Breizh C@mp

Mix de technologies

Guillaume Laforge

@glaforge

Restlet — the Web API platform

Groovy project team

Take a Groovy Rest!

Restlet

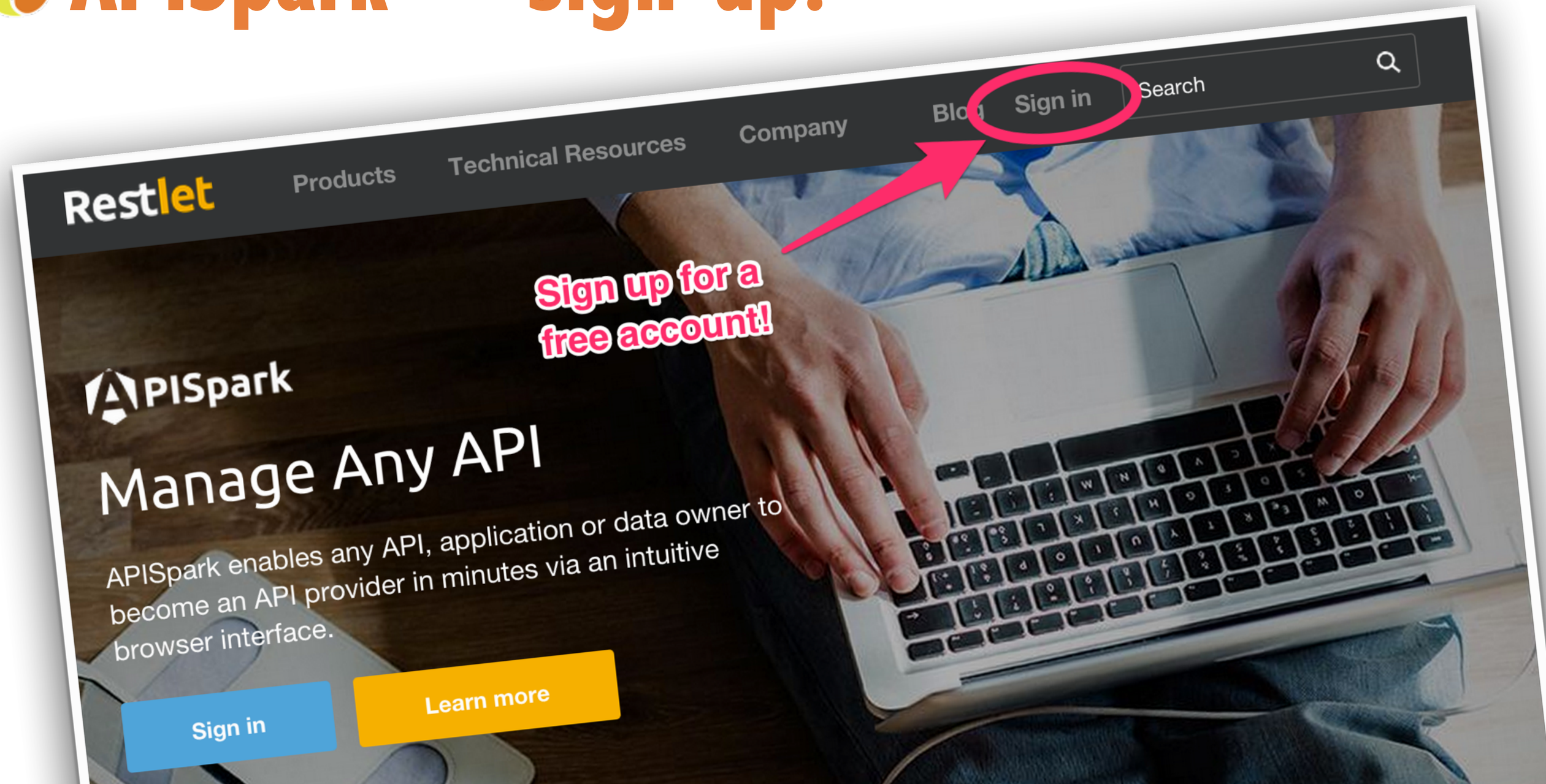
We know about APIs!

RESTLET Studio

RESTLET Framework

RESTLET APISpark

APISpark — sign-up!




