KARATE FRAMEWORK

POWERFUL AND SIMPLE FRAMEWORK FOR REST API AUTOMATION TESTING

DREAM BUSINESS SCENARIO

Automation Done During Story Writing

Automation Done During Grooming Session

Developer Or Test Engineer Can Write Script

Automation Supports Continuous Integration

DREAM AUTOMATION SCENARIO

Code Review Done In 5 Minutes

New Automated Script In 30 Minutes

Auto Generated Reports, Including Performance

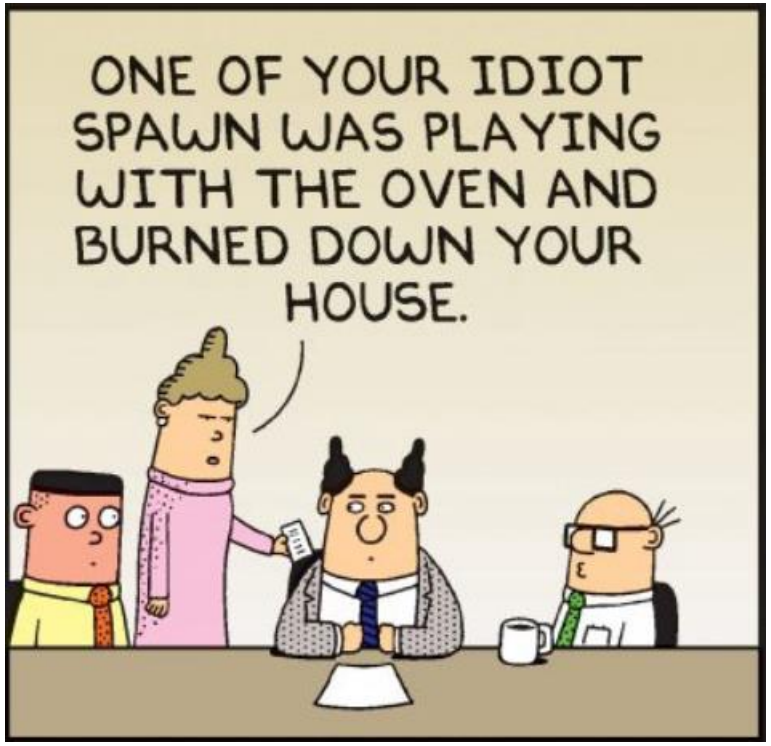
Refactor Script To Accommodate Change In 10 Mints

Mock Service Up and Running In Less Than 10 Mints

CURRENT CHALLENGES

- Support BDD and CI/CD
- Easy, fast way to collaboratively write automation tests
- Writing automation test time consuming (Limited resource, learning curve)
- Shared space API request (Language challenge, single source of truth for API contract)
- Contributing to automation was little hard (Shared environment/Release candidate, Time to develop and review changes)

IF YOU'VE NEVER FAILED, YOU'VE NEVER LIVED



OBJECTIVES

- Small learning curve
- Support continuous integration,
- Able to maintain it, expand it easily and fast
- Produce reports, test coverage
- Give meaningful detail information when test failed and easy to debug
- Tool should support all requirements and support advance technologies
- Whole team should be able to contribute (Technical and Business People)

