INTRODUCTION TO POSTMAN

CONTENT

- About Postman
- Building you first test case in Postman
- Variable Scopes
- Environments
- Importing and Exporting Collections
- Building Assertions
- Collection Runner
- Authentication and Authorization

01

What is Postman?

What is Postman?

Postman is a popular platform, which is used to build, test, modify, document and maintain API. Its simple GUI helps developers and test engineers to view and send HTTP requests.

Key Features

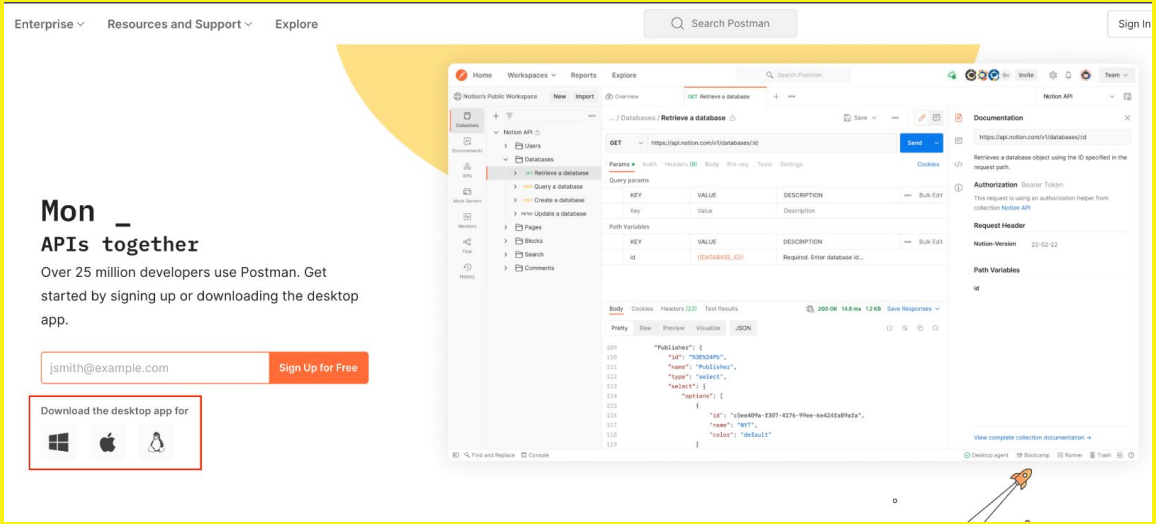
Key features:

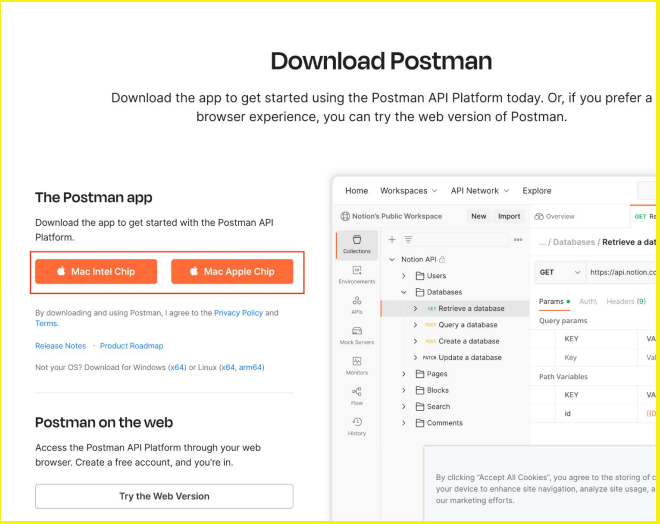
1. API request builder
2. Grouping requests into collections
3. Automating your testing
4. Environment variables
5. Mock Servers
6. Documentation
7. Continuous integration
8. Effective collaboration

and more..

Installation Process

https://www.postman.com/





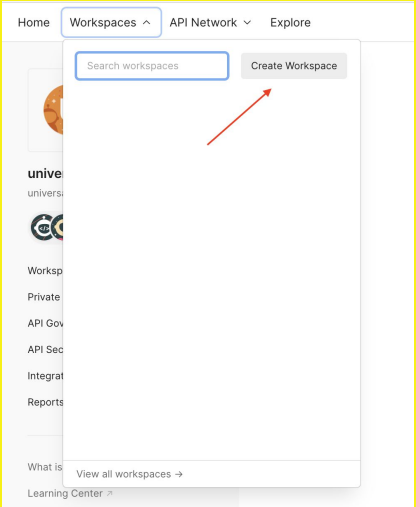
More info: https://learning.postman.com/docs/getting-started/installation-and-updates/

WOMEN WHO CODE /london

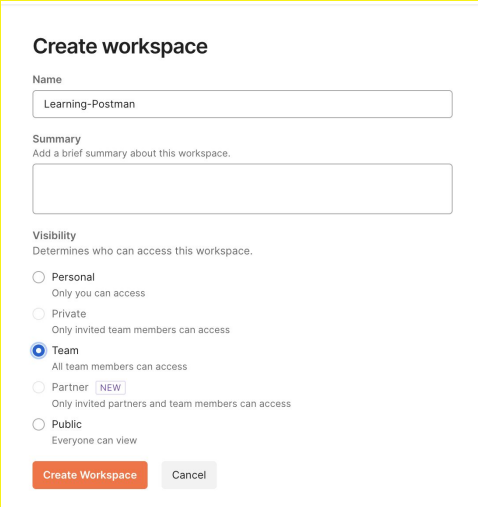
Main blocks in Postman (1)

WORKSPACE allows us to create a high-level structure of your space in Postman.