STAR
WARS

Quick intro to REST

Restlet

ROY FIELDING

REST

DISSERTATION

ROY FIELDING

REST

DISSERTATION

Representational State Transfer

Architectural properties

- Performance
- Scalability
- Simplicity
- Modifiability
- Visibility
- Portability
- Reliability

Architectural constraints

- Client-server
- Stateless
- Cacheable
- Layered system
- Code on demand *(optional)*
- Uniform interface

REST — Uniform interface

- Identification of **resources**
- Manipulation of resources through **representations**
- Self-descriptive messages
- **HATEOAS**
  (Hypermedia As The Engine Of Application State)

REST — Uniform interface

- Identification of **resources**
- Manipulation of resources through **representations**
- Self-descriptive messages
- **HATEOAS**
  (Hypermedia As The Engine Of Application State)

Resource as URIs
http://api.co/cars/123

REST — Uniform interface

- Identification of **resources**
- Manipulation of resources through **representations**
- Self-descriptive messages
- **HATEOAS**
  (Hypermedia As The Engine Of Application State)

Resource as URIs
http://api.co/cars/123

JSON, XML…

REST — Uniform interface

- Identification of **resources**
- Manipulation of resources through **representations**
- Self-descriptive messages
- **HATEOAS**
  (Hypermedia As The Engine Of Application State)

Resource as URIs
http://api.co/cars/123

JSON, XML…

HTTP GET, POST, PUT, DELETE
media types, cacheability…

REST — Uniform interface

- Identification of **resources**
- Manipulation of resources through **representations**
- Self-descriptive messages
- **HATEOAS**
  (Hypermedia As The Engine Of Application State)

Resource as URIs
http://api.co/cars/123

JSON, XML…

HTTP GET, POST, PUT, DELETE
media types, cacheability…

Hypermedia APIs
HAL, JSON-LD, Siren…

HTTP methods / URIs for collection/item

	http://api.co/v2/cars/	http://api.co/v2/cars/1234
GET	List all the cars	Retrieve an individual car
POST	Create a new car	
PUT	Replace the entire collection with a whole new list of cars	Replace or create an individual car
DELETE	Delete all the cars	Delete an individual car

Common HTTP Status Codes — 1xx

100

Continue

Common HTTP Status Codes — 2xx

200
OK

201
Created

202
Accepted

203
Non-Authoritative Information

204
No Content

206
Partial Content

Common HTTP Status Codes — 3xx

301
Moved Permanently

302
Found

303
See Other

304
Not Modified

307
Temporary Redirect

308
Permanent Redirect

Common HTTP Status Codes — 4xx

400

Bad Request

401

Unauthorized

403

Forbidden

404

Not Found

405

Method Not Allowed

429

Too Many Requests

Common HTTP Status Codes — 5xx

500

Internal Server Error

501

Not Implemented

503

Service Unavailable

509

Bandwidth Limit Exceeded

REST — Server-side

Restlet

Solutions for your REST backend

- Grails
- Ratpack
- Spring Boot
- gServ
- Restlet Framework
- any other Web framework!

LUKE Skywalker


May the Groovy force be with you!
LUKE
Skywalker

gServ

```
def gserv = new GServ()

def bkResource = gserv.resource("/books") {
    get "/faq", file("BooksFaq.html")

    get ":id", { id ->
        def book = bookService.get( id )
        writeJson book
    }

    get "/", {  ->
        def books = bookService.allBooks()
        responseHeader "content-type", "application/json"
        writeJSON books
    }
}

gserv.http {
    static_root  '/public/webapp'
    get '/faq', file("App.faq.html")

}.start(8080)
```

Ratpack

```
@Grab('io.ratpack:ratpack-groovy:0.9.17')
import static ratpack.groovy.Groovy.ratpack
import static groovy.json.JsonOutput.toJson

ratpack {
    handlers {
        get("/people/:id") {
            respond byContent.json {
                response.send toJson(
                    [name: 'Luke Skywalker'])
            }
        }
    }
}
```