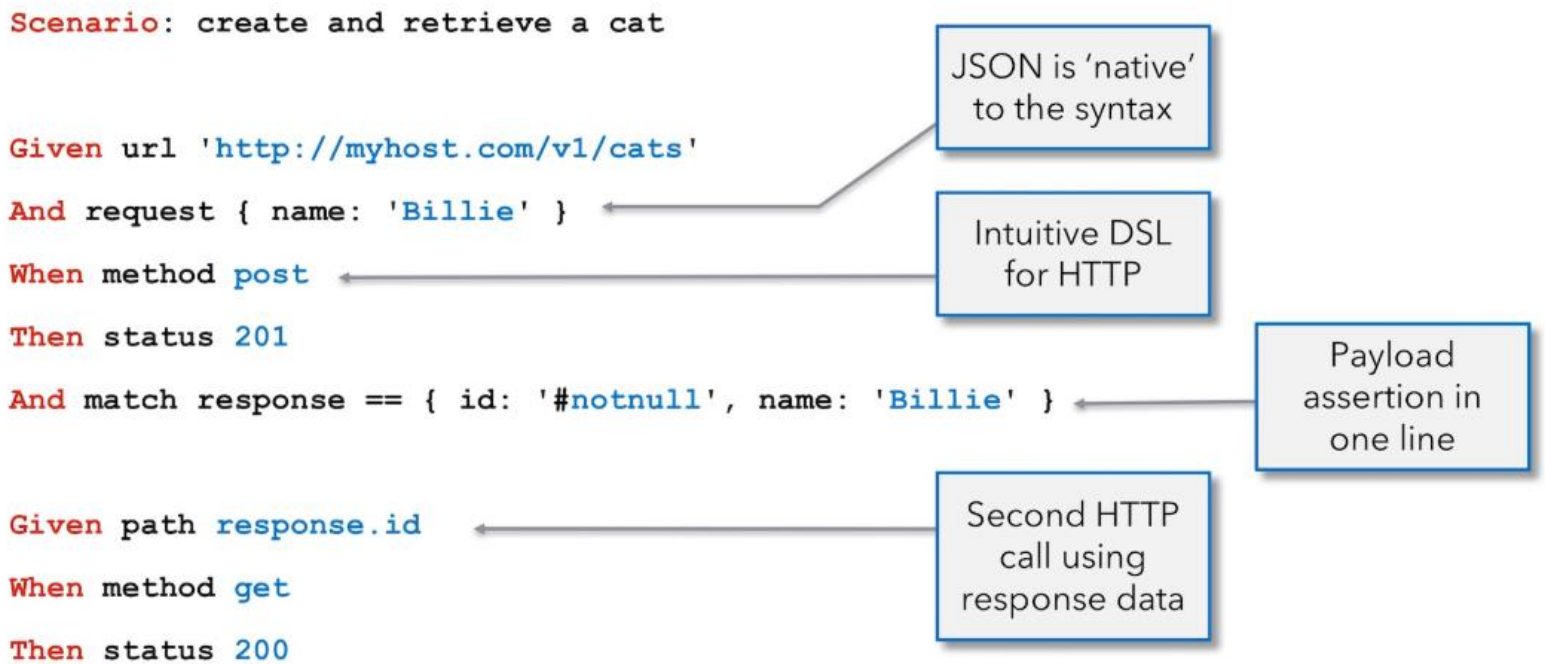
OBJECTIVES…

- Open source
- Performance testing
- Easy to add mock/stubs
- Configure to run tests in parallel
- Able to cover majority positive, negative and boundary conditions
- Able to configure automation suite as per environment (staging/integration/prod)

SOLUTION

WHAT IS KARATE FRAMEWORK

If you are familiar with Cucumber / Gherkin, the big difference here is that **you don't need to write extra "glue" code or Java "step definitions" !**



LESS CODE == LESS BUGS

Karate

```
Scenario: create, retrieve a pet
 Given url 'http://mysite.io/v2/pet'
 And request { name: 'Wolf' }
 When method post
 Then status 200
 And match response ==
      { id:'#number', name:'Wolf' }
```

and ... that's all :)

Cucumber

```
Scenario: create, retrieve a pet
 Given pet endpoint is up
 And request where name is 'Wolf'
 When we send post request
 Then status is 200
 And returned JSON with id, name
```

+ steps
+ business logic
+ pojos

SIMPLER CODE == LESS BUGS

Karate

```
Scenario: create, retrieve a pet
 Given url 'http://mysite.io/v2/pet'
 And request { name: 'Wolf' }
 When method post
 Then status 200
 And match response ==
      { id:'#number', name:'Wolf' }
```

