15 must return FizzBuzz	15	FizzBuzz

Personnalisation de tests en Javascript

```
* def sorted = read('classpath:sorted.js')
```

```
And assert sorted(response, byOne)
```

```
function(array, comparator) {  
  var result = true;  
  for(var i = 1; i < array.length && result ; i++) {  
    if(! comparator(array[i - 1], array[i])) {  
      result = false;  
    }  
  }  
  return result;  
}
```

Limitations / Difficultés

- Parfois complexes (espaces remplacés par des + dans les paramètres par exemple)
- Quand un test est compliqué, il devient moins lisible
- Nous ne sommes pas arrivés à utiliser les syntaxes Javascript ES6
- Besoin d'une JVM et de maven (ou gradle)

Liens

- Github : <https://github.com/intuit/karate>
- Documentation de karaté : <https://intuit.github.io/karate/>
- Karaté sur stackoverflow :
<https://stackoverflow.com/questions/tagged/karate>
- Documentation de cucumber : <https://cucumber.io/docs>
- Démo : <https://github.com/GeoffreyBoulay/KarateFizzBuzz>
- Fizz Buzz : https://en.wikipedia.org/wiki/Fizz_buzz

Démo sur Github

Résumé

- Test de non régression d'API REST
- Tests des résultats avec des syntaxes Javascript
- Personnalisable
- Lisible et réutilisable par les non-développeurs