Restlet Framework

```
@GrabResolver('http://maven.restlet.com')
@Grab('org.restlet.jse:org.restlet:2.3.2')
import org.restlet.*
import org.restlet.resource.*
import org.restlet.data.*
import org.restlet.routing.*
import static org.restlet.data.MediaType.*

class PeopleResource extends ServerResource {
    @Get('json')
    String toString() { "{'name': 'Luke Skywalker’}" }
}

new Server(Protocol.HTTP, 8183, PeopleResource).with { server ->
    start()
    new Router(server.context).attach("/people/{user}", PeopleResource)

    assert new ClientResource('http://localhost:8183/people/1')
            .get(APPLICATION_JSON).text.contains('Luke Skywalker')

    stop()
}
```

Restlet Framework

```
@GrabResolver('http://maven.restlet.com')
@Grab('org.restlet.jse:org.restlet:2.3.2')
import org.restlet.*
import org.restlet.resource.*
import org.restlet.data.*
import org.restlet.routing.*
import static org.restlet.data.MediaType.*

class PeopleResource extends ServerResource {
    @Get('json')
    String toString() { "{'name': 'Luke Skywalker'}" }
}

new Server(Protocol.HTTP, 8183, PeopleResource).with { server ->
    start()
    new Router(server.context).attach("/people/{user}", PeopleResource)

    assert new ClientResource('http://localhost:8183/people/1')
            .get(APPLICATION_JSON).text.contains('Luke Skywalker')

    stop()
}
```

Starts in milliseconds


Beep, let's move to the client side of the REST force!

REST client-side

tlet

REST clients — Web, Java, Android

Web browser
(client-side JavaScript)

- Raw AJAX, jQuery
- Angular's http request, Restangular
- Restful.js

Java-based app
(Android or Java backend)

- Groovy wslite
- Groovy HTTPBuilder
- RetroFit / OkHttp
- Restlet Framework

+ GrooScript! :-)


WEB

Restful.js (+ GrooScript ?)

An entity is made from the HTTP response data fetched from the endpoint. It exposes a `data()` method:

```
var articleCollection = api.all('articles');  // http://api.example.com/articles

// http://api.example.com/articles/1
api.one('articles', 1).get().then(function(response) {
    var articleEntity = response.body();

    // if the server response was { id: 1, title: 'test', body: 'hello' }
    var article = articleEntity.data();
    article.title; // returns `test`
    article.body; // returns `hello`
    // You can also edit it
    article.title = 'test2';
    // Finally you can easily update it or delete it
    articleEntity.save(); // will perform a PUT request
    articleEntity.remove(); // will perform a DELETE request
}, function(response) {
    // The reponse code is not >= 200 and < 400
    console.log('Invalid response');
});
```


JAVA


JAVA
& GROOVY

new URL('...').text... or not?

```
'https://starwars.apispark.net/v1/people/1'.toURL().text
```

new URL('...').text... or not?

```
'https://starwars.apispark.net/v1/people/1'.toURL().text
```

getText() will do an HTTP GET on the URL

403

403

The API doesn't accept a Java user agent

Know your GDK!

public String **getText(**Map **parameters)**

Read the content of this URL and returns it as a String. The default connection parameters can be modified by adding keys to the *parameters map*:

connectTimeout : the connection timeout

readTimeout : the read timeout

useCaches : set the use cache property for the URL connection

allowUserInteraction : set the user interaction flag for the URL connection

requestProperties : a map of properties to be passed to the URL connection

Parameters:

new URL('...').text... take 2!

```
'https://starwars.apispark.net/v1/people/1'
        .toURL()
        .getText(requestProperties:
            ['User-Agent': 'Firefox',
             'Accept'    : 'application/json'])
```

new URL('...').text... take 2!

```
'https://starwars.apispark.net/v1/people/1'
        .toURL()
        .getText(requestProperties:
            ['User-Agent': 'Firefox',
             'Accept'    : 'application/json'])
```

Let's ask for JSON payload

Now JSON-parsed...

```
import groovy.json.*

assert new JsonSlurper().parseText(
        'https://starwars.apispark.net/v1/people/1'
                .toURL()
                .getText(requestProperties:
                    ['User-Agent': 'Firefox',
                     'Accept': 'application/json'])
).name == 'Luke Skywalker'
```

...and spockified!

```
class StarWars extends Specification {
    def "retrieve Luke"() {
        given: "a URL to the resource"
        def url = 'https://starwars.apispark.net/v1'.toURL()

        when: "I retrieve and parse the people/1"
        def parsed = url.getText(requestProperties:
                ['User-Agent': 'Firefox',
                 'Accept':     'application/json'])

        then: "I expect to get Luke"
        parsed.contains "Luke Skywalker"
    }
}
```