REST-assured

```
public void createPetTest() {
 given()
   .contentType(ContentType.JSON)
   .body("{\"name\": \"Wolf\"}")
   .when()
   .post("http://mysite.io/v2/pet")
   .then().statusCode(200)
   .body("id",
Matchers.instanceOf(Integer.class))
   .body("name", equalTo("Wolf"));
}
```

IF YOU NEED MORE COMPLEXITY

genUserJson.js

```
function(id) {
    var result = {};
    Result.id = id;
    result.name = 'MyPet' + id;
    return result;
}
```

```
Scenario: update the pet
 * def id = 5;
 * def petJson = call read('genUserJson.js') id
 Given url 'http://mysite.io/v2/pet'
 And request petJson
 When method put
 Then status 200
 And match response == { id:'#(id)', name:'#(petJson.name)' }
```

DRY

```
Scenario: update pre-defined pet
 * def createdPet = call read('createPet.feature')
 Given url 'http://mysite.io/v2/pet'
 And request { id:'#(createdPet.response.id)', name:'newPetsName' }
 When method put
 Then status 200
```

ASSERTION CAPABILITIES

```
Scenario: create and retrieve a cat with ID
 Given url 'http://petstore.mysite.io/v2/pet'
 And request { name:'KORAT', id:'a9f7a56b-8d5c-455c-9d13-808461d17b91'
 When method post
 Then status 200
 And match response == { name:'#regex [A-Z]{5}', id:'#uuid' }
```

SCHEMA VALIDATION

```
* def subNode = { price:'#string', name:'#regex[0-9X]' }
* def isValidTime = read('time-validator.js')
When method get
Then match response ==
      """
          {
            id: '#regex[0-9]+',
            odd: '#(subNode)',
            data: {
                    countryId: '#ignore',
                    countryName: '#string',
                    status: '#number? _ >= 0',
                    time: '#? isValidTime(_)'
                  }
          }
      """
```

Markers

```
#ignore
#null
#notnull
#present
#notpresent
#number
#uuid
#array
#object
#string
##string
#? EXPR
#[NUM] EXPR
#regex STR
```

DDT

```
@regression
Scenario Outline: create <name> cat with age = <age>
 Given url 'http://mysite.io/v2/pet'
 And request { name: '<name>', age: '<age>' }
 When method post
 Then status 200
 And match response == { id:'#number', name:'<name>', age:'<age>' }

 # the single cell in the table can be any valid karate expression:
 # csv, js, js function, json, variable, XPath or JsonPath.
 Examples:
 | read('kittens.csv') |
```

kittens.csv

```
name,age
Bob,1
Wild,5
Nyan,3
```

PARALLEL EXECUTION

```
@KarateOptions ( tags = { "@regression", "~@ignore" } )
public class RegressionTests {
 @Test
 public void regressionTests() {
   Results results = Runner.parallel(getClass(), 5, "target/reports");
   assertTrue(results.getErrorMessages(), results.getFailCount() == 0);
 }
}
```

REPORTING

	Steps						Scenarios			Features	
Feature	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
pet-crud.feature	359	0	0	0	0	359	4	0	4	3:9.501	Passed
pet-ddt.feature	337	0	0	0	0	337	6	0	6	4:3.578	Passed
pet-create.feature	192	0	0	0	0	192	4	0	4	2:14.821	Passed
pet-get.feature	50	0	0	0	0	50	2	0	2	47.804	Passed
pet-update.feature	293	0	0	0	0	293	6	0	6	3:40.158	Passed
pet-negative.feature	278	1	0	0	0	279	3	1	4	2:44.472	Failed
pet-boundary.feature	181	0	0	0	0	181	6	0	6	1:58.102	Passed
	6416	**3**	**1**	**0**	**0**	**6420**	**134**	**3**	**137**	**1:10:3.896**	**35**
	99.94%	**0.05%**	**0.02%**	**0.00%**	**0.00%**		**97.81%**	**2.19%**			**91.43%**

