Advanced Postman

WOMEN WHO CODE
/london

01
Forking

WOMEN WHO CODE
/london

What is Forking and why we use it

What Is versioning?

- Allows you to work collaboratively to build an API
- Can contribute to a collection without editing the collection

What is forking and why use it?

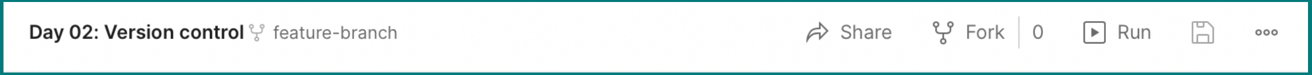
- Forking is a link with the parent collection where you can pull updates and merge to the parent collection
- Can fork collection or environment
- Make updates to fork and merge them to parent element
- Create a pull request where you can let reviewers look at your changes. Reviewers can make comments on your changes and will decide whether to approve them and merge them into the parent element (All part of versioning and the collaborative person)

WOMEN WHO CODE /london

How do we Fork?

Steps to Fork:

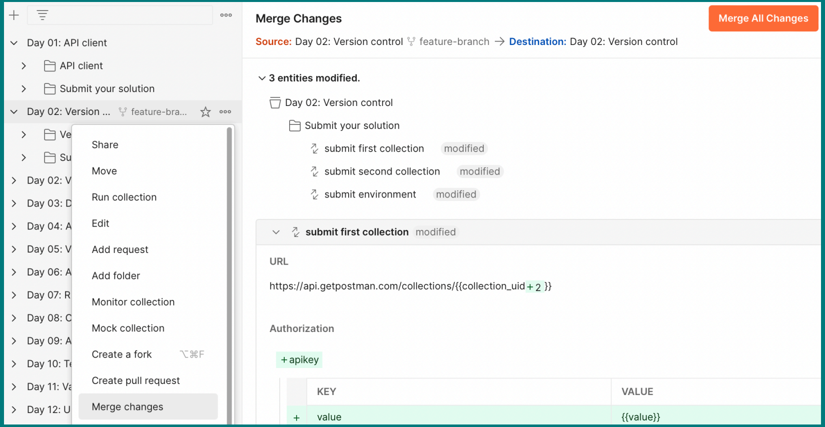
1. Clicking on the name of your collection
2. Clicking on Fork and providing the name



Day 02: Version control feature-branch Share Fork 0 Run

Now that we've forked our collection what can we do?

- Add additional tests
- Make changes to existing tests
- Update the documentation
- Merge Fork collection back into the parent collection and deleting the parent collection (This step also allows you to first check any conflicts and allows you to resolve them)



Merge Changes Merge All Changes

Source: Day 02: Version control feature-branch → Destination: Day 02: Version control

3 entities modified.

Day 02: Version control
Submit your solution
submit first collection modified
submit second collection modified
submit environment modified

submit first collection modified

URL

https://api.getpostman.com/collections/{{collection_uid+2}}

Authorization

+apikey

KEY	VALUE
+ value	{{value}}

02

Mock Servers

WOMEN WHO CODE /london

What is a mock server?
Why we use a mock server?

A mock server is a fake API server that simulates another severs response. We can create one in three ways:
1.From scratch
2.Existing collections where we use responses in our testing
3.From postman history

<u>We may want to use a mock server for the following reasons:</u>

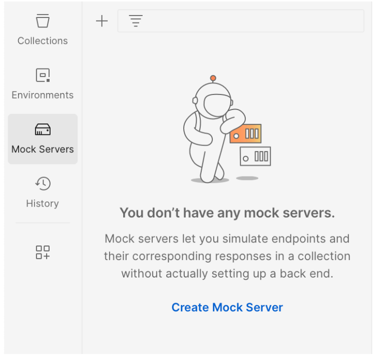
- Test a New API or test new functionality on an existing API
- We want to develop an app but need data from the server that doesn’t exist
- To define the structure, how should the response be like.
- Test specific scenario - development ready and talks to real server
- An example where the real server talks to gateway, has no equivalent for testing purpose
- Test edge cases and failures - server not available, front end app should show error. This server produces this error code - how does the system perform
- Allows development/testing teams to develop or write tests against a service that’s not complete
- Allow you to test the first iteration of the development, during the design process

Once the API is ready to be deployed you swap the mock server with the real server.