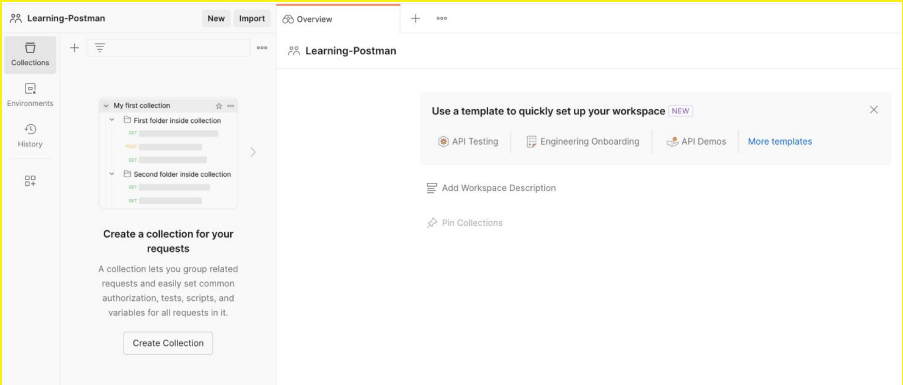
1



2



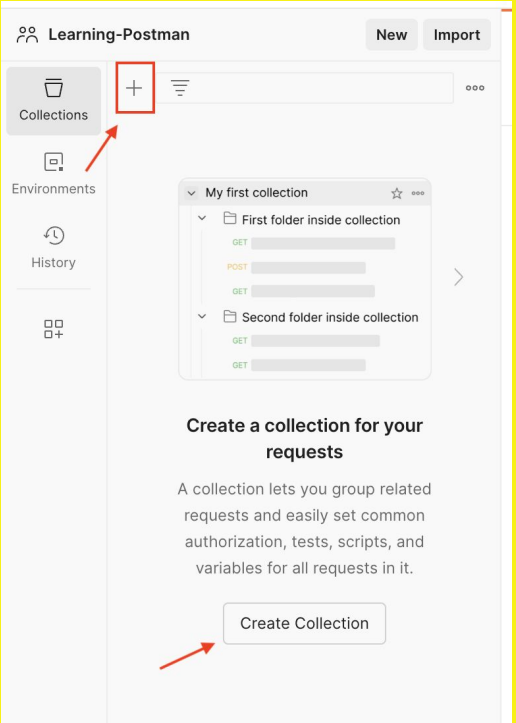
3



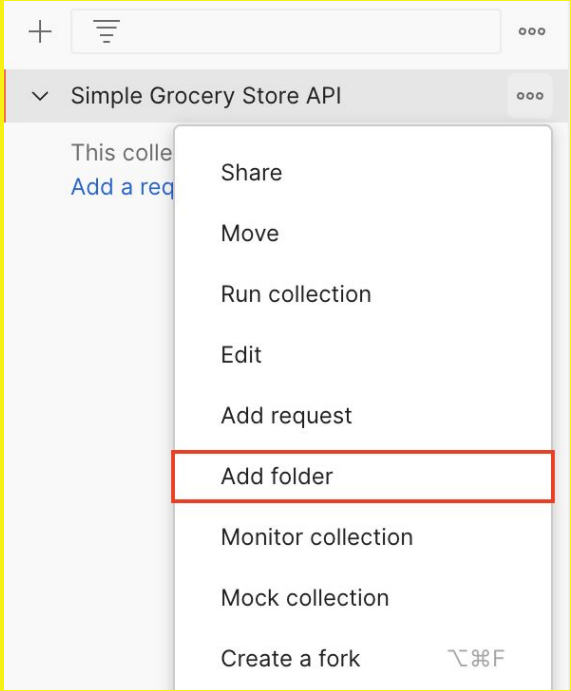
WOMEN WHO CODE /london

Main blocks in Postman (2)

COLLECTIONS are a group of saved requests.



Inside the collection we can group requests using FOLDERS and SUBFOLDERS



WOMEN WHO CODE /london

Main blocks in Postman (3)

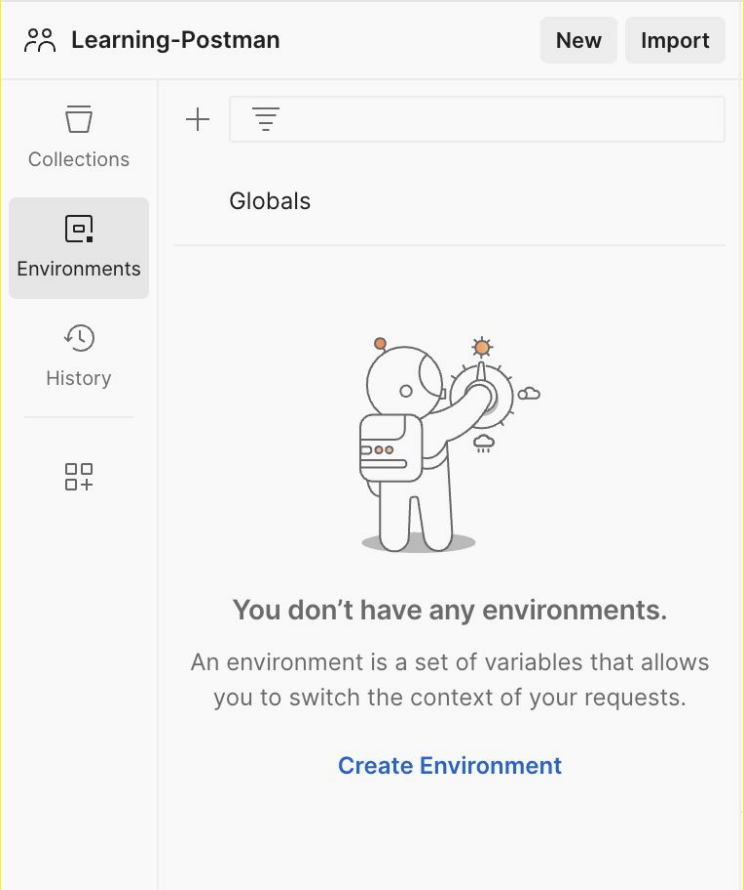
The best practice is:

1. Folders are created for every endpoint. The title of the folder is usually the name of the endpoint.
2. For integration tests it is needed to group API requests into separate folders to test connection between them. In other words, when it is needed to chain multiple requests, folders could be helpful.

