Quick-Introduction-to-Postman-and-API-Testing-for-Beginners

## Introduction

### Course Materials

Github tool rental repo

<https://github.com/vdespa/quick-introduction-to-postman/blob/main/simple-tool-rental-api.md>

Discord server

<https://discord.com/invite/FuDvgVzcDy>

### How to install

<https://www.postman.com/downloads/>

### Release notes

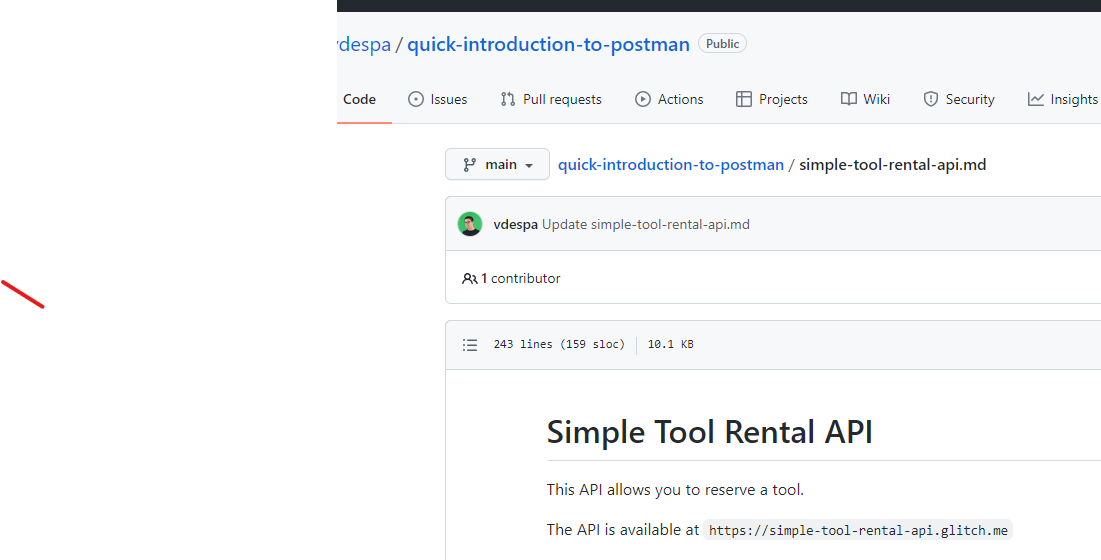
<https://www.postman.com/downloads/release-notes/>