WOMEN WHO CODE /london

Configure a mock server

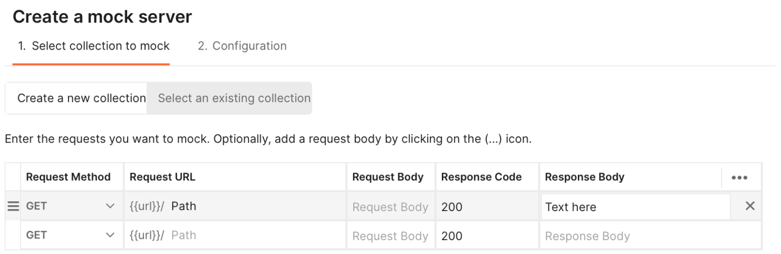
1. New Mock Server
Unique path and response body



You don't have any mock servers.

Mock servers let you simulate endpoints and their corresponding responses in a collection without actually setting up a back end.

Create Mock Server


Create a mock server

Enter the requests you want to mock. Optionally, add a request body by clicking on the (...) icon.

Request Method	Request URL	Request Body	Response Code	Response Body
GET	{{url}}/ Path	Request Body	200	Text here
GET	{{url}}/ Path	Request Body	200	Response Body

2.Create mock server from collection body
Go to collection >> Ellipses >> Mock collection
No environment, save mock server URL

In both cases postman generates the following:

- A collection
- A mock server URL
- A new env which we can select to map our mock server to the URL

WOMEN WHO CODE /london

Examples

Send a request to a mock server and postman matches the request to a saved example in a collection.
Postman then responds with the data you added to the examples.
You can change the response body and status codes

You can also have multiple Examples
In header add x-mock-response-name Name of request

Mock servers allow you to troubleshoot or debug requests and allow you to view logs of the calls you made.

Let’s see all this in action!

WOMEN WHO CODE /london

03

Faker Library

WOMEN WHO CODE /london

Reminder

From the javascript presentation:

- Faker Library lets us generate dynamic variables on the fly
- We would need this due to the Pesticide Paradox

Pesticide Paradox

If the same tests are repeated over and over again, eventually the same tests will not be able to find any new bugs.

So we use the Faker Library to generate dynamic data.

Faker Library Variable List: https://learning.postman.com/docs/writing-scripts/script-references/variables-list/
Starts with a dollar sign and camelCase:

$randomFullName
$randomCity
$randomCompanyName

How can you use Faker in Postman?

- In console to check the variables in pre-requisite script
- Body request - test random data which can check length or certain characters
- As a parameter in your URL

```
{
  "name": "{{$randomFullName}}",
  "organization": "{{randomCompanyName}}",
  "address" : {
    "street": "{{$randomStreetAddress}}",
    "city": "{{$randomCity}}"
  }
  "phone": "$randomPhoneNumber"
}
```

WOMEN WHO CODE /london

04
JSON schema validation

Reminder

JSON is a way to represent data in your request

Parse JSON string, if it’s valid create javascript object
Let personJS = json.parse(json);
console.log(personJS);

Arrays - store data, can find the element of the array using:
array[1] where array is name and 1 is the number of element
(Starts from 0!!)