More info: https://learning.postman.com/docs/getting-started/navigating-postman/

Environments

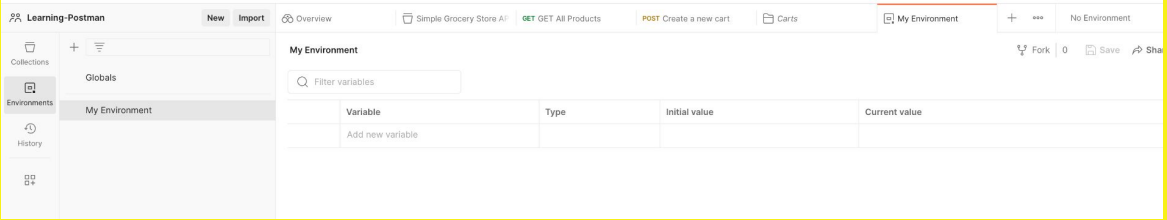
Environments in Postman allows you to run the same set of tests against different data sets. We could have several environment configurations, like development, staging, production. You can access all environments from Environments in the sidebar.



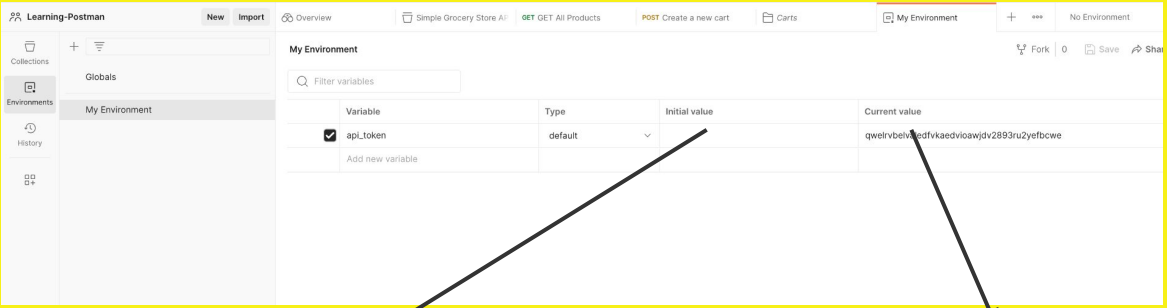
WOMEN WHO CODE /london

Create Environment

1



2



3

Select an active environment

The Initial Value is synced to your account using the Postman servers. It's shared with any collaborators who have access to the environment.

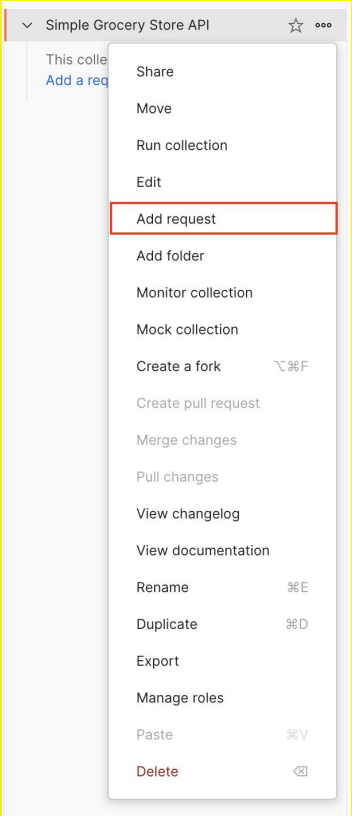
The Current Value is used in your local instance of Postman, and is never synced to your account or shared with your team unless you choose to persist it.

WOMEN WHO CODE /london

02

Sending your first request

Send your first request (1)



The request usually contains:

1. Method
2. Base URL/Endpoint
3. Resource
4. Payload (applicable for certain requests)
5. Headers

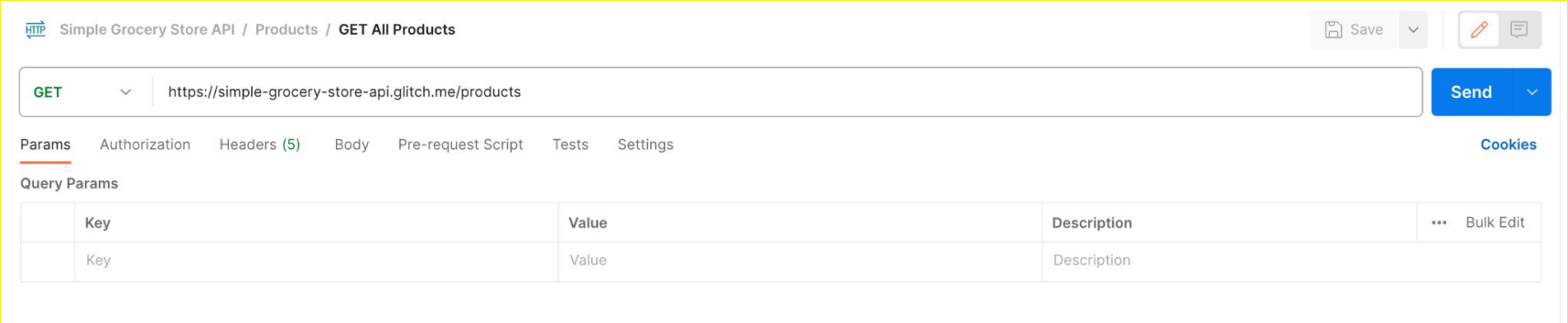
Simple Grocery Store API Documentation

WOMEN WHO CODE /london

Send your first request (2)

GET All Products

1. **Method - GET**
2. **Base URL/Endpoint - https://simple-grocery-store-api.glitch.me**
3. **Resource - /products**



Send your first request (3)

Create a New Cart:

1. **Method - POST**
2. **Base URL/Endpoint - https://simple-grocery-store-api.glitch.me**
3. **Resource - /carts**
4. **Request body:**

```
{
   "created": true,
   "cartId": "bx0-ycNjqIm5IvufuuZ09"
}
```

