

# [AT Ralph] Restful API Test

**Target:** Introduce about the tool to test API

**For Challenge:** Develop you own restful webservice

## Part 5: Popular test tools

PHP: PHPUnit

Windows / Linux command line: curl

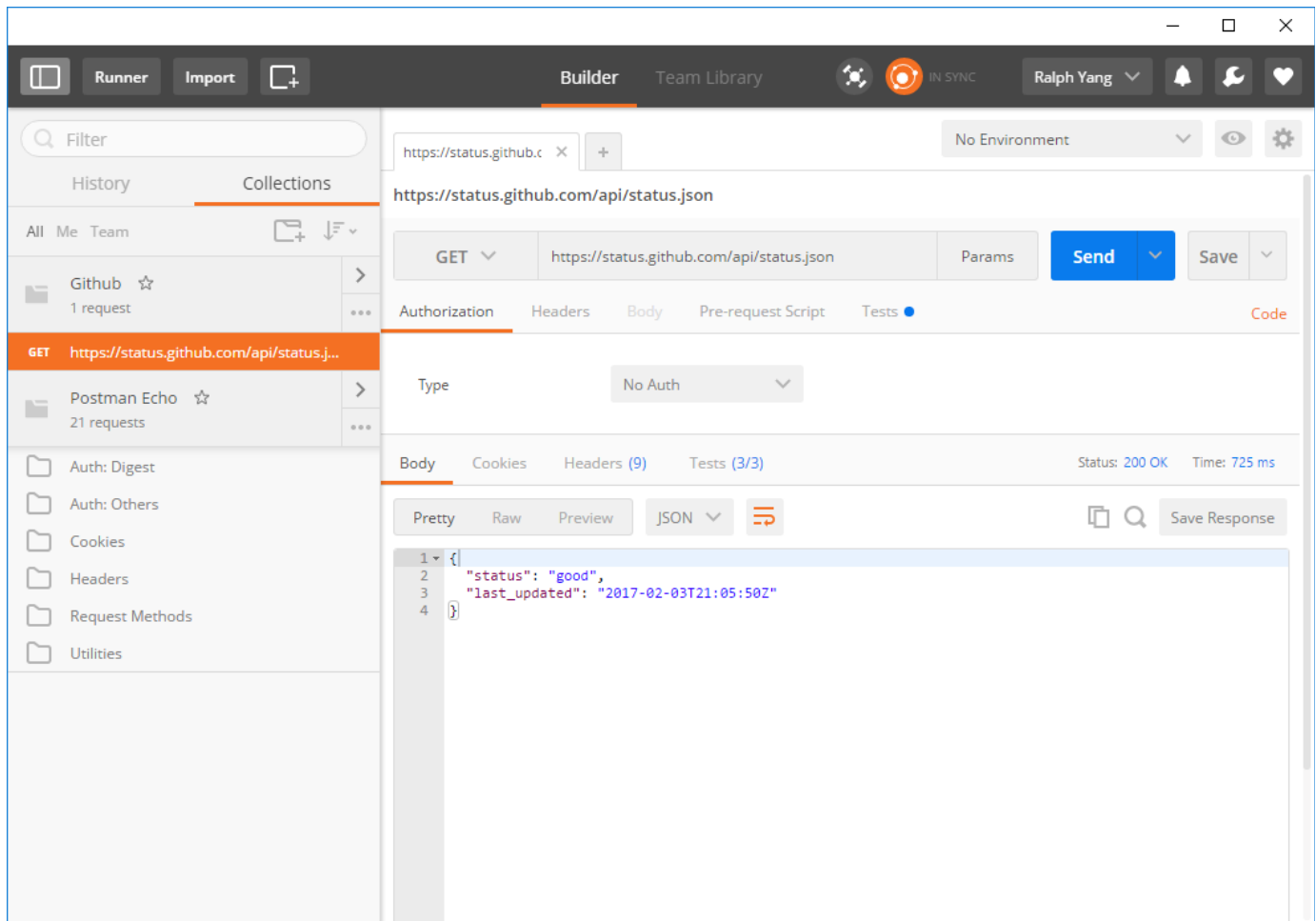
Chrome: Postman, HDC

Firefox: HttpRequester, Poster, RestClient

Other tools: Jmeter, SoapUI, Fiddler

### Postman:

[https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?utm\\_source=plus](https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?utm_source=plus)



How to play with the Postman:

1. Install the Postman and Postman chrome Interceptor plugin
2. Enable the Interceptor in both Postman and chrome browser
3. Login into the [insight.equilar.com](https://insight.equilar.com), see what Postman man capture
4. Try send HTTP Get request, check the return value
5. Write the tests for HTTP api test

6. Save the test into the collections
7. Run collections in Postman RUNNER

A powerful tool easily for API automation test without doing coding, below is example to test <https://status.github.com/api/status.json>

### Postman Tests Script Example

```
tests["Status code is 200"] = responseCode.code === 200;
// validate json schema
var schema = {
  properties: {
    status: {type: 'string'},
    last_updated: {type: 'string'}
  }
};
tests["Valid data schema"] = tv4.validate(responseBody, schema);
// check status
var jsonData = JSON.parse(responseBody);
tests["Github status is good"] = jsonData.status === 'good';
```

#### Practice 1:

Use postman to do below test for QA Insight concurrency session API:

