intuit turbotax quickbooks mint

Karate

MicroServices API Testing Made Simple

Peter Thomas | Distinguished Engineer | Intuit

@ptrthomas | @KarateDSL

What you can expect from this session

API Testing

demos

Test Doubles

demos

Perf Testing

demos

Q & A

API-tests with Karate

- Easier to Write
- Readable and Concise
- Faster Execution
- Better Assertions
- Integrates into existing CI / CD

Developer Productivity

Maintain-ability

Quality

Karate

github.com/intuit/karate

Intuit Open Source

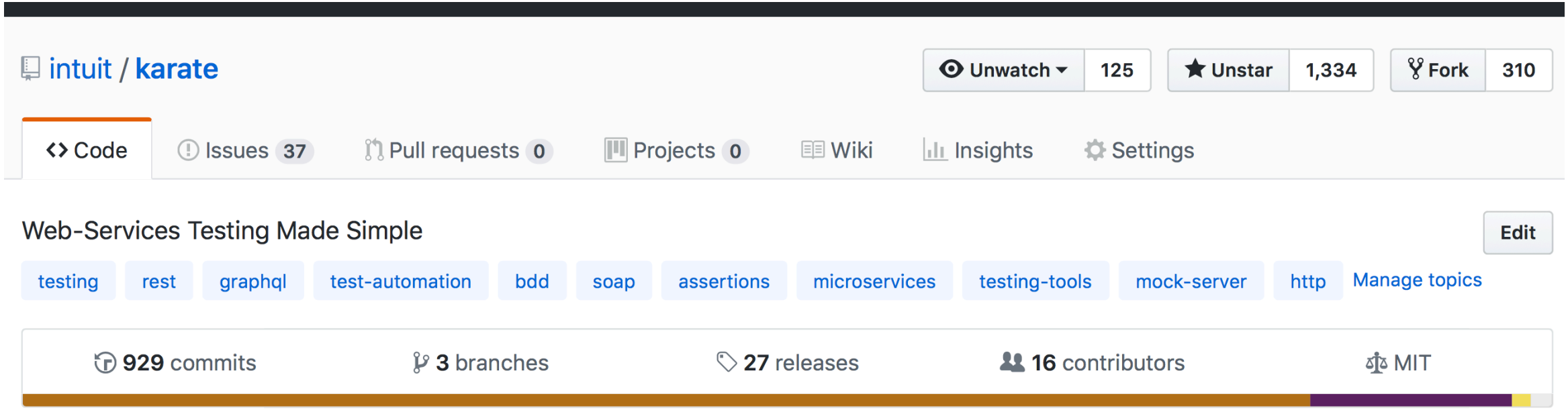
Released Feb-2017

A DSL for writing web-service acceptance tests

Inspired by Cucumber | BDD syntax

GitHub Stats

https://github.com/intuit/karate



Hello World

```
Scenario: create and retrieve a cat

Given url 'http://myhost.com/v1/cats'
And request { name: 'Billie' }
When method post
Then status 201
And match response == { id: '#notnull', name: 'Billie' }

Given path response.id
When method get
Then status 200
```

JSON is 'native' to the syntax

Intuitive DSL for HTTP

Payload assertion in one line

Second HTTP call using response data

Java vs JSON

```java
package com.intuit.karate.demo.domain;

import java.util.ArrayList;
import java.util.List;

public class Cat {

    private int id;
    private String name;
    private List<Cat> kittens;

    public void addKitten(Cat kitten)
        if (kittens == null) {
            kittens = new ArrayList<>(
        }
        kittens.add(kitten);
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name)
        this.name = name;
    }

    public List<Cat> getKittens() {
        return kittens;
    }

    public void setKittens(List<Cat>
        this.kittens = kittens;
    }

}
```

```java
Cat billie = new Cat();
billie.setName("Billie");
Cat bob = new Cat();
bob.setId(23);
bob.setName("Bob");
billie.addKitten(bob);
Cat wild = new Cat();
wild.setId(42);
wild.setName("Wild");
billie.addKitten(wild);
```

```
* def billie =
"""
{
  name: 'Billie',
  kittens: [
      { id: 23, name: 'Bob' },
      { id: 42, name: 'Wild' }
  ]
}
"""
```

```
* match billie.kittens contains { id: '#? _ > 25', name: '#string' }
```

```java
private static boolean hasKitten(Cat cat, Cat kitten) {
    if (cat.getKittens() != null) {
        for (Cat kit : cat.getKittens()) {
            if (kit.getId() == kitten.getId()) {
                if (kit.getName() == null) {
                    if (kitten.getName() == null) {
                        return true;
                    }
                } else if (kit.getName().equals(kitten.getName())) {
                    return true;
                }
            }
        }
    }
    return false;
}
```

Validation examples: JsonPath & RegEx

Type	*Directive*
Ignore / Exists	`#ignore | #null | #notnull`
Primitive Type	`#boolean | #number | #string`
Complex Type	`#array | #object`
Special	`#uuid`
RegEx	`#regex <regex string>`
Predicate	`#? <expression>`

```
Given def cat =
"""
{
  name: 'Billie',
  kittens: [
      { id: 23, name: 'Bob' },
      { id: 42, name: 'Wild' }
  ]
}
"""
```

```
Then match cat.kittens[*].id == [23, 42]

Then match cat.kittens[*].id contains 23
Then match cat.kittens[*].id contains [42, 23]

Then match each cat.kittens contains { id: '#number' }

Then match each cat.kittens == { id: '#notnull', name: '#regex [A-Z][a-z]+' }


* def isLessThanFifty = function(x) { return x < 50 }

Then match each cat.kittens contains { id: '#? isLessThanFifty(_)' }
```

JSON ‘templating’

Type	*Directive*
Replace	#(<expression>)

```
Given def foo = 42
And def bar = { hello: 'world' }
And def temp = { id: '#(foo)', value: '#(bar.hello)' }
Then match temp == { id: 42, value: 'world' }
```

```
Given def temperature = { celsius: 100, fahrenheit: 212 }
Then match temperature contains { fahrenheit: '#($.celsius * 1.8 + 32)' }
```