Send your first request (4)

Simple Grocery Store API / Carts / **Create a new cart**

Save

POST https://simple-grocery-store-api.glitch.me/carts

Params Authorization Headers (7) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "created": true,
3   "cartId": "bx0-ycNjqIm5IvufuuZ09"
4 }
```

Format of request body should be selected from dropdown.

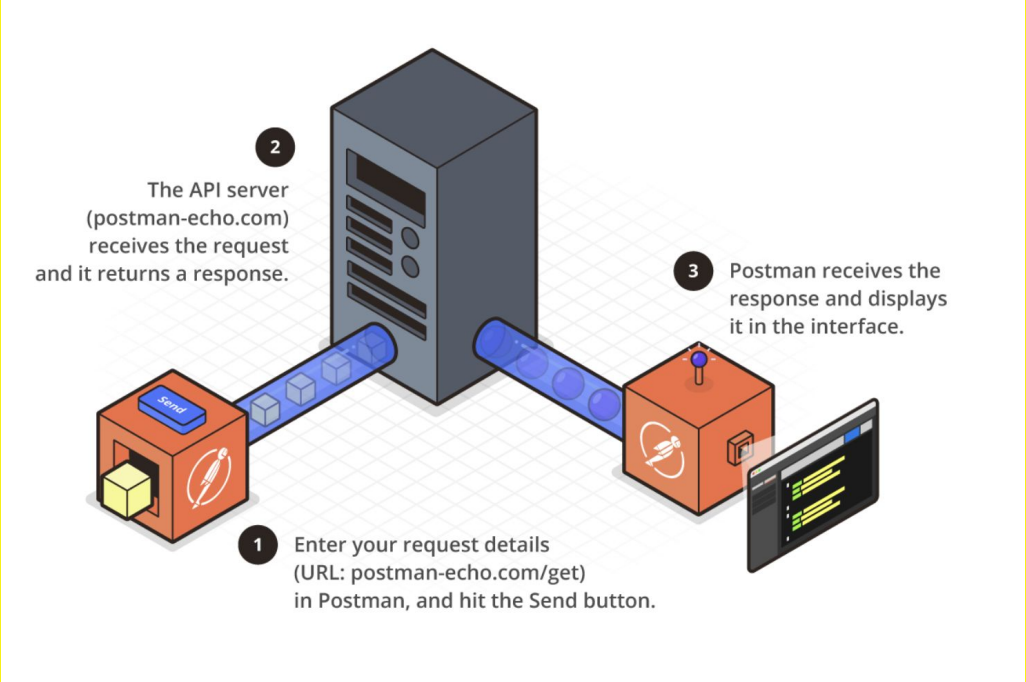
none form-data x-www-form-urlencoded raw binary GraphQL Text

