
Google Books API

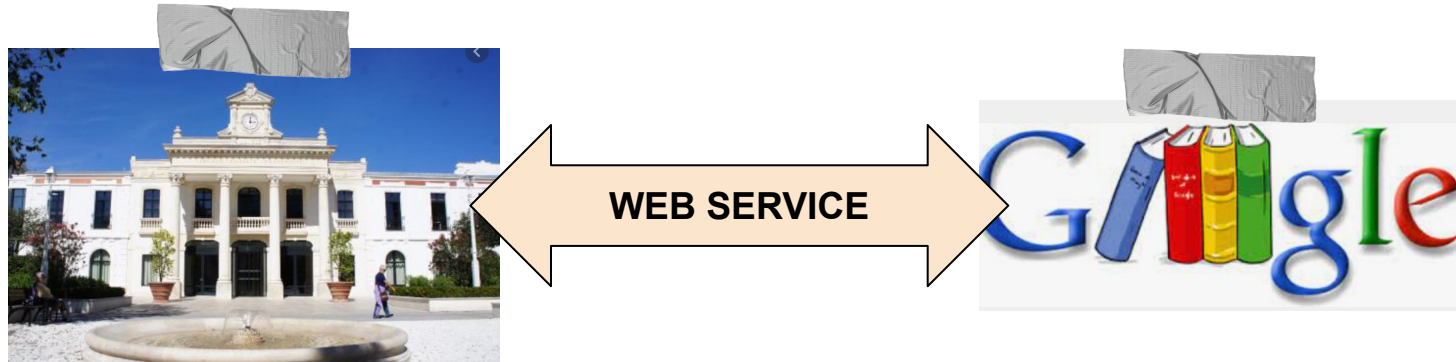
Tests autos

POSTMAN

WCS - Sylvain Viole - Projet 1 - 04/2021

Contexte :

Dans le cadre de la refonte de son application web, la **BIBLIOTHEQUE** de la **MAIRIE d'ARCACHON** souhaite mettre en place un **WEB SERVICE** de collecte d'informations auprès de l'API **GOOGLE BOOKS** en vue d'alléger sa BDD.



—
Objectif :

Tester la fiabilité de l'API GOOGLE BOOKS



Critères :

- **Statut** des réponses
- **Constance** des réponses
- **Cohérences** des réponses
- **Vitesse** des réponses

—
Besoin client :

Rechercher,

Consulter,

Gérer un catalogue



Critères :

- Titre, Auteurs, ISBN, Langue, Date
- Vignette, info, preview
- Ajouter et supprimer un livre du catalogue

Méthodologie : 3 phases

PHASE EXPLORATOIRE

Découverte de l'API
Vérification de la doc

Test manuels

PHASE COMPOSANT

Challenge de use cases
complets

Test autos

PHASE INTEGRATION

Challenge de fonctionnalités
atomiques

Fonctionnalités testées

RECHERCHE :

- **filtres** (inauthor, intitle, isbn)
- **paramètres** (orderBy, maxResults, startIndex, langRestrict),

CONSULTATION :

- Id, auteur, titre
- Langue
- Date
- Thumbnail, info, preview

CATALOGUE :

- **Ajout** d'un livre au catalogue
- **Suppression** d'un livre au catalogue



ENV TECHNIQUE

- **POSTMAN**
Envoie, teste et documente les scénarios et cas de tests.
- **JEU DE DONNEE**
Fichier externe qui va hydrater les requêtes postman en fonction des scénarios de tests. Il permet en outre de la scalabilité sur les itérations et cas de tests.
- **NEWMAN - hmlextra**
Exporte les résultats de tests dans un format lisible.
- **GITHUB / DISCORD**
Déploie les rapports de tests et notifie les équipes.