XML and XPath

```
* def foo =
"""
<records>
  <record index="1">a</record>
  <record index="2">b</record>
  <record index="3" foo="bar">c</record>
</records>
"""

# json style
* assert foo.records.record.length == 3

# xpath assertions
* match foo count(/records//record) == 3
* match foo //record[@index=2] == 'b'
* match foo //record[@foo='bar'] == 'c'
```

```
* def actual = [{ a: 1, b: 'x' }, { a: 2, b: 'y' }]

* def schema = { a: '#number', b: '#string' }
* def partSchema = { a: '#number' }
* def badSchema = { c: '#boolean' }
* def mixSchema = { a: '#number', c: '#boolean' }

* def shuffled = [{ a: 2, b: 'y' }, { b: 'x', a: 1 }]
* def first = { a: 1, b: 'x' }
* def part = { a: 1 }
* def mix = { b: 'y', c: true }
* def other = [{ a: 3, b: 'u' }, { a: 4, b: 'v' }]
* def some = [{ a: 1, b: 'x' }, { a: 5, b: 'w' }]
```

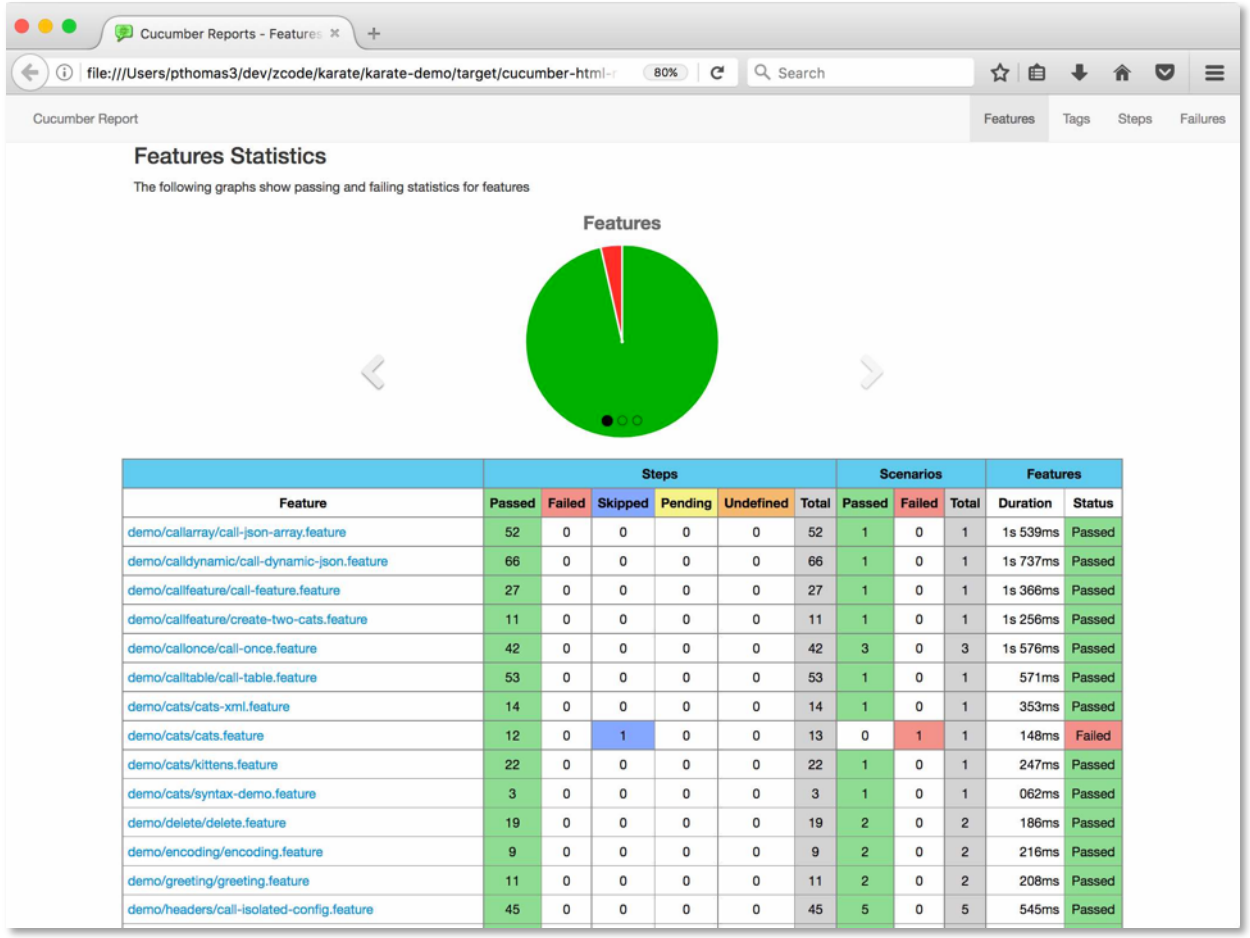
```
* match actual contains '#(^part)'
```

```
* assert actual.length == 2
* match actual == '#[2]'
```