```
1
```

- Text
- JavaScript
- JSON
- HTML
- XML

WOMEN WHO CODE /london

Sending requests in Postman



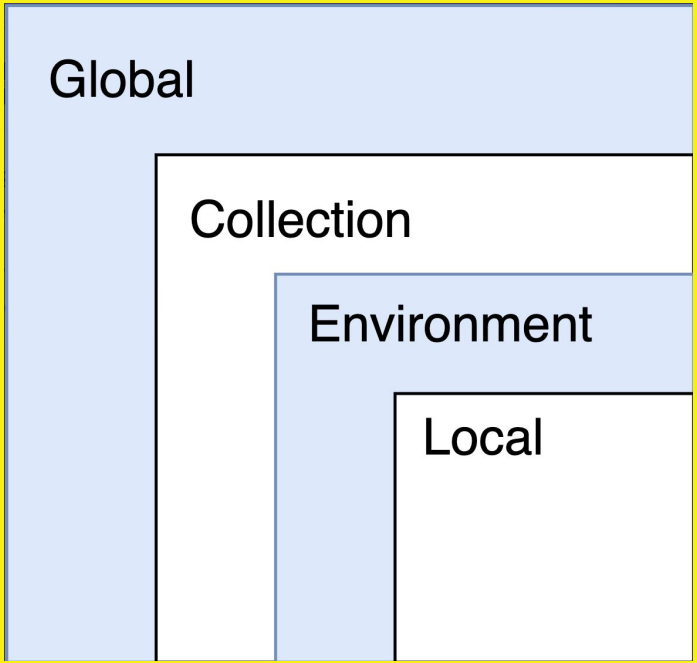
Source: https://learning.postman.com/docs/getting-started/sending-the-first-request/

03

Variables in Postman

Variables in Postman

Variables are the symbolic representation of the value which allows you to create, store and reuse values without typing it again in scripts.



Postman supports different types of scopes:

1. **Global enables you to access data between collections, requests, test scripts, and environments.**
2. **Collections are available throughout the requests in a collection and are independent of environments.**
3. **Environment variables enable you to scope your work to different environment, for example you can have different sets of values for production, staging and development environments.**
4. **Local variables are temporary variables that are accessed in your request scripts.**

WOMEN WHO CODE /london

How to set and get variables in Postman (1)

The syntax to set and get variables based on each scope is:

GLOBAL

SET: pm.global.set('variable_key', 'variable_value')
GET: pm.global.get('variable_key', 'variable_value')

COLLECTION

SET: pm.collectionVariables.set('variable_key', 'variable_value')
GET: pm.collectionVariables.get('variable_key', 'variable_value')

ENVIRONMENT

SET: pm.environment.set('variable_key', 'variable_value')
GET: pm.environment.get('variable_key', 'variable_value')

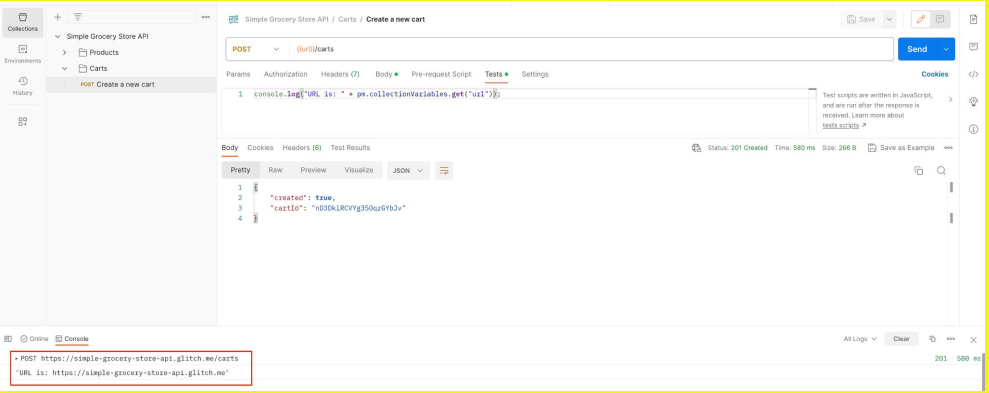
LOCAL

SET: pm.variables.set('variable_key', 'variable_value')
GET: pm.variables.get('variable_key', 'variable_value')

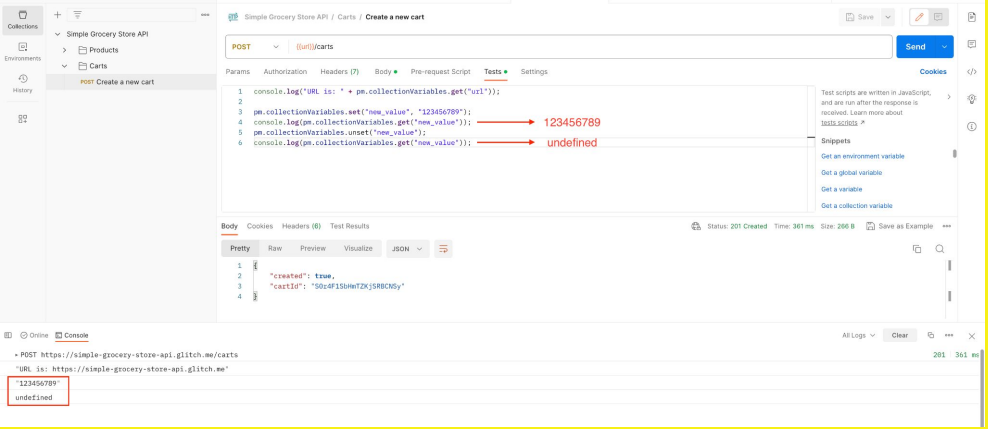
How to set and get variables in Postman (2)

Example in Postman:

1. Navigate to request and try to print value of collection variable “url” into console:



2. Set new collection variable “new_value”, print it, unset and print again



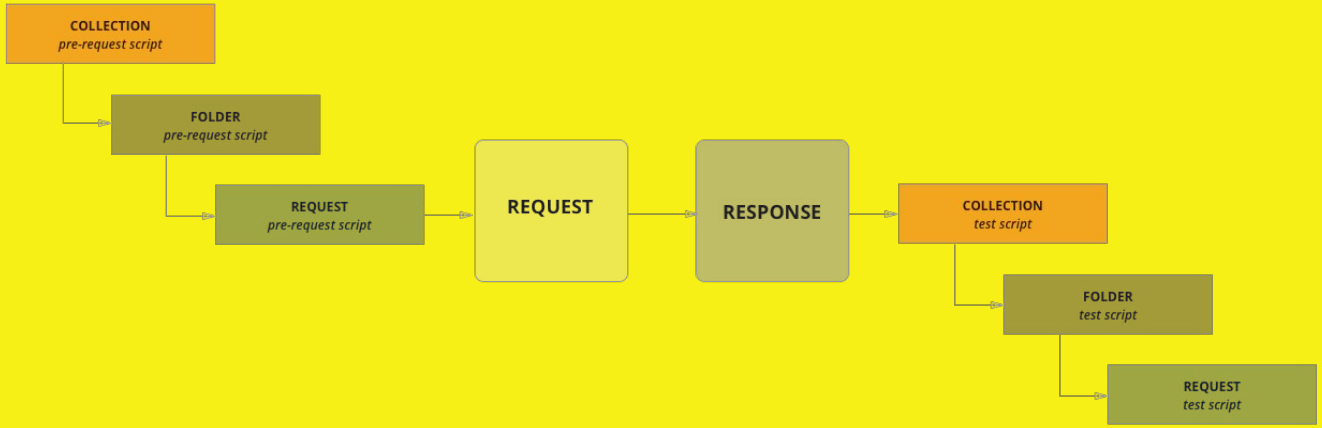
04

Scripting in Postman

Scripting in Postman

This is a correct order of executing scripts in Postman:

1. Pre-request script
2. Send the request
3. Getting response
4. Running tests



Source: https://learning.postman.com/docs/writing-scripts/intro-to-scripts/

Assertions in Postman (1)

Parsing API response

```
const responseJson = pm.response.json();
```

Testing response body

```
pm.test("Person is Jane", () => {
  const responseJson = pm.response.json();
  pm.expect(responseJson.name).to.eql("Jane");
  pm.expect(responseJson.age).to.eql(23);
});
```

Testing status codes

```
pm.test("Status code is 200", () => {
  pm.response.to.have.status(200);
});
```

Example of Response

```
{
   "name": "Jane",
   "age": 23,
}
```

WOMEN WHO CODE /london

Assertions in Postman (2)

Testing headers

```
pm.test("Content-Type header is present", ()
=> {
  pm.response.to.have.header("Content-Type");
});
```

Testing response time

```
pm.test("Response time is less than 200ms", ()
=> {

pm.expect(pm.response.responseTime).to.be.belo
w(200);
});
```

Example of Response

```
{
   "name": "Jane",
   "age": 23,
}
```

Assertions in Postman (3)

Validate JSON Schema