### A quick intro to APIs

We’re hear talking about web apis

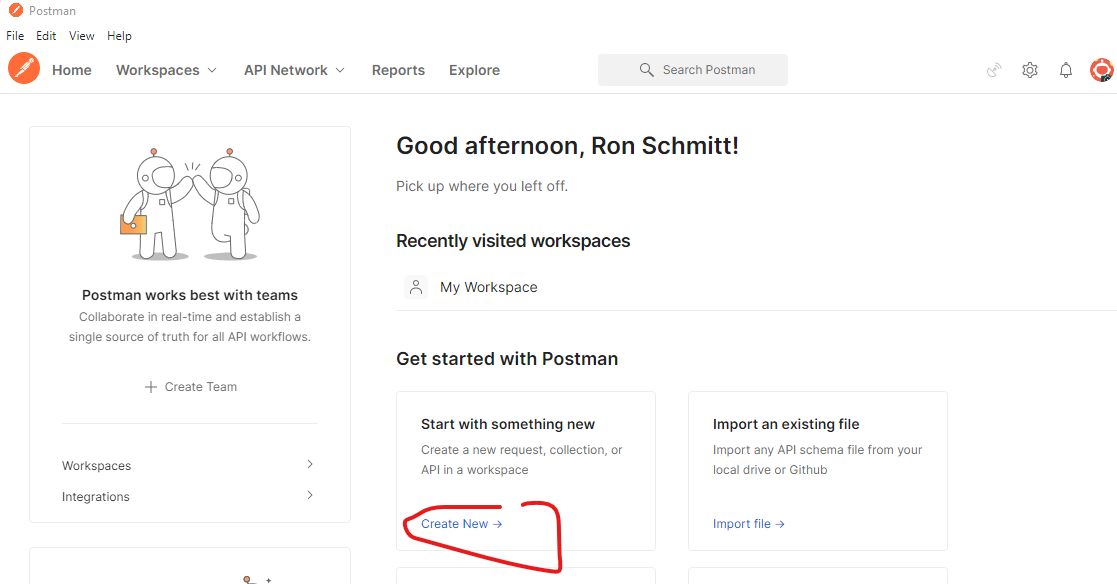
### Your first Postman request

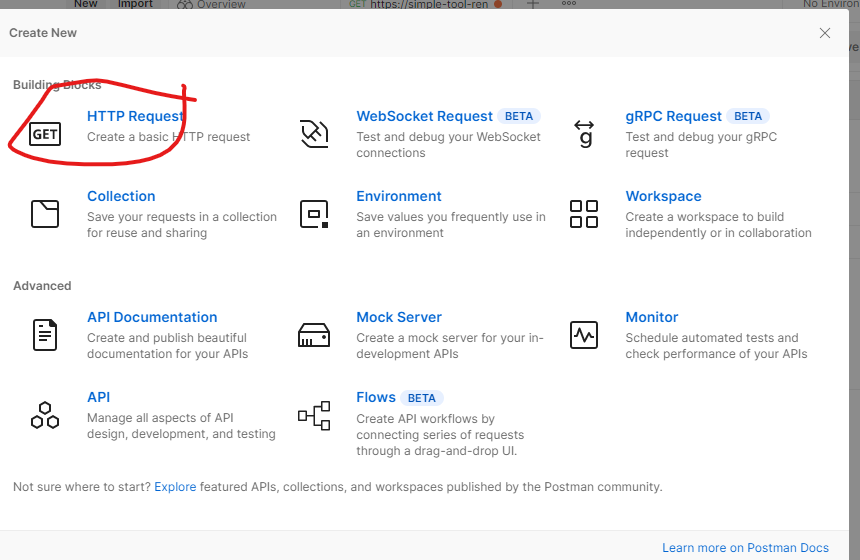
Grab the address form the git repo



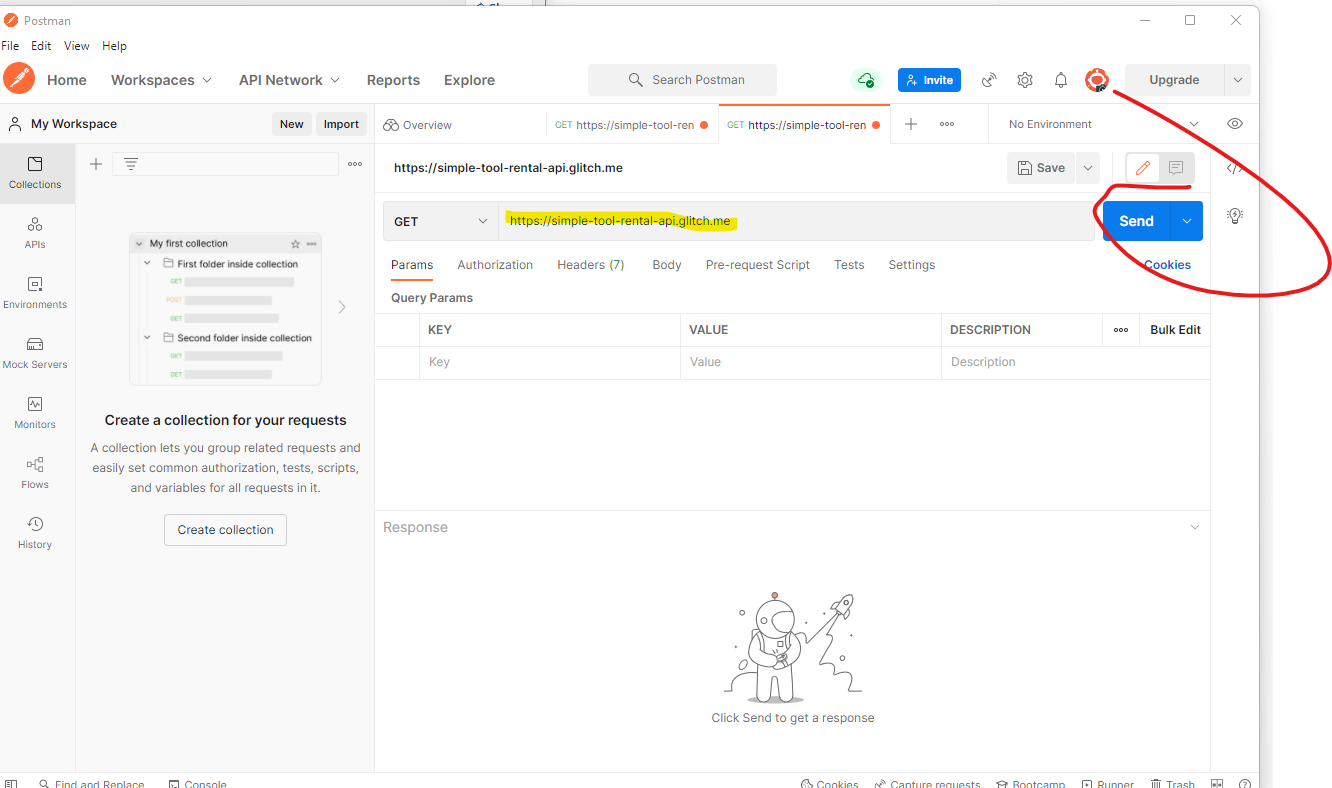
<https://simple-tool-rental-api.glitch.me>