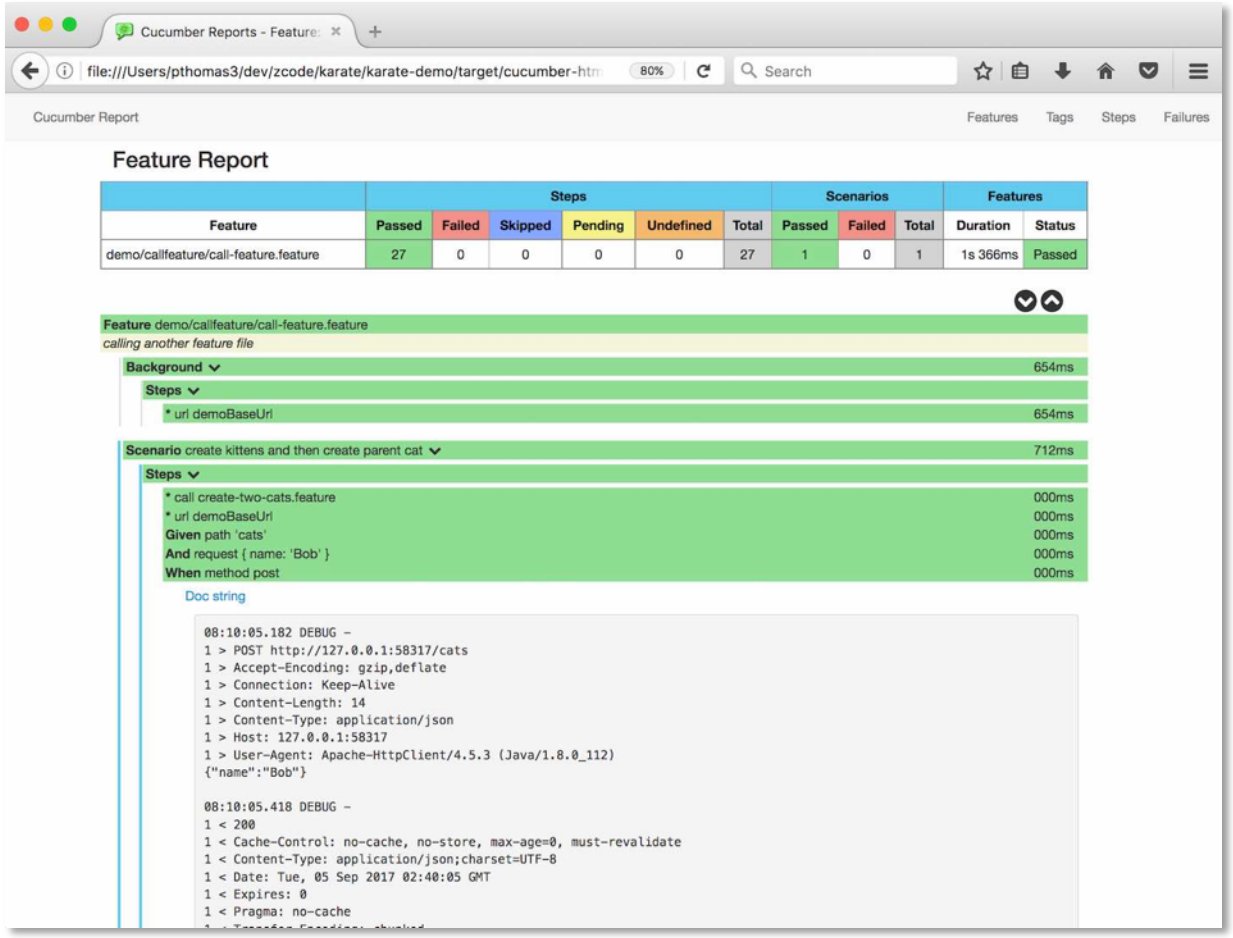
Karate Java Interop

```
# use jdbc to validate
* def config = { username: 'sa', password: '', url: 'jdbc:h2:mem:testdb',
* def DbUtils = Java.type('com.intuit.karate.demo.util.DbUtils')
* def db = new DbUtils(config)
```

Example Report



	Steps						Scenarios			Features	
Feature	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
demo/callarray/call-json-array.feature	52	0	0	0	0	52	1	0	1	1s 539ms	Passed
demo/calldynamic/call-dynamic-json.feature	66	0	0	0	0	66	1	0	1	1s 737ms	Passed
demo/callfeature/call-feature.feature	27	0	0	0	0	27	1	0	1	1s 366ms	Passed
demo/callfeature/create-two-cats.feature	11	0	0	0	0	11	1	0	1	1s 256ms	Passed
demo/callonce/call-once.feature	42	0	0	0	0	42	3	0	3	1s 576ms	Passed
demo/calltable/call-table.feature	53	0	0	0	0	53	1	0	1	571ms	Passed
demo/cats/cats-xml.feature	14	0	0	0	0	14	1	0	1	353ms	Passed
demo/cats/cats.feature	12	0	1	0	0	13	0	1	1	148ms	Failed
demo/cats/kittens.feature	22	0	0	0	0	22	1	0	1	247ms	Passed
demo/cats/syntax-demo.feature	3	0	0	0	0	3	1	0	1	062ms	Passed
demo/delete/delete.feature	19	0	0	0	0	19	2	0	2	186ms	Passed
demo/encoding/encoding.feature	9	0	0	0	0	9	2	0	2	216ms	Passed
demo/greeting/greeting.feature	11	0	0	0	0	11	2	0	2	208ms	Passed
demo/headers/call-isolated-config.feature	45	0	0	0	0	45	5	0	5	545ms	Passed



	Steps						Scenarios			Features	
Feature	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
demo/callfeature/call-feature.feature	27	0	0	0	0	27	1	0	1	1s 366ms	Passed

Feature demo/callfeature/call-feature.feature
calling another feature file

Background 654ms
Steps
* url demoBaseUrl 654ms

Scenario create kittens and then create parent cat 712ms
Steps
* call create-two-cats.feature 000ms
* url demoBaseUrl 000ms
Given path 'cats' 000ms
And request { name: 'Bob' } 000ms
When method post 000ms