```
const schema = {
  "$schema":
"http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "name": {
      "type": "string"
    },
    "age": {
      "type": "integer"
    }
  },
  "required": [
    "name",
    "age"
  ]
}
pm.test('Schema is valid', function() {
  pm.response.to.have.jsonSchema(schema);
});
```

Example of Response

```
{
    "name": "Jane",
    "age": 23,
}
```

More information about assertions:
https://learning.postman.com/docs/writing-scripts/script-references/test-examples/

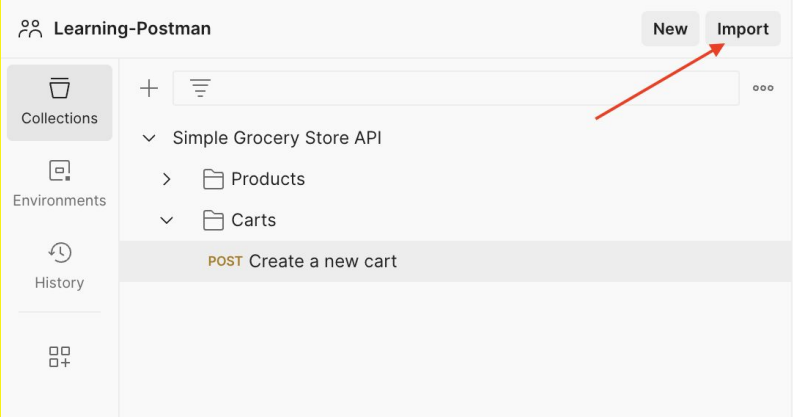
WOMEN WHO CODE /london

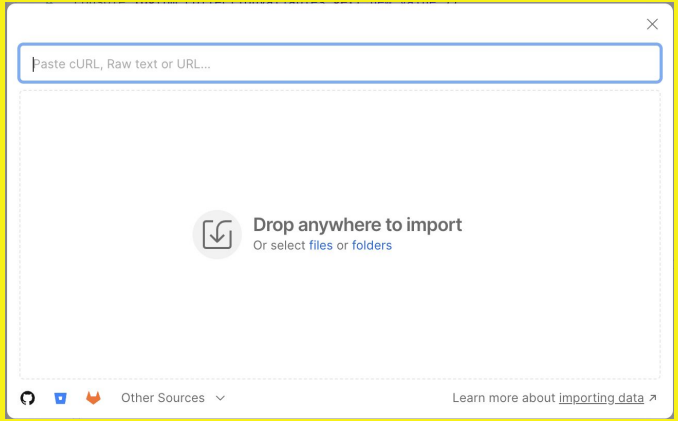
05

Importing and exporting Collections

Importing and exporting collections (1)

Importing

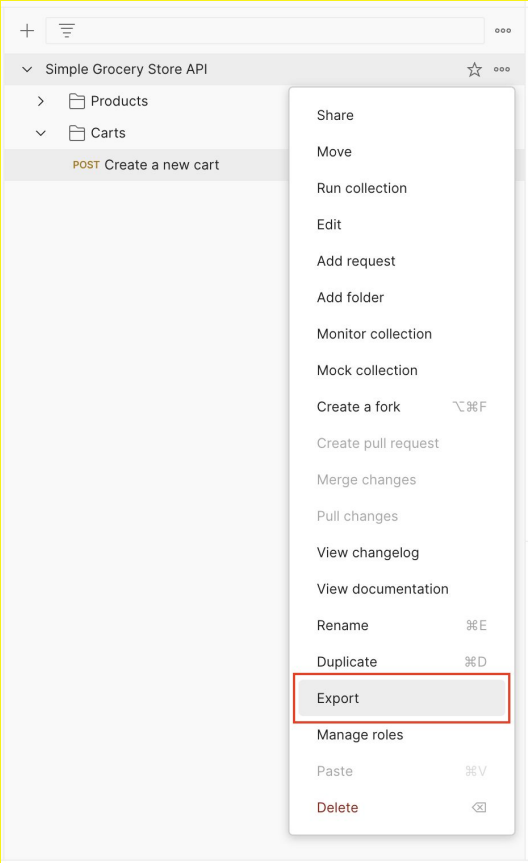




It is possible to import data from folders, files, using cURL, by dragging and dropping files or from code repositories by integrating them with Postman.

Importing and exporting collections (2)

Exporting



This collection will be exported as a JSON file.

WOMEN WHO CODE /london

06

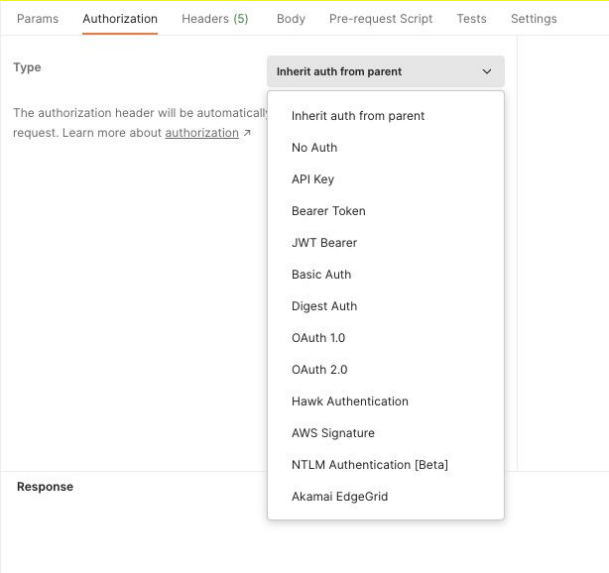
Authorization / Authentication

Authentication/Authorization (1)

If you start working with a third-party API, you might notice different types of authentication required.

You can see the "Authorization" section in three levels:

1. Collection,
2. Folder,
3. Request



There are several types of Authentication methods supported by Postman:

1. Basic Authentication
2. API Key
3. Bearer Token

WOMEN WHO CODE /london

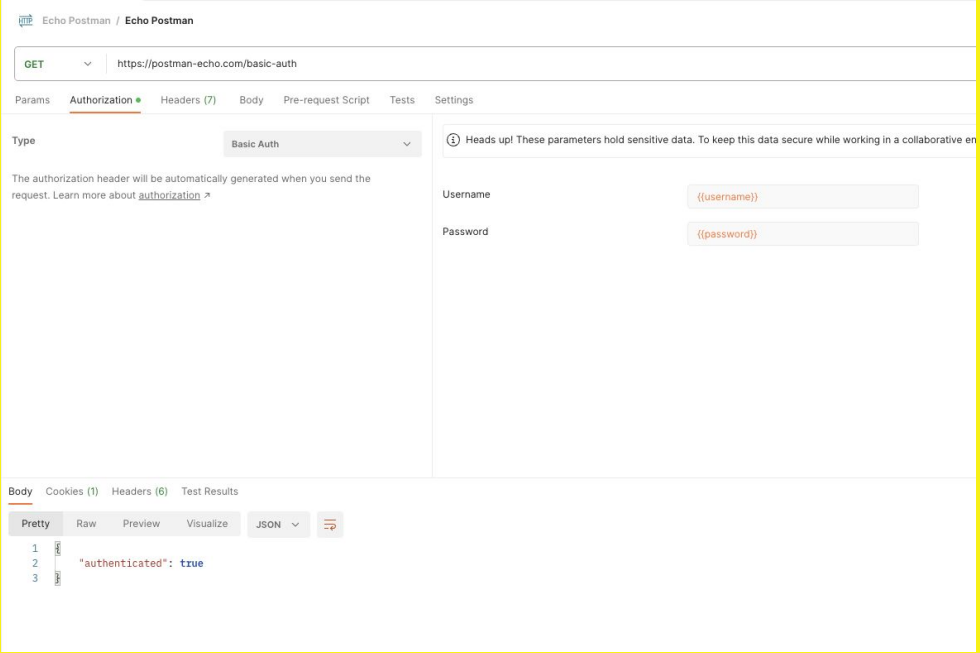
Authentication/Authorization (2)

Basic Access Authentication - simple method for implementing authentication where client provides username and password while making a request to a protected resource.

Basic username:password // where username:password pair encoded as Base64

EXAMPLE:

Method: GET
Endpoint:
https://postman-echo.com/basic-auth
Authorization Type: Basic Auth
Username: postman
Password: password
Save the details and click "**Send**"



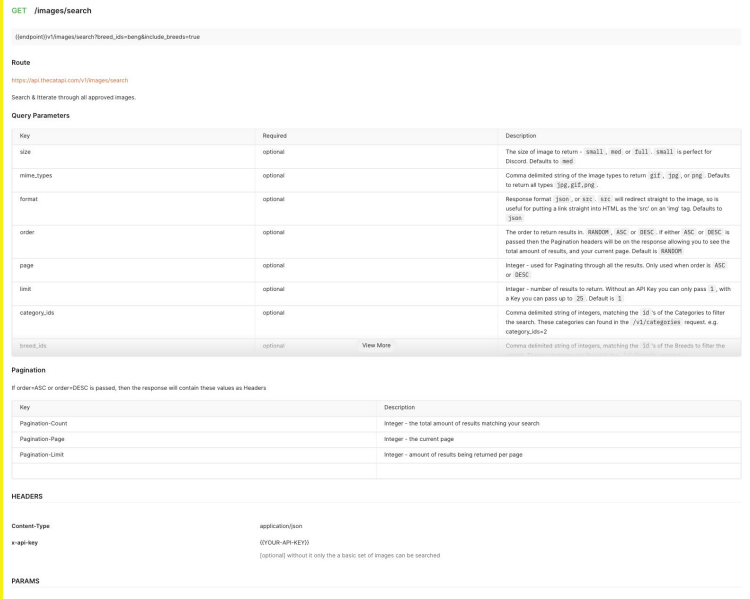
WOMEN WHO CODE /london

Authentication/Authorization (3)

API keys is a way to authenticate an application accessing the API, without referencing an actual user.