Some tips on navigating the response

We have the following response, how do we get the yoga Hobby for Jamie from the left hand side:

1.const response = pm.response.json();
2.Use console.log()
3.Always go step by step

i) Array with name Employees
ii) Jamie is the second element - index 1
iii) Array with name Hobbies
iv) yoga is the third element - index 2

Let's go through a couple of examples!!!

```
let employees =[
   {
      "firstName":"Jake",
      "lastName":"Doe",
      "Age":30,
      “Hobbies":[
         “travelling”,
         "reading",
         “hiking”,
      ]
   },
   {
      "firstName":"Jamie",
      "lastName":"Doe",
      "Age":29,
      “Hobbies":[
         “travelling",
         “Music",
         "yoga"
      ]
   }
]
```

Example 1- Navigate JSON schema

```json
{
  "Response": {
    "MetaInfo": {
      "Timestamp": "2021-07-23T07:41:38.720+0000"
    },
    "Data": [
      {
        "_type": "SearchResultsAddress",
        "DataId": 2912,
        "Results": [
          {
            "Relevance": 0,
            "Address": {
              "State": "Colorado",
              "Street": "239 Monares Ln",
              "City": "Monte Vista",
              "PostalCode": "81144",
              "Active": false
            }
          },
          {
            "Relevance": 1,
            "Address": {
              "State": "Colorado",
              "Street": "206 Lyell St",
              "City": "Erie",
              "PostalCode": "80516"
              "Active": true
            }
          }
        ]
      }
    ]
  }
}
```

How do I get to this?

WOMEN WHO CODE /london

Example 2 - validate JSON schema

```
if (pm.response.code == 200);
{
const schema = {
 'type': 'object',
 'properties': {
     'next': {
         type: 'string'
     },
     'count': {
         type: 'number'
     },
     'results': {
         type: 'array'
     }
 },
 required: ['results']
};
```

WOMEN WHO CODE /london

Troubleshooting JSON Formatting

Put something wrong troubleshoot console.log
Check console and can see error

ReferenceError
TypeError
SyntaxError

Inspect variables in the request body in console
Conditional breakpoints

Something wrong with formatting underline in Red
https://jsonformatter.curiousconcept.com/#

Search postman intergalactic find advanced api debugging click on it and fork to your workspace

Let’s see this in action!!!

WOMEN WHO CODE /london

05
Visualiser

WOMEN WHO CODE /london

What is visualiser?

Generates data visualisation in response body:
Pie charts, maps, bar charts, render HTML

If you're generating a lot of data in your response, we can use conditional logic to hide data/ show data

This is set up in test tab in Postman and will display in the visualise tab in Postman response

Two things:

1. *template - add html and css to template*
2. *pm.visualizer.set(template,pm.response.json())*

Let's see this in action!!!

06

Monitor

WOMEN WHO CODE
/london

Why and how do we use a monitor?

If an API is released to Production we can tell by using monitor if the API has gone down or if there's performance issues

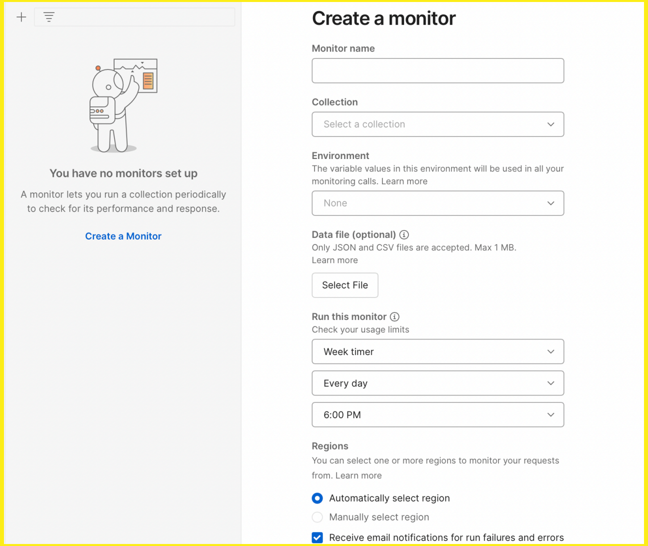
Similar to mock servers we can create them from scratch or we can create them against a collection

We can delete monitors, since there's a limit of 1000 monitoring calls per months for all postman users

If one test case fails the entire monitor fails, we can avoid this by updating the collection so only certain tests run. This feature is called postman.setNextRequest("request you would like Postman to run next") and can be added on the Tests tab.
We can also use Postman.setNextRequest(null) to stop executing the monitor.