open postman

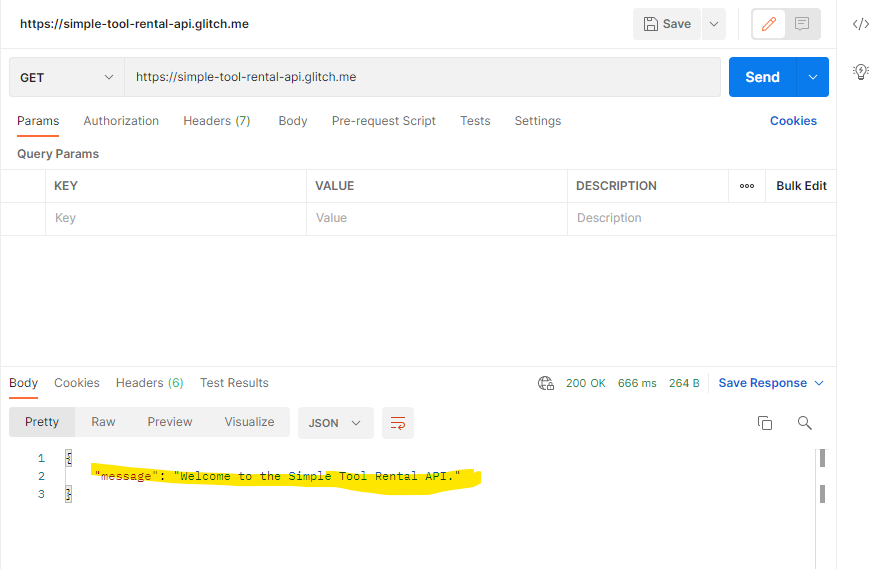




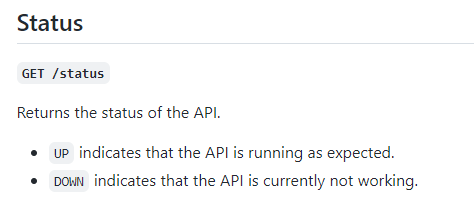
Paste the address in. Always make sure not to paste extra spaces or carriage returns.



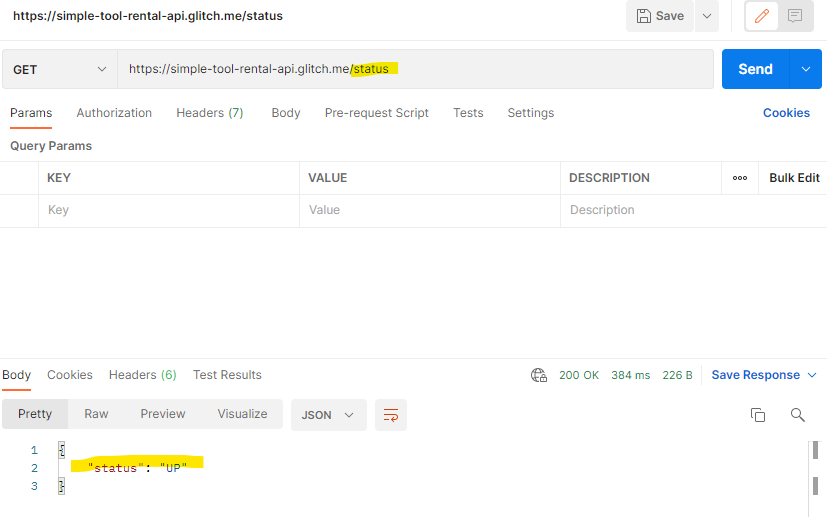
Result



Back to course website, the first type of message is **status**



Now append status to the address



### The HTTP protocol explained

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages>

#### Request

##### Start line

HTTP requests are messages sent by the client to initiate an action on the server. Their start-line contain three elements:

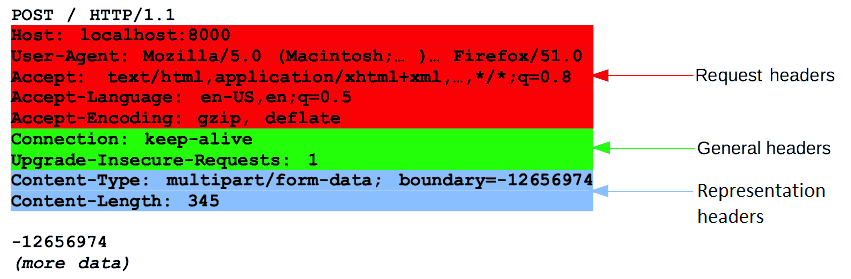
1. An [*HTTP method*](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods), a verb (like [GET](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/GET), [PUT](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/PUT) or [POST](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST)) or a