Some APIs use query parameters, some use the Authorize header, some use the body parameters. To clarify you have to look into documentation.

EXAMPLE:

1. For the demo we are going to use Dogs API and will take a look at Search Images Endpoint: https://documenter.getpostman.com/view/5578104/2s935hRnak#9e7e4cf9-0e0a-4258-8ace-ed1862843c96



GET /images/search

{{endpoint}}/v1/images/search?breed_ids=beng&include_breeds=true

Route

https://api.thecatapi.com/v1/images/search

Search & Itterate through all approved images.

Query Parameters

Key	Required	Description
size	optional	The size of image to return - `small` , `med` or `full` . `small` is perfect for Discord. Defaults to `med`
mime_types	optional	Comma delimited string of the image types to return `gif` , `jpg` , or `png` . Defaults to return all types `jpg,gif,png` .
format	optional	Response format `json` , or `src` . `src` will redirect straight to the image, so is useful for putting a link straight into HTML as the 'src' on an 'img' tag. Defaults to `json`
order	optional	The order to return results in. `RANDOM` , `ASC` or `DESC` . If either `ASC` or `DESC` is passed then the Pagination headers will be on the response allowing you to see the total amount of results, and your current page. Default is `RANDOM`
page	optional	Integer - used for Paginating through all the results. Only used when order is `ASC` or `DESC`
limit	optional	Integer - number of results to return. Without an API Key you can only pass `1` , with a Key you can pass up to `25` . Default is `1`
category_ids	optional	Comma delimited string of integers, matching the `id` 's of the Categories to filter the search. These categories can found in the `/v1/categories` request. e.g. category_ids=2
breed_ids	optional	Comma delimited string of integers, matching the `id` 's of the Breeds to filter the

View More

Pagination

If order=ASC or order=DESC is passed, then the response will contain these values as Headers

Key	Description
Pagination-Count	Integer - the total amount of results matching your search
Pagination-Page	Integer - the current page