...and spockified!

```
class StarWars extends Specification {
    def "retrieve Luke"() {
        given: "a URL to the resource"
        def url = 'https://starwars.apis                    RL()

        when: "I retrieve and parse the people/1"
        def parsed = url.getText(requestProperties:
                ['User-Agent': 'Firefox',
                 'Accept':     'application/json'])

        then: "I expect to get Luke"
        parsed.contains "Luke Skywalker"
    }
}
```

Neat for GET, but not other methods supported

...and spockified!

```
class StarWars extends Specification {
    def "retrieve Luke"() {
        given: "a URL to the resource"
        def url = 'https://starwars.apis                    RL()

        when: "I retrieve and parse the people/1"
        def parsed = url.getText(requestProperties:
                ['User-Agent': 'Firefox',
                 'Accept':     'application/json'])

        then: "I expect to get Luke"
        parsed.contains "Luke Skywalker"
    }
}
```

Neat for GET, but not other methods supported

Raw text, no JSON parsing

Groovy wslite

```groovy
@Grab('com.github.groovy-wslite:groovy-wslite:1.1.0')
import wslite.rest.*
import wslite.http.*

class StarWars3 extends Specification {
    static endpoint = 'https://starwars.apispark.net/v1'

    def "retrieve Luke"() {
        given: "a connection to the API"
        def client = new RESTClient(endpoint)

        when: "I retrieve the people/1"
        def result = client.get(path: '/people/1', headers:
                ['User-Agent': 'Firefox', 'Accept': 'application/json'])

        then: "I expect to get Luke"
        result.json.name == "Luke Skywalker"
    }
}
```

Groovy wslite

```groovy
@Grab('com.github.groovy-wslite:groovy-wslite:1.1.0')
import wslite.rest.*
import wslite.http.*

class StarWars3 extends Specification {
    static endpoint = 'https://starwars.apispark.net/v1'

    def "retrieve Luke"() {
        given: "a connection to the API"
        def client = new RESTClient(endpoint)

        when: "I retrieve the people/1"
        def result = client.get(path: '/people/1', headers:
                ['User-Agent': 'Firefox', 'Accept': 'application/json'])

        then: "I expect to get Luke"
        result.json.name == "Luke Skywalker"
    }
}
```

Transparent JSON parsing

Groovy wslite

```
def client = new RESTClient(endpoint)


def result = client.get(path: '/people/1', headers:
        ['User-Agent': 'Firefox', 'Accept': 'application/json'])


result.json.name == "Luke Skywalker"
```

Groovy HTTPBuilder

```
@Grab('org.codehaus.groovy.modules.http-builder:http-builder:0.7.1')
import groovyx.net.http.RESTClient
import static groovyx.net.http.ContentType.JSON

class StarWars4 extends Specification {
    static endpoint = 'https://starwars.apispark.net/v1/'

    def "retrieve Luke"() {
        given: "a connection to the API"
        def client = new RESTClient(endpoint)

        when: "I retrieve the people/1"
        def result = client.get(path: ‘people/1', contentType: JSON,
                headers: ['User-Agent': 'Firefox'])

        then: "I expect to get Luke"
        result.data.name == "Luke Skywalker"
    }
}
```

Groovy HTTPBuilder

```
def client = new RESTClient(endpoint)


def result = client.get(path: ‘people/1', contentType: JSON,
        headers: ['User-Agent': 'Firefox'])


result.data.name == "Luke Skywalker"
```

Restlet Framework client

```
@GrabResolver('http://maven.restlet.com')
@Grab('org.restlet.jse:org.restlet:2.3.2')
import org.restlet.resource.*
import static org.restlet.data.MediaType.*

class StarWars extends Specification {
    static endpoint = 'https://starwars.apispark.net/v1'

    def "retrieve Luke"() {
        given: "I create a people resource"
        def resource = new ClientResource("${endpoint}/people/1")

        when: "I retrieve the resource"
        def representation = resource.get(APPLICATION_JSON)

        then: "I expect that resource to be Luke!"
        representation.text.contains "Luke Skywalker"
    }
}
```