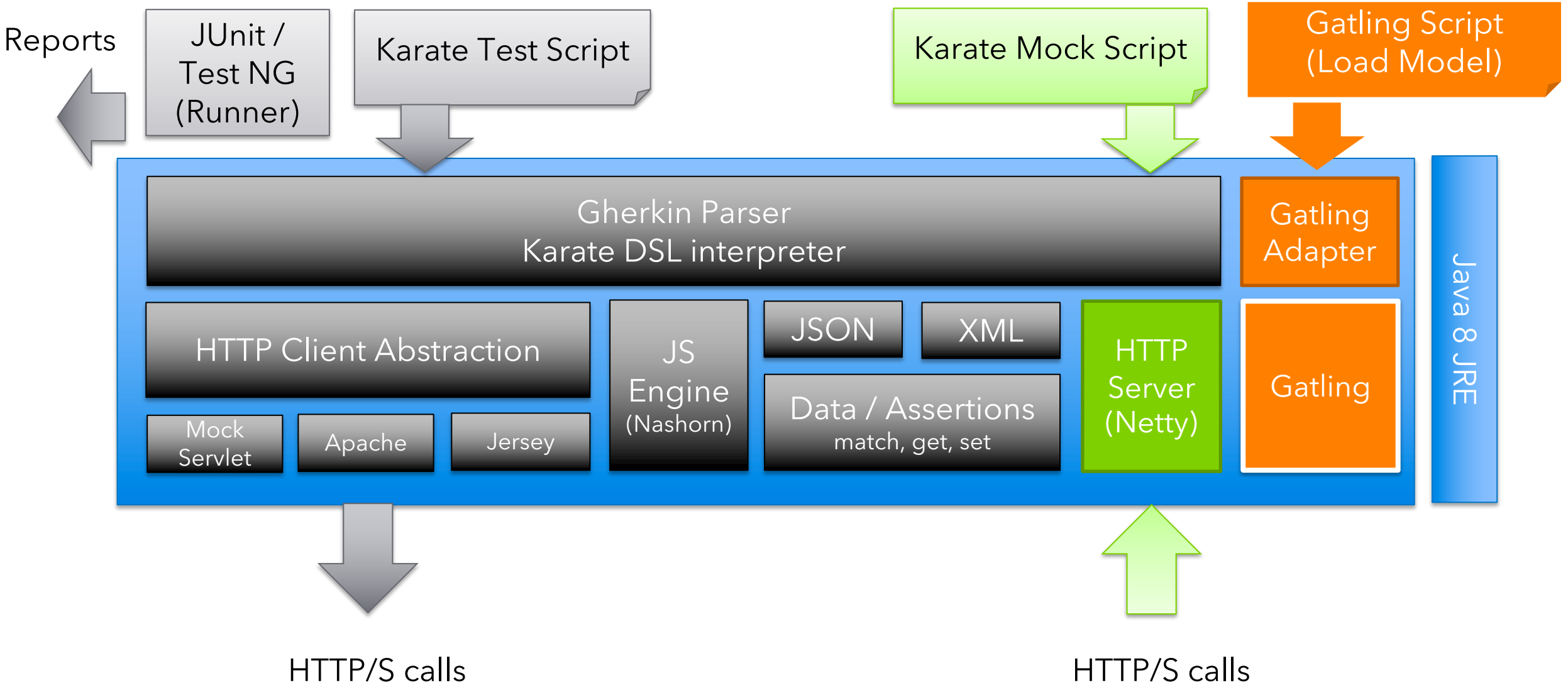
Doc string

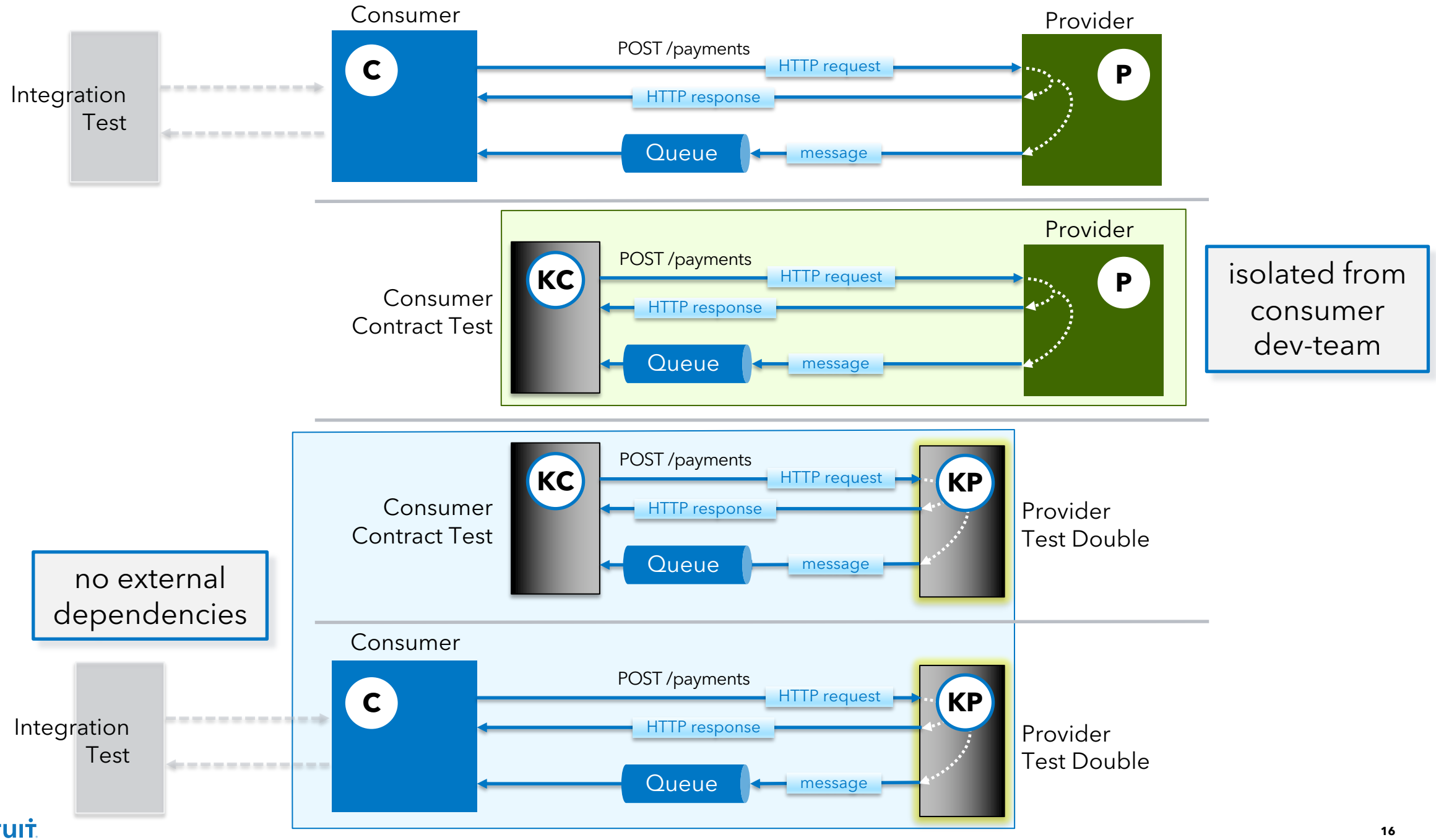
```
08:10:05.182 DEBUG -
1 > POST http://127.0.0.1:58317/cats
1 > Accept-Encoding: gzip,deflate
1 > Connection: Keep-Alive
1 > Content-Length: 14
1 > Content-Type: application/json
1 > Host: 127.0.0.1:58317
1 > User-Agent: Apache-HttpClient/4.5.3 (Java/1.8.0_112)
{"name":"Bob"}

08:10:05.418 DEBUG -
1 < 200
1 < Cache-Control: no-cache, no-store, max-age=0, must-revalidate
1 < Content-Type: application/json;charset=UTF-8
1 < Date: Tue, 05 Sep 2017 02:40:05 GMT
1 < Expires: 0
1 < Pragma: no-cache
```