Organisation du test

Scénario



Requête

Cas de test



pm.test

Itération



jeu de donnée
+
conditions de script

The screenshot shows the Postman interface with a collection named 'gbook_test' expanded. The 'Monitors' tab is selected, showing a test named 'GET [S102] Check content'. The test script is visible in the right pane, showing a series of assertions for a GET request to a book's content. The script includes assertions for the response ID, title, and image links.

```
1 pm.test("[S102C001_${data.iteration}] Response <id> matches request id", () => {
2   pm.expect(pm.response.json().id).to.eql(data.bookId)
3 });
4
5 pm.test("[S102C002_${data.iteration}] Response <title> matches ${data.expected.
6   any', () => {
7   pm.expect(pm.response.json().volumeInfo.title).to.eql(data.expected.title)
8 });
9
10 pm.test("[S102C003_${data.iteration}] Response has <thumbnail> property pointing
11   to an image", () => {
12   pm.expect(pm.response.json().volumeInfo).to.have.property('imageLinks');
13   pm.expect(pm.response.json().volumeInfo.imageLinks).to.have.property
14     ('thumbnail');
15   const imageUrl = pm.response.json().volumeInfo.imageLinks.thumbnail;
16   pm.expect(imageUrl).to.be.a('string');
17   pm.expect(imageUrl).to.satisfy(url => {return url.startsWith('http')});
18   if(imageUrl) {
19     pm.sendRequest(imageUrl, (error, response) => {
20       console.log(error ? error : "image request ok")
21       pm.expect(response.code).to.eql(200);
22       pm.expect(response.headers.get('Content-Type')).includes('image')
23     })
24   }
25 });
26
27 pm.test("[S102C004_${data.iteration}] Response <previewLink> is valid and
28   pointing to an html page", () => {
29   pm.expect(pm.response.json().volumeInfo).to.have.property('previewLink');
30   const previewUrl = pm.response.json().volumeInfo.previewLink;
```

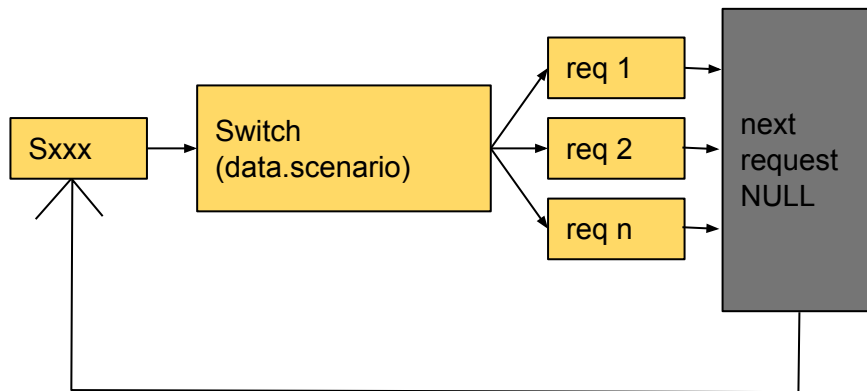

Organisation du jeu de donnée

- Scénario :
- Itération
- Cas de test
- Data input
- Réponse attendue

```
{  
  "scenario": "S089",  
  "iteration": 1,  
  "testCase": "C002",  
  "qData": "arcachon",  
  "parameter": "maxResults",  
  "value": "xxx",  
  "expected": {  
    "code": 400,  
    "msgPart": "max_results"  
  }  
},  
,
```

Routing

Afin d'orienter les itérations du jeu de donnée vers la bonne requête j'ai combiné un switch case sur une requête initiale avec un `setRequest(null)` sur chaque requête.



```
Params Authorization Headers (7) Body Pre-request Script Te

1 pm.collectionVariables.unset('iteration')
2
3 switch (data.scenario) {
4   case "S001":
5     postman.setNextRequest('[S001] Critical Tests');
6     break;
7   case "S002":
8     postman.setNextRequest('[S002] q filter');
9     break;
10  case "S003":
11    postman.setNextRequest('[S003] orderBy');
12    break;
13  case "S004":
14    postman.setNextRequest('[S004] maxResults');
15    break;
16  case "S005":
17    postman.setNextRequest('[S005] startIndex below');
18    break;
```

```
1 if(data.scenario === "S003") {
2   postman.setNextRequest(null)
3 }
4
```

Documentation

Chaque scénario et case de test est documenté en gherkin dans la documentation de POSTMAN

Convention de nommage

[SxxxCyyy_i] <description>

Chaque case de test est nommé par son Scénario, son Cas et son itération :

```
pm.test('[S101C001_${data.iteration}] Response object has <volumeInfo> property
```

TEST SCENARIO : [S004] VALID REQUEST FORMAT WITH MAXRESULTS PARAMETERS

Verb	Description
GIVEN	The Client
WHEN	I request the service for books with valid endpoint
AND	maxResults parameter is set with a valid value
THEN	I should get a valid response with a number of items matching accordingly

1 TEST CASE :

[case]	Description	Iteration
[C001]	Response has a count of at least maxResults items	2