Configure a monitor

New monitor



You have no monitors set up

A monitor lets you run a collection periodically to check for its performance and response.

Create a Monitor

Create a monitor

Monitor name

Collection

Select a collection

Environment

The variable values in this environment will be used in all your monitoring calls. Learn more

None

Data file (optional)

Only JSON and CSV files are accepted. Max 1 MB. Learn more

Select File

Run this monitor

Check your usage limits

Week timer

Every day

6:00 PM

Regions

You can select one or more regions to monitor your requests from. Learn more

- [✓] Automatically select region
- [ ] Manually select region

- [✓] Receive email notifications for run failures and errors

Schedule frequency of how often monitor runs overnight

In monitor section we can view the monitor created and we can hit run to execute test cases outside of the schedule run

WOMEN WHO CODE /london

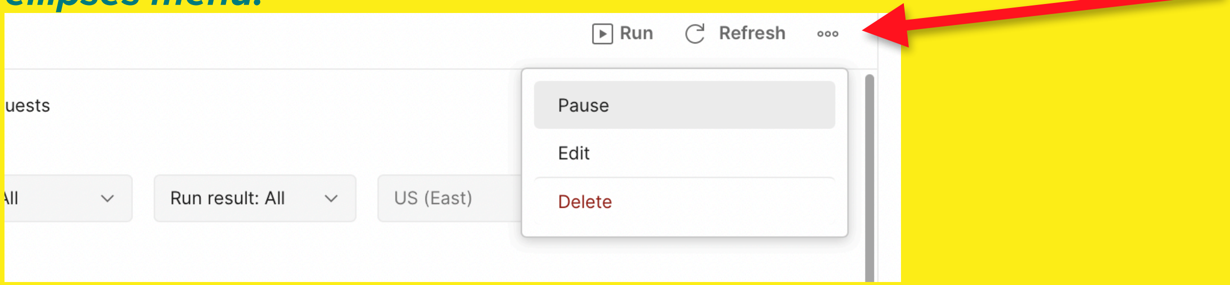
Check a monitor

The monitor will show if the collection of API's is healthy/unhealthy

We can run a summary, see what time a monitor was executed and what tests have passed. Bottom of the page we can see test results, console log, which requests were executed

You can add filters to check an individual request and see how long it takes to execute

We can pause and restart a monitor but if it's not needed anymore we can delete from the ellipses menu:



Let's see this action!!!

07
Newman

Installing and working with Newman

Newman is a way to run a collection (like scheduling a monitor)

First install node.js (Javascript Runtime Environment which can execute Javascript without opening a browser).
Then install Newman (which runs inside node.js runtime environment)

Check terminal afterwards:

```
% node --version
v20.3.0
% newman --version
5.3.2
```

To run a collection:
newman run (link to collection) - this can be the URL or a file location

This generates a report and tests that failed.

Can run html and JUnit reports (stored in XML format).
Different commands you can use:
https://github.com/postmanlabs/newman#junitxml-reporter

08

Graphql

Intro to graphQL

GraphQL versus REST API

- Uses one single end point for all queries, streamlining interactions and simplifying architectures.
- Flexible queries rather then providing predefined data structure
- Easily fetch related data with nested queries - useful for complex data requirements

GraphQL operations

1.Queries - allow clients to request specific data from server in a single efficient request
2.Mutations enable clients to modify or create data on the server in a controlled or predictable manner
3.Subscription provide real time updates to clients by allowing them to subscribe to changes or events from the server

How to configure a request

Two ways to configure requests:

1. New >> graphQL request - endpoint: https://graphql.postman-echo.com/graphql
When you add the URL to your request the schema is loaded using introspection.
In the query tab you can add or remove fields/subfields.
When saving the request you create a new collection since you can't save it in the same place as your HTTP requests

2. Another way to configure requests is to load your schema directly using introspection

Introspection:

GraphQL introspection allows you to query a graphQL server for information about the underlying schema. Postman will auto fetch the schema and provide information on the different fields and data types

Mutation operation

Allows you to create data on the server. Starts with the mutation keyword then the name of your object

```
mutation CreatePerson {
  createPerson(person: {name: "Dwight", age:
36}) {
id
name
age
}}
```

Here you can specify the value for name and age or you can remove these fields from your request.