Demo

Components


Reports
JUnit / Test NG (Runner)
Karate Test Script
Karate Mock Script
Gatling Script (Load Model)
Gherkin Parser
Karate DSL interpreter
Gatling Adapter
Java 8 JRE
HTTP Client Abstraction
Mock Servlet
Apache
Jersey
JS Engine (Nashorn)
JSON
XML
Data / Assertions
match, get, set
HTTP Server (Netty)
Gatling
HTTP/S calls
HTTP/S calls


Consumer
Provider
Integration Test
C
POST /payments
HTTP request
HTTP response
Queue
message
P
Consumer Contract Test
KC
Provider
isolated from consumer dev-team
KP
Provider Test Double
no external dependencies

Comparison with WireMock

```
@BeforeClass
public static void before() {
    System.setProperty("wiremock.port", WIREMOCK_RULE.port() + "");
    String uuid = UUID.randomUUID().toString();
    stubFor(post(urlEqualTo("/v1/cats"))
            .withHeader("Content-Type", equalTo("application/json"))
            .willReturn(aResponse()
                    .withStatus(201)
                    .withHeader("Content-Type", "application/json")
                    .withBody("{ \"id\": \"" + uuid + "\", name: \"Dummy\" }")));
    stubFor(get(urlEqualTo("/v1/cats/" + uuid))
            .willReturn(aResponse()
                    .withStatus(200)
                    .withHeader("Content-Type", "application/json")
                    .withBody("{ \"id\": \"" + uuid + "\", name: \"Dummy\" }")));
    stubFor(post(urlEqualTo("/v1/dogs"))
            .withHeader("Content-Type", equalTo("application/json"))
            .willReturn(aResponse()
                    .withStatus(201)
                    .withHeader("Content-Type", "application/json")
                    .withBody("{ \"id\": \"" + uuid + "\", name: \"Dummy\" }")));
    stubFor(get(urlEqualTo("/v1/dogs/" + uuid))
            .willReturn(aResponse()
                    .withStatus(200)
                    .withHeader("Content-Type", "application/json")
                    .withBody("{ \"id\": \"" + uuid + "\", name: \"Dummy\" }")));
    stubFor(delete(urlEqualTo("/v1/deleteEmptyResponse"))
            .willReturn(null));
    stubFor(get(urlEqualTo("/v1/multiparams"))
            .willReturn(aResponse().withStatus(200)
                    .withHeader("Content-Type", "application/json")
                    .withBody("\n{ \"success\": true }")));
    stubFor(get(urlEqualTo("/v1/spaces"))
            .willReturn(aResponse().withStatus(200)
                    .withHeader("Content-Type", "text/plain")
                    .withBody("\n    \n")));
}
```

```
Scenario: pathMatches('/v1/cats')
    * def responseStatus = 201
    * def response = { id: '#(uuid())', name: 'Dummy' }

Scenario: pathMatches('/v1/cats/{uuid}')
    * def response = { id: '#(uuid())', name: 'Dummy' }

Scenario: pathMatches('/v1/dogs')
    * def responseStatus = 201
    * def response = { id: '#(uuid())', name: 'Dummy' }

Scenario: pathMatches('/v1/dogs/{uuid}')
    * def response = { id: '#(uuid())', name: 'Dummy' }

Scenario: pathMatches('/v1/multiparams')
    * def response = { success: true }

Scenario: pathMatches('/v1/german')
```

```
Feature: stateful mock server

Background:
* def nextId = call read('increment.js')
* def cats = {}

Scenario: pathMatches('/cats') && methodIs('post') && typeContains('xml')
    * def cat = request
    * def id = nextId()
    * set cat /cat/id = id
    * set catJson
      | path | value        |
      | id   | id           |
      | name | cat.cat.name |
    * eval cats[id + ''] = catJson
    * def response = cat

Scenario: pathMatches('/cats') && methodIs('post')
    * def cat = request
    * def id = nextId()
    * set cat.id = id
    * eval cats[id + ''] = cat
    * def response = cat

Scenario: pathMatches('/cats')
    * def response = $cats.*

Scenario: pathMatches('/cats/{id}') && acceptContains('xml')
    * def cat = cats[pathParams.id]
    * def response = <cat><id>#(cat.id)</id><name>#(cat.name)</name></cat>

Scenario: pathMatches('/cats/{id}')
    * def response = cats[pathParams.id]

Scenario: pathMatches('/cats/{id}/kittens')
    * def response = cats[pathParams.id].kittens
```

re-usable JS function from file

initial “state”

XPath

pure JS expression for HTTP request matching

XML to JSON transform

update “state”