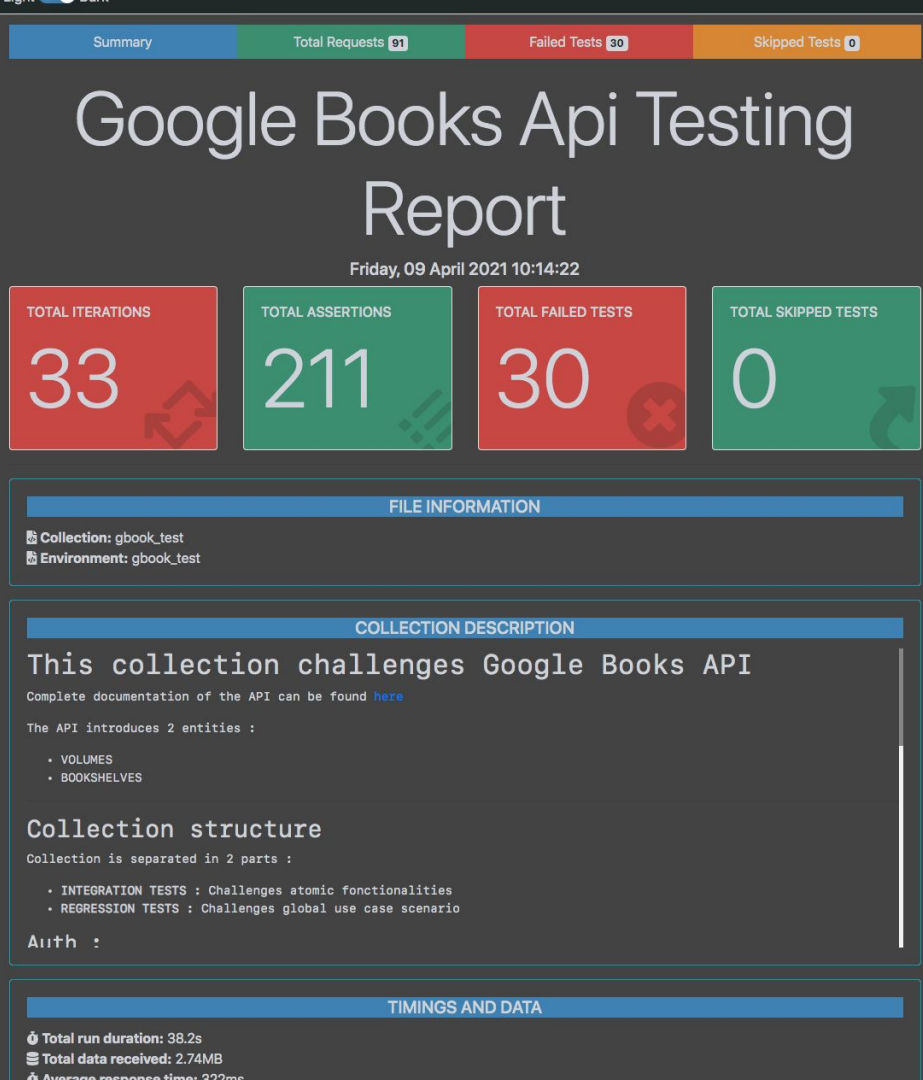
Datas :

- qData = "arcachon", "arcachon"
- maxResults = 20, 40

[case]	Description	Iteration
[C002]	Response has 0 item	1

Datas :

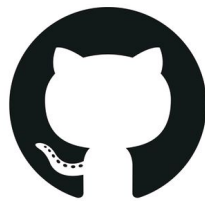
- qData = "arcachon"
- maxResults = 0



RAPPORT de TEST

Les rapports de tests sont générés automatiquement avec l'utilitaire NEWMAN.

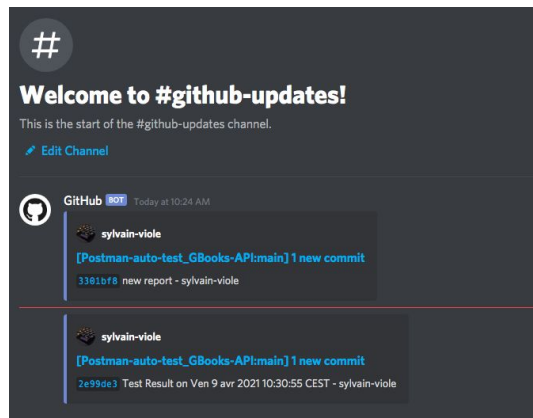
```
→ wcs_projet-1_google-books-api-tests (main) bash launchTest.sh
```



Automatisation et notification

Un script permet de :

- lancer les tests
- pusher le rapport sur un repo GH
- notifier un serveur discord





ANALYSE

→ **Fiabilité de l'API**

Les tests ont permis de remonter que l'API google était fiable à 90%, certaines réponses cependant ne sont pas cohérentes (tri par date, langues, liens)

→ **Actions à mener**

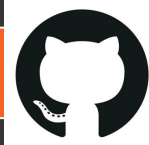
Le client est informé des limitations et intègre ces contraintes dans le dev de son web service.

→ **Recommandation**

S'agissant d'un service externe, il est recommandé de lancer les tests avant chaque itération du web service mais aussi à intervalles régulier.

—

MERCI



https://github.com/sylvain-viole/Postman-auto-test_GBooks-API