Mutation operation - Variables

```
mutation CreatePerson($name: String!, $age:
Int!) {
  createPerson(person: {name: $name, age:
$age}) {
id
name
age
}
```

Declare variable and type of the variable.
! Means it’s a required field

“Name” : {{use faker library}}
Or you can declare as collection variables in the scripts tab as a before query

GraphQL scripts are called Before Query and After Response - set collection variables then check if the response returns a status 200

WOMEN WHO CODE /london

Subscription operation

Subscription allows us to fetch data that changes frequently and you can subscribe to the data available in the different fields.

```
subscription Greetings {
greetings
}
```

Will return a stream of data with hello greetings in different languages

WOMEN WHO CODE /london

Summary of GraphQL

Can have multiple operations - query, mutation and subscription in the same query. With the query drop down button you can select which one you want to run

Can't use collection runner since these requests are asynchronous (same with gRPC)

Can find out more here: https://learning.postman.com/docs/sending-requests/graphql/graphql-overview/

Let's see this in action!!!

WOMEN WHO CODE /london

09

gPRC (google remote procedure call)

WOMEN WHO CODE /london

What is gRPC?

Framework for remote procedure calls which is commonly used for micro services and mobile development.

Unlike http web api's like rest, we can do asynchronous and synchronous calls, it's faster, efficient and you can generate client side and server side code from proto file

gRPC uses http2 as a transport layer to submit data between client and server.

HTTP vs HTTP 2

- *gRPC can layer streaming requests*
- *can send multiple packets to server*
- *server can stream response*
- *HTTP single request and single response*
- *HTTP uses https:// or http:// for secured or unsecured connection, gRPC uses TLS toggle*

WOMEN WHO CODE /london

How to configure gRPC?

New >> gRPC
Add the server URL

gRPC supports 4 methods:

1. *unary*

Traditional request and response

2. *server streaming*

Send a request to the server and receive multiple messages back

3. *client streaming*

Send multiple requests, server waiting for many client requests before sending a response

4. *bidirectional streaming*

Both the client and the server can send and receive multiple requests and responses simultaneously on a single connection.

WOMEN WHO CODE /london

Testing gRPC

Server reflection - postman automatically fetches schema or you can import proto file (which generates a mock server).

Message - can generate example message (not sure what payload should look like).

Status code 0 OK: https://grpc.github.io/grpc/core/md_doc_statuscodes.html

Scripts tab - write javascript before invoke method and after we receive a response

End streaming for client and sever streaming - end connection to complete the call

Schema driven framework that uses protocol buffers to describe the interface and structure of payload

Service definition tab: load proto file

Let's see this in action!!!

WOMEN WHO CODE /london

Resources & What’s Next?

Some great videos on advanced Postman testing:
https://www.youtube.com/watch?v=sB2HHrezQOo
https://www.youtube.com/watch?v=zrmQAgixMpU
https://testautomationu.applitools.com/postman-tutorial/

Exercises:
https://quickstarts.postman.com
https://www.postman.com/postman/workspace/15-days-of-postman-for-testers/collection/1559645-032fb22a-9afb-4c56-b8f0-4042db96a4f3
https://academy.postman.com/grpc-and-postman

What’s next
Invite to Slack channel to ask any further questions
Visit our github: https://github.com/WomenWhoCode/WWCode-London/blob/main/resources/quality-guardians/README.md
Complete 30 day challenge with us: https://www.postman.com/postman/workspace/30-days-of-postman-for-developers/overview

WOMEN WHO CODE /london

Thank you

Follow us!
https://linktr.ee/wwcode_london

WOMEN WHO CODE®
/london