JsonPath “query” on state

URI path matching

JSON to XML transform

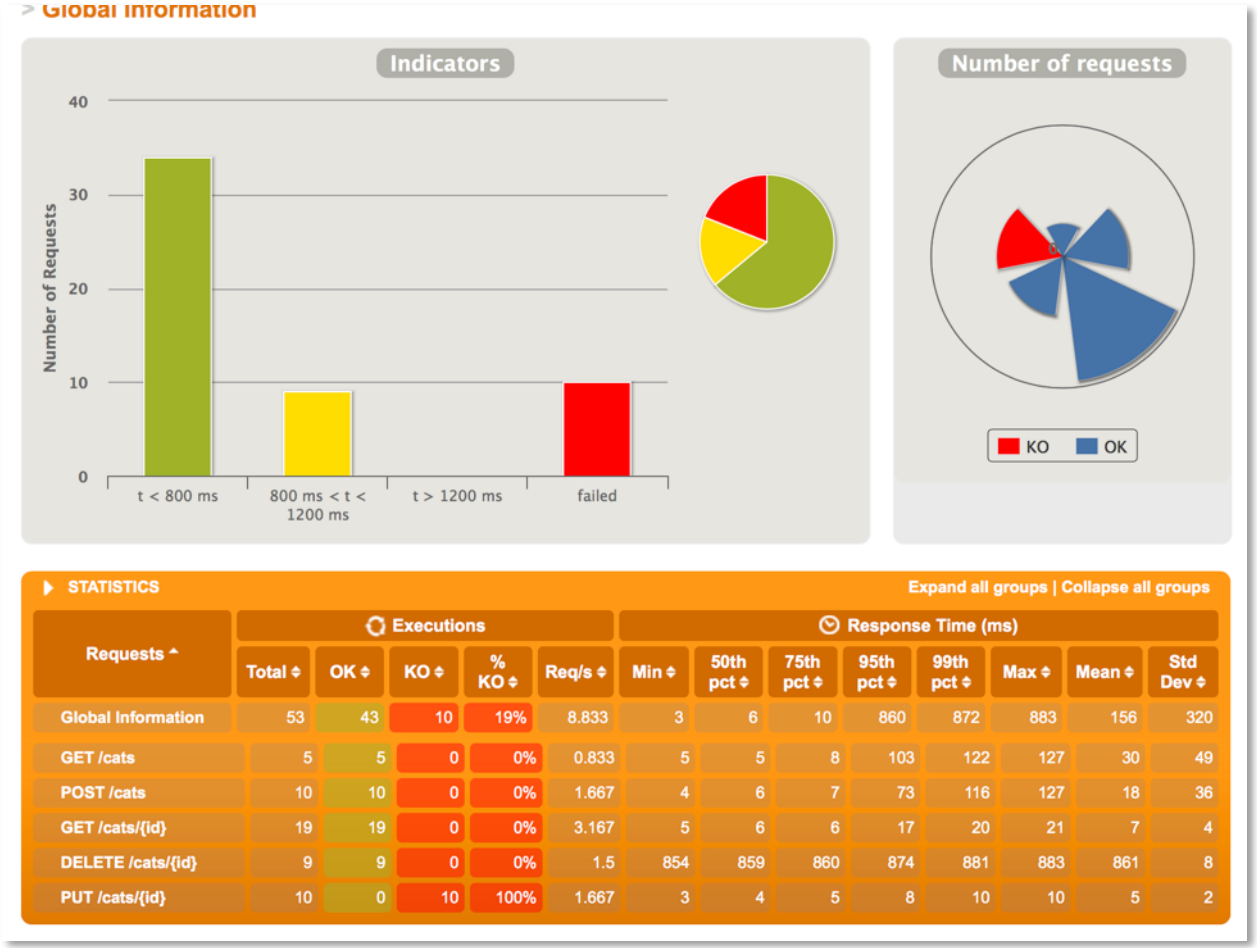
XML response

lookup “state”

Demo

Karate-Gatling

- Re-use Karate scripts as performance tests
- Gatling integration

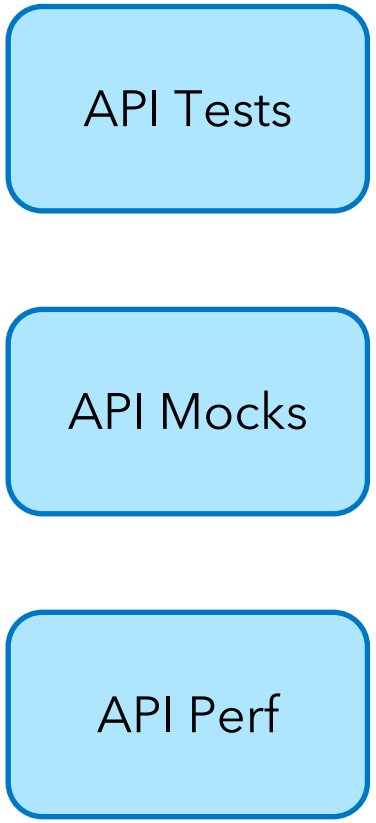


Requests	Executions					Response Time (ms)							
	Total	OK	KO	% KO	Req/s	Min	50th pct	75th pct	95th pct	99th pct	Max	Mean	Std Dev
Global Information	53	43	10	19%	8.833	3	6	10	860	872	883	156	320
GET /cats	5	5	0	0%	0.833	5	5	8	103	122	127	30	49
POST /cats	10	10	0	0%	1.667	4	6	7	73	116	127	18	36
GET /cats/{id}	19	19	0	0%	3.167	5	6	6	17	20	21	7	4
DELETE /cats/{id}	9	9	0	0%	1.5	854	859	860	874	881	883	861	8
PUT /cats/{id}	10	0	10	100%	1.667	3	4	5	8	10	10	5	2