Restlet Framework client

```groovy
@GrabResolver('http://maven.restlet.com')
@Grab('org.restlet.jse:org.restlet:2.3.2')
import org.restlet.resource.*
import static org.restlet.data.MediaType.*

class StarWars extends Specification {
    static endpoint = 'https://starwars.apispark.net/v1'

    def "retrieve Luke"() {
        given: "I create a people resource"
        def resource = new ClientResource("${endpoi

        when: "I retrieve the resource"
        def representation = resource.get(APPLICATION_JSON)

        then: "I expect that resource to be Luke!"
        representation.text.contains "Luke Skywalker"
    }
}
```

Raw text, no JSON parsing

Restlet Framework client

```
def resource = new ClientResource("${endpoint}/people/1")

def representation = resource.get(APPLICATION_JSON)

representation.text.contains "Luke Skywalker"
```

RetroFit

```groovy
@Grab('com.squareup.retrofit:retrofit:1.9.0')
import retrofit.*
import retrofit.http.*

interface StarWarsService {
    @Headers(["Accept: application/json",
              "User-Agent: Firefox"])
    @Get('/people/{id}')
    People retrieve(@Path('id') String id)
}

class People { String name }

def restAdapter = new RestAdapter.Builder()
        .setEndpoint('https://starwars.apispark.net/v1/')
        .build()

def service = restAdapter.create(StarWarsService)
assert service.retrieve('1').name == 'Luke Skywalker'
```

Miscellaneous

httpie

```
[09:51:37] glaforge@glaforge-mbp: scripts $ http https://starwars.apispark.net/v1/people/1
HTTP/1.1 200 OK
Accept-Ranges: bytes
CF-RAY: 1f01f136a86d068b-LAX
Connection: keep-alive
Content-Length: 690
Content-Location: http://swapi.co/api/people/1
Content-Type: application/json
Date: Tue, 02 Jun 2015 08:55:57 GMT
ETag: W/"8b5b704987769e6152784a56632a85f2"
Server: cloudflare-nginx
Vary: Accept
Via: HTTP/1.1 vegur
X-Frame-Options: SAMEORIGIN

{
    "birth_year": "19BBY",
    "created": "2014-12-09T13:50:51.644000Z",
    "edited": "2014-12-20T21:17:56.891000Z",
    "eye_color": "blue",
    "films": [
        "http://swapi.co/api/films/7/",
```