Pagination-Limit	Integer - amount of results being returned per page

HEADERS

Content-Type	application/json
x-api-key	{{YOUR-API-KEY}} [optional] without it only the a basic set of images can be searched

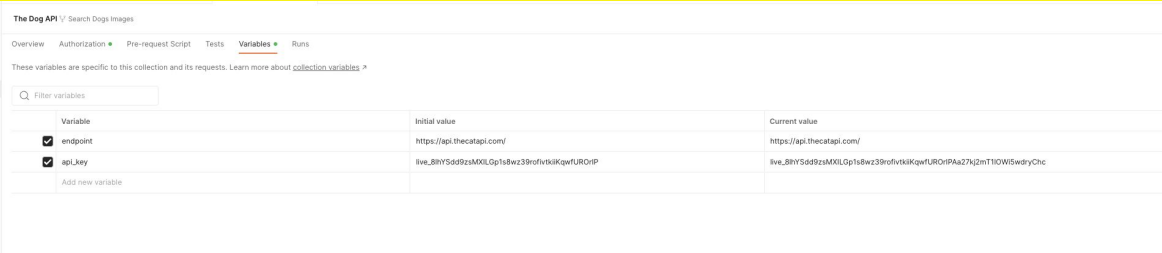
PARAMS

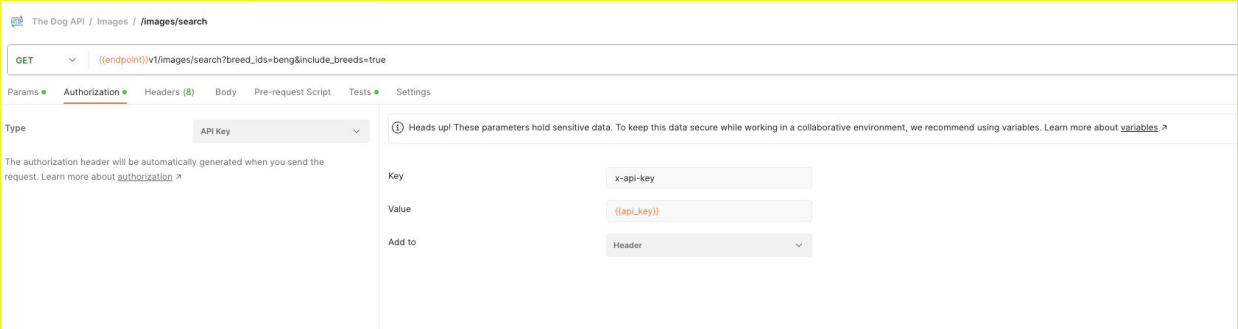
WOMEN WHO CODE /london

Authentication/Authorization (4)

EXAMPLE (continue):

2. Register on the website for free and retrieve the key from the confirmation email.
3. Navigate to the request to collection variables, create a new collection variable “api_key” and provide the value into the “Current value” column.





4. Navigate to the Authorization tab and select API Key method. Provide name of the collection variable, hit “Save”
5. Send the request

WOMEN WHO CODE /london

Authentication/Authorization (5)

Bearer tokens enable requests to authenticate using an access key, such as a JSON Web Token (JWT). The token is a text string, included in the request header.

Postman will append the token value to the text Bearer in the required format to the request Authorization header as follows:

Bearer “YOUR_API_KEY”

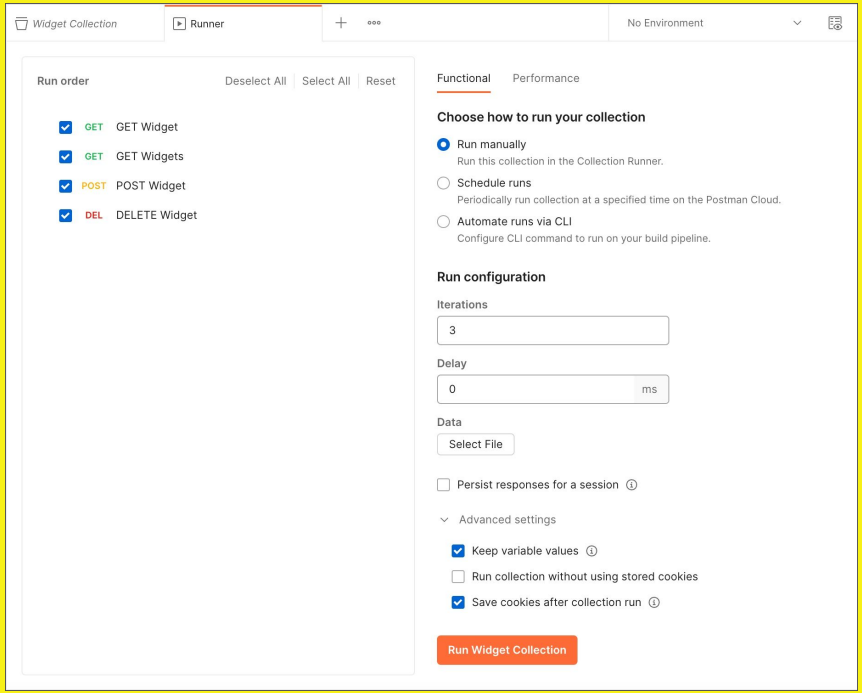
COLLECTION RUNNER

The Collection Runner enables you to run a collection's requests in a specified sequence to test the functionality of your API. It logs your request test results and can use scripts to pass data between requests and alter the request workflow.

1. Select Collections in the sidebar and select the collection you want to run.
2. Select **Run**
3. Choose any configuration options:

- *Iterations* - The number of iterations for your collection run. You can also run collections multiple times with different data sets to build workflows.
- *Delay* - An interval delay in milliseconds between each request.
- *Data* - A data file for the collection run.
- *Persist responses for a session* - Log the response headers and bodies so you can review them after running the collection. For large collections, persisting responses may affect performance.

4. When you've completed your configuration, select Run (collection name).



Thank you

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

WOMEN WHO CODE®
/london