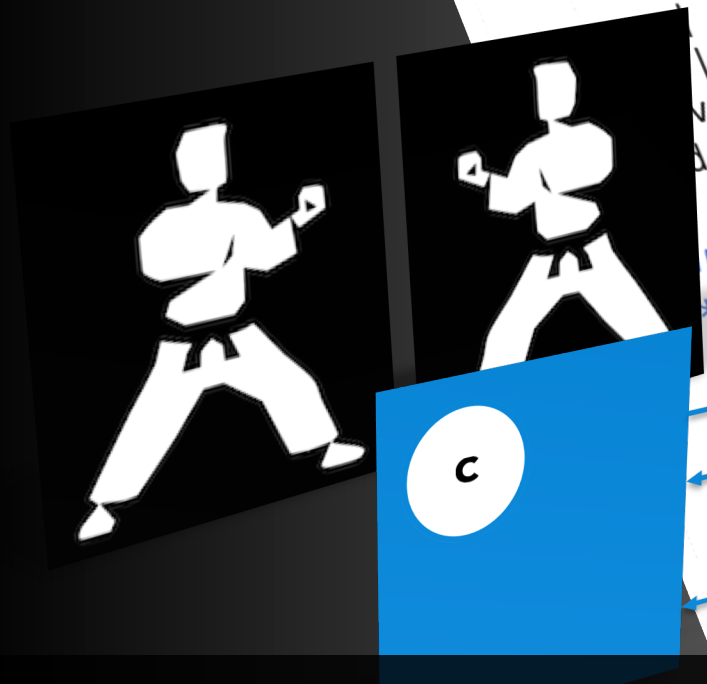
Demo

Karate Features

- REST, GraphQL as well as XML / SOAP support
- User defined extensibility via functions (Java or JavaScript)
- Configuration switching (e.g. dev | e2e | pre-prod)
- Simple 'plug and play' HTTP Header & Auth management
- Comprehensive HTTP support: SSL, Proxy, File-Upload
- JUnit XML reports compatible with all CI tools
- Speed-Run - Parallel Execution of Tests
- Mock Servlet – test without starting a container
- Test Doubles – state-ful "smart" mocks for any HTTP(S) API
- Performance Testing – re-use Karate Scripts as Gatling Scripts
- Websocket / Asynchronous message listener support



Thank You !



https://github.com/intuit/karate

@ptrthomas | @KarateDSL

Thank You !

Q & A

https://github.com/intuit/karate

@ptrthomas | @KarateDSL

Extra Slides

Comparison with REST-assured

Capability	REST-assured	Karate
Compilation not required		
Parallel Execution	? (partial)	
Data Driven Testing	(needs TestNG etc.)	
Environment Switching		
Match full payload in one step. Complex Assertions / "Deep Equals"	(needs Hamcrest etc)	
Update JSON payload / object		
HTTP Mocks / Test-Doubles	(needs Wiremock etc)	
Re-use as Performance Tests		

http://tinyurl.com/karatera

Detailed Comparison

```
@Test public void
lotto_resource_returns_200_with_expected_id_and_winners() {

  when().
    get("/lotto/{id}", 5).
  then().
    statusCode(200).
      body("lotto.lottoId", equalTo(5),
      "lotto.winners.winnerId", containsOnly(23, 54));
```

```
Scenario: lotto resource returns 200 with expected id and winners

Given path 'lotto', 5
When method get
Then status 200
And match $.lotto.lottoId == 5
And match $.lotto.winners[*].winnerId contains only [23, 54]
```

```
  given().
    param("key1", "value1").
    param("key2", "value2").
  when().
    get("/somewhere").
  then().
    body(containsString("OK"));
```

```
Given param key1 = 'value1'
And param key2 = 'value2'
And path 'somewhere'
When method get
Then response contains 'OK'
```

```
@DataProvider(name = "pitstops")
public Object[][] createPitstopData() {
    return new Object[][] {
            { "1", 1 },
            { "2", 3 },
            { "3", 2 },
            { "4", 2 }
    };
}
```

```
Scenario Outline: given race number, validate number of pitstops for Max Verstappen in 2015

Given path 'api/f1/2015/<race>/drivers/max_verstappen/pitstops.json'
When method get
Then assert response.MRData.RaceTable.Races[0].PitStops.length == <stops>
```

```
@Test(dataProvider = "pitstops")
public void checkNumberOfPitstopsForMaxVerstappenIn2015(String raceNumber, int numberOfPitstops) {
    given().
        pathParam("raceNumber", raceNumber).
    when().
        get("/api/f1/2015/{raceNumber}/drivers/max_verstappen/pitstops.json").
    then().
        assertThat().
        body("MRData.RaceTable.Races[0].PitStops",hasSize(numberOfPitstops));
}
```

Karate Environment Switching

karate-config.js

```
function() {
    var env = karate.env; // get java system property 'karate.env'
    karate.log('karate.env system property was:', env);
    if (!env) {
        env = 'dev'; // a custom 'intelligent' default
    }
    var config = { // base config JSON
        appId: 'my.app.id',
        appSecret: 'my.secret',
        someUrlBase: 'https://some-host.com/v1/auth/',
        anotherUrlBase: 'https://another-host.com/v1/'
    };
    if (env == 'stage') {
        // over-ride only those that need to be
        config.someUrlBase = 'https://stage-host/v1/auth';
    } else if (env == 'e2e') {
        config.someUrlBase = 'https://e2e-host/v1/auth';
    }
    // don't waste time waiting for a connection
    // or if servers don't respond within 5 seconds
    karate.configure('connectTimeout', 5000);
    karate.configure('readTimeout', 5000);
    return config;
}
```

```
Feature: cats with kittens

Background:
* url someUrlBase

Scenario: create cat with kittens

# create bob cat
Given path 'cats'
```

```
mvn test -DargLine="-Dkarate.env=e2e"
```

Metrics – port of an open-source example

Purpose	REST-assured File Name	REST-assured	Karate	Karate File Name
/auth end-point helper	src/main/java/com/restfulbooker/api/api/Auth.java	18		
/booking end-point helper	src/main/java/com/restfulbooker/api/api/Booking.java	35		
Auth request payload POJO	src/main/java/com/restfulbooker/api/payloads/request/AuthPayload.java	44		
Part of Booking request POJO	src/main/java/com/restfulbooker/api/payloads/request/BookingDatesPayload.java	26		
Booking request payload POJO	src/main/java/com/restfulbooker/api/payloads/request/BookingPayload.java	105		
Auth response payload POJO	src/main/java/com/restfulbooker/api/payloads/response/AuthResponse.java	19		
Booking details response POJO	src/main/java/com/restfulbooker/api/payloads/response/BookingDetails.java	46		
Part of Booking details POJO	src/main/java/com/restfulbooker/api/payloads/response/BookingDetailsDates.java	24		
Booking response payload POJO	src/main/java/com/restfulbooker/api/payloads/response/BookingResponse.java	29		
Actual Test	src/test/java/com/restfulbooker/api/ApiTest.java	85	51	src/test/java/restfulbooker/api-test.feature
Date format helper			5	src/test/java/restfulbooker/new-date.js
JUnit test runner			11	src/test/java/restfulbooker/ApiTest.java
	Total lines of code	431	67	