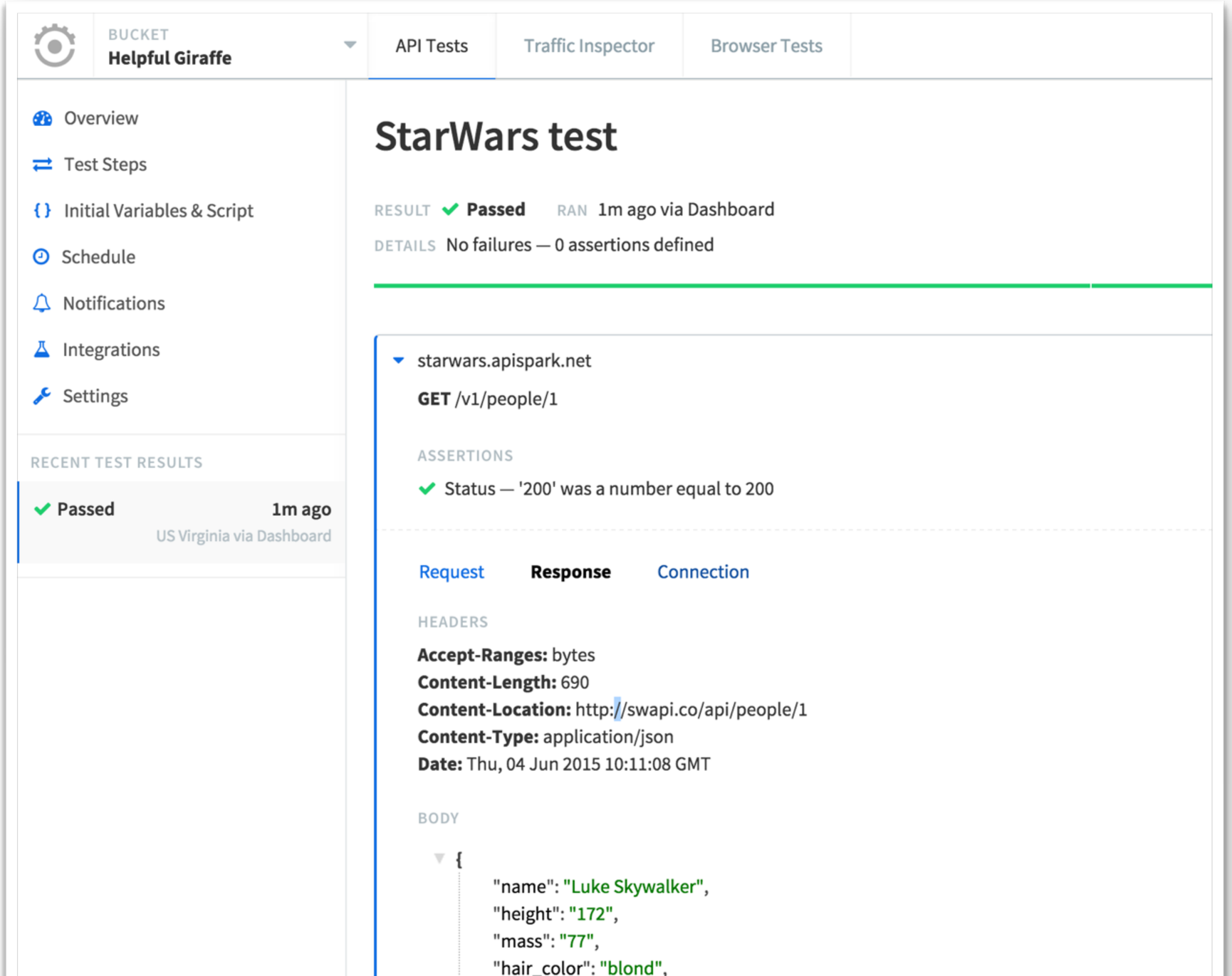
REST-assured

```
@Grab('com.jayway.restassured:rest-assured:2.4.1')
import static com.jayway.restassured.RestAssured.*
import static org.hamcrest.Matchers.*

when().
        get('https://starwars.apispark.net/v1/people/1').
then().
        statusCode(200).
        body('name', equalTo('Luke Skywalker')).
        body('gender', equalTo('male'))
```