OTHER FEATURES

JsonPath, XPath expressions

Reused for perf. testing (Gatling)

Embedded UI for step-by-step debug

Websocket support, async capability

GraphQL API testing capability

Built-in environment switcher

Integration with CI pipelines

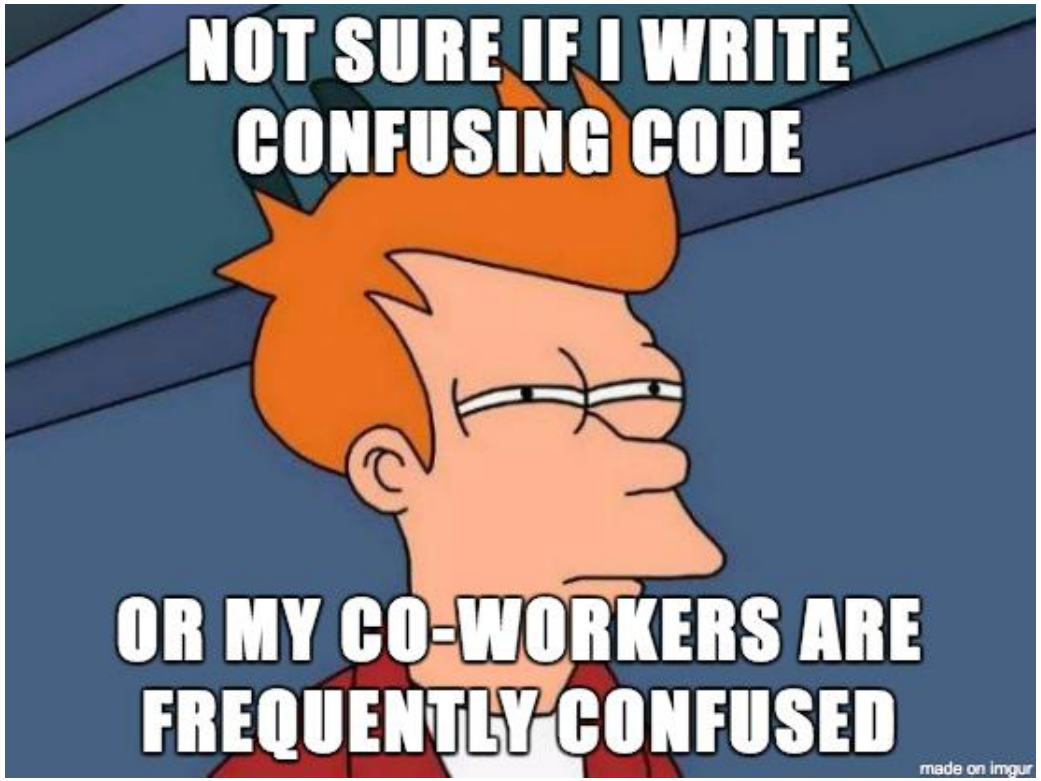
API mock or test-doubles

Allure reporting support

Postman import (exp.)

... and many others

AND DEMO…


NOT SURE IF I WRITE CONFUSING CODE
OR MY CO-WORKERS ARE FREQUENTLY CONFUSED
made on imgur

SUMMARY

- **Low entry barrier**: it's so simple to get started that I believe any manual tester or technical BA can use, easily.
- **Human readable**: no rocket science, just some self descriptive Karate DSL so everyone can understand WTH the test is doing
- **Speed to value**: test suite can be done, real fast! Indeed, I usually brief my automation tester about the API spec and he would finish his test suite & way before I can complete my APIs. When the API is done, he could tell me all the defects he found upon running the test **within an hour.**
- **Test report**: Karate is based on Cucumber so reporting is built in
- **Low maintenance**: this is like the summary of the above