<http://10.1.60.110/insight/api/getData/15362/1>

- Test the valid user active session return 200 ok
- Test the invalid request return 404 error
- Test the valid user session return body status is 15362, session id is string
- Test the valid user session return body ipAddress is "10.1.60.110"

#### ▼ Practice 1 Solution (Don't look at it until you done)

```
tests["response code is 200"] = responseCode.code === 200;

tests["invalid code is 404"] = responseCode.code === 404;

var schema = {
  properties: {
    status: {insightUserId: '15362'},
    sessionId: {type: 'string'}
  }
};

tests["Valid data schema"] = tv4.validate(responseBody, schema);

// check status
var jsonData = JSON.parse(responseBody);
tests["Github status is good"] = jsonData.ipAdress === '10.1.60.110';
```

#### Fiddler:

<https://www.telerik.com/download/fiddler/fiddler2>

Able to monitor all the HTTP traffic in the target machine (monitor all the traffic through 8888 port).

The screenshot shows the Fiddler Web Debugger interface. The left pane displays a list of captured requests, including HTTP and HTTPS traffic to various hosts like api.bing.com, insight.equilar.com, and urs.microsoft.com. The right pane shows a detailed view of a selected POST request to `https://urs.microsoft.com/urs.aspx?query=ins`. The request body is visible in the XML view, showing a complex JSON payload. The bottom status bar indicates the current session is capturing all processes, with 1/459 requests shown.

Practice 2:

Play with the Fiddler, use browser or other tools do some HTTP request, monitor those HTTP request in Fiddler

## Part 7: Program your tests

We already has tools such as Postman, why need use program language to write automation test case?

because \_\_\_\_\_ and \_\_\_\_\_ .

Python: urllib

Java: httpclient, RestTemplate

[TODO]

## Part 8: Develop - Design the restful API

HTTP API Design Guide

<https://github.com/interagent/http-api-design>

Good API example:

Github API v3: <https://developer.github.com/v3/>


Twitter: <https://dev.twitter.com/>

Angelist: API - AngelList


Instagram: <https://www.instagram.com/developer/>

### Some of the API Design approach:


- Restful Version:

 `https://api.example.com/v1/`

- Restful Filtering:

 `?limit=10`: limit the maximum return result  
`?offset=10`: define the start point of the return record  
`?page=2&per_page=100`  
`?sortby=name&order=asc`  
`?animal_type_id=1`

- Error Handle:

 `{`  
    `error: "Invalid API key"`  
`}`

## Part 9: Develop - Develop your own restful webservice

[TODO]

You can build your restful webservice by springboot + swagger in **5 mins**

### Ralph's demo: ugly but works


<https://github.com/ycj28c/SpringBoot-Zoo-Demo.git>

<https://github.com/ycj28c/Stock-Market-All-Stack/tree/master/Jersey-Stock-WebServices>

### Chen's project: Doorman

## Best Practice:

We have a webservice implement below API:

 GET /zoos: list all the zoos  
POST /zoos: create a new zoo  
GET /zoos/ID: get information for particular zoo  
PUT /zoos/ID: update all information for target zoo information  
PATCH /zoos/ID: update some information for target zoo  
DELETE /zoos/ID: delete a zoo  
GET /zoos/ID/animals: list all the animals of the target zoo  
DELETE /zoos/ID/animals/ID: delete target animal in target zoo

You can either run it in your local by <https://github.com/ycj28c/SpringBoot-Zoo-Demo> or visit <http://54.219.154.2:8080/> for practice.

**Practice:** Please use the curl and Postman to test all the GET/POST/PUT/DELETE APIs

Here are some **curl** examples:

```
//Get
curl -X GET --header 'Accept: application/json' 'http://54.219.154.2:8080/zoos'

//Post
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
  "address": "ralph%27s home", \
  "animalsList": [ \
    { \
      "id": 1, \
      "name": "Monkey" \
    } \
  ], \
  "name": "ralph yang test", \
  "website": "www.github.com/ycj28c" \
}' 'http://54.219.154.2:8080/zoos'

//Put
curl -X PATCH --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
  "address": "Put ralph%27s home", \
  "animalsList": [ \
    { \
      "id": 0, \
      "name": "Put Monkey" \
    } \
  ], \
  "name": "Put", \
  "website": "Put www.github.com/ycj28c" \
}' 'http://54.219.154.2:8080/zoos/8'

//Delete
curl -X DELETE --header 'Accept: application/json' 'http://54.219.154.2:8080/zoos/8'
```

Here are the **postman** test example:

Import this JSON AT Ralph Zoo Demo.postman\_collection.json to your postman and run

PS: You can also use swagger document to check the API Description in <http://54.219.154.2:8080/swagger-ui.html>

## Reference:

### Introduction:

- Principles of good RESTful API Design  
<https://codeplanet.io/principles-good-restful-api-design/>
- Best Practices for Designing a Pragmatic RESTful API  
<http://www.vinaysahni.com/best-practices-for-a-pragmatic-restful-api>
- HTTP API Design Guide

<https://github.com/interagent/http-api-design>

- REST best practices

<https://bourgeois.me/rest/>

- Thoughts on RESTful API Design

<https://restful-api-design.readthedocs.io/en/latest/>

## Example:

- Enchant REST API

<http://dev.enchant.com/api/v1>

- Github API v3

<https://developer.github.com/v3/>

## Tools:

Postman

Fiddler: <https://www.telerik.com/download/fiddler/fiddler2>