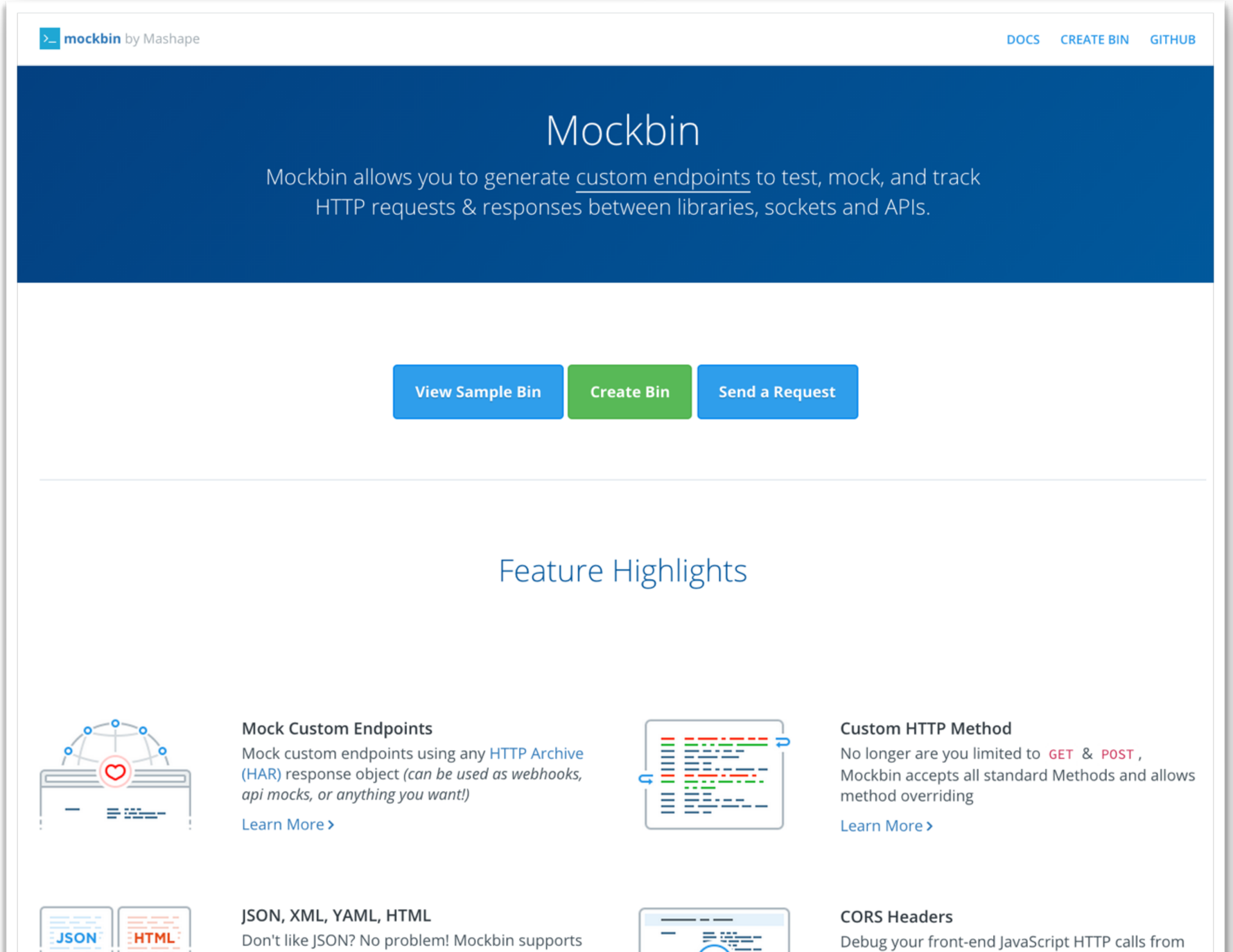
REST-assured — a Spock approach

```
@Grab('org.spockframework:spock-core:1.0-groovy-2.4')
import spock.lang.*

@Grab('com.jayway.restassured:rest-assured:2.4.1')
import static com.jayway.restassured.RestAssured.*
import static org.hamcrest.Matchers.*

class starwars_rest_assured_spock extends Specification {
    def "rest assured and spock"() {
        expect:
        get('https://starwars.apispark.net/v1/people/1').then().with {
            statusCode 200
            body 'name', equalTo('Luke Skywalker')
        }
    }
}
```

Runscope


BUCKET
Helpful Giraffe
API Tests
Traffic Inspector
Browser Tests
Overview
Test Steps
Initial Variables & Script
Schedule
Notifications
Integrations
Settings
RECENT TEST RESULTS
Passed
1m ago
US Virginia via Dashboard
StarWars test
RESULT Passed
RAN 1m ago via Dashboard
DETAILS No failures — 0 assertions defined
starwars.apispark.net
GET /v1/people/1
ASSERTIONS
Status — '200' was a number equal to 200
Request
Response
Connection
HEADERS
Accept-Ranges: bytes
Content-Length: 690
Content-Location: http://swapi.co/api/people/1
Content-Type: application/json
Date: Thu, 04 Jun 2015 10:11:08 GMT
BODY
{
"name": "Luke Skywalker",
"height": "172",
"mass": "77",
"hair_color": "blond",

Mockbin


mockbin by Mashape
DOCS
CREATE BIN
GITHUB
Mockbin
Mockbin allows you to generate custom endpoints to test, mock, and track HTTP requests & responses between libraries, sockets and APIs.
View Sample Bin
Create Bin
Send a Request
Feature Highlights
Mock Custom Endpoints
Mock custom endpoints using any HTTP Archive (HAR) response object (can be used as webhooks, api mocks, or anything you want!)
Learn More ›
Custom HTTP Method
No longer are you limited to GET & POST, Mockbin accepts all standard Methods and allows method overriding
Learn More ›
JSON
HTML
JSON, XML, YAML, HTML
Don't like JSON? No problem! Mockbin supports
CORS Headers
Debug your front-end JavaScript HTTP calls from


dws4.me


Arg! The Groovy REST force is too strong!

Thanks for your attention!

Restlet

Questions & Answers

Restlet