Reduction of lines of code from 431 → 67

Link: https://github.com/mwinteringham/api-framework/pull/3

“Service Virtualization” – Competition (OSS)

Consumer Driven Contracts

Tool	URL	CDC	Comment
WireMock	http://wiremock.org		Java, popular
Mountebank	http://www.mbtest.org		ThoughtWorks
Pact	https://docs.pact.io		Steep Learning Curve
Spring Cloud Contract	http://cloud.spring.io/spring-cloud-contract		New, Steep Learning Curve, but “Spring Brand”
Hoverfly	https://hoverfly.io		Cross-Platform

Karate vs JSON Schema

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Product set",
  "type": "array",
  "items": {
    "title": "Product",
    "type": "object",
    "properties": {
      "id": {
        "description": "The unique identifier for a product",
        "type": "number"
      },
      "name": {
        "type": "string"
      },
      "price": {
        "type": "number",
        "minimum": 0,
        "exclusiveMinimum": true
      },
      "tags": {
        "type": "array",
        "items": {
          "type": "string"
        },
        "minItems": 1,
        "uniqueItems": true
      },
      "dimensions": {
        "type": "object",
        "properties": {
          "length": {
            "type": "number"
          },
          "width": {
            "type": "number"
          },
          "height": {
            "type": "number"
          }
        },
        "required": [
          "length",
          "width",
          "height"
        ]
      },
      "warehouseLocation": {
        "description": "Coordinates of the warehouse with the product",
        "$ref": "http://json-schema.org/geo"
      }
    },
    "required": [
      "id",
      "name",
      "price",
      "dimensions"
    ]
  }
}
```

```
Scenario: using karate's simpler alternative to json-schema

* def warehouseLocation = { latitude: '#number', longitude: '#number' }
* def productStructure =
"""
{
  id: '#number',
  name: '#string',
  price: '#number? _ > 0',
  tags: '##[_ > 0] #string',
  dimensions: {
    length: '#number',
    width: '#number',
    height: '#number'
  },
  warehouseLocation: '##(warehouseLocation)'
}
"""
* def json = read('products.json')
* match json == '#[] productStructure'
```

optional array of strings that if present should be non-empty

optional

array

Jordan Mechner liked your Tweet · 11m

> **Karate** @KarateDSL
> introducing the final #KarateDSL logo in all its SVG glory

```
class CatsGatlingSimulation extends Simulation {

  MockUtils.startServer()

  val httpConf = http.baseURL(System.getProperty("mock.cats.url"))

  val create = scenario("create")
    .exec(
      http("POST /cats")
      .post("/")
      .body(StringBody("""{ "name": "Billie" }"""))
      .check(status.is(200))
      .check(jsonPath("$.name").is("Billie"))
      .check(jsonPath("$.id")
        .saveAs("id")))

    .exec(
      http("GET /cats/{id}")
      .get("/${id}")
      .check(status.is(200))
      .check(jsonPath("$.id").is('${id}"))
      // intentional assertion failure
      .check(jsonPath("$.name").is("Billi")))

    .exec(
      http("PUT /cats/{id}")
      .put("/${id}")
      .body(StringBody("""{ "id":"${id}", "name": "Bob" }"""))
      .check(status.is(200))
      .check(jsonPath("$.id").is("${id}"))
      .check(jsonPath("$.name").is("Bob")))

    .exec(http("GET /cats/{id}")
      .get("/${id}")
      .check(status.is(200)))

  val delete = scenario("delete")
    .exec(
      http("GET /cats")
      .get("/")
      .check(status.is(200))
      .check(jsonPath("$[*].id").findAll.optional
        .saveAs("ids")))

    .doIf(_.contains("ids")) {
      foreach("${ids}", "id") {
        exec(
          http("DELETE /cats/{id}")
          .delete("/${id}")
          .check(status.is(200))
          .check(bodyString.is("")))

        .exec(
          http("GET /cats/{id}")
          .get("/${id}")
          .check(status.is(404)))
      }
    }

  setUp(
    create.inject(rampUsers(10) over (5 seconds)).protocols(httpConf),
    delete.inject(rampUsers(5) over (5 seconds)).protocols(httpConf)
  )

}
```

```
Feature: cats crud

  Background:
    * url karate.properties['mock.cats.url']

  Scenario: create, get and update cat
    Given request { name: 'Billie' }
    When method post
    Then status 200
    And match response == { id: '#uuid', name: 'Billie' }
    * def id = response.id

    Given path id
    When method get
    Then status 200
    # intentional assertion failure
    And match response == { id: '#(id)', name: 'Billi' }

    Given path id
    When request { id: '#(id)', name: 'Bob' }
    When method put
    Then status 200
    And match response == { id: '#(id)', name: 'Bob' }

    When method get
    Then status 200
```

```
Feature: delete all cats found

  Background:
    * url karate.properties['mock.cats.url']

  Scenario: get all cats and then delete each by id
    When method get
    Then status 200

    * def delete = read('cats-delete-one.feature')
    * def result = call delete response
```

```
Feature: delete cat by id and verify

  Scenario:
    Given url karate.properties['mock.cats.url']
    And path id
    When method delete
    Then status 200
    And match response == ''

    Given path id
    When method get
    Then status 404
```

```
class CatsKarateSimulation extends Simulation {

  MockUtils.startServer()

  val protocol = karateProtocol("/cats/{id}")

  val create = scenario("create").exec(karateFeature("classpath:mock/cats-create.feature"))
  val delete = scenario("delete").exec(karateFeature("classpath:mock/cats-delete.feature"))

  setUp(
    create.inject(rampUsers(10) over (5 seconds)).protocols(protocol),
    delete.inject(rampUsers(5) over (5 seconds)).protocols(protocol)
  )

}
```

The “missing” Cucumber Features that Karate adds

Capability	Cucumber	Karate
Step Definitions built-in, no Java code needed		
Re-Use Feature files from other Features		
Dynamic Data-Driven Testing		
Parallel Test Execution and Reporting		
Option to run routines only once per Feature		
Embedded JS, Global Config, Env. Switching		

https://stackoverflow.com/questions/47795762/in-karate-how-we-can-collaboratively-work-along-with-ba-to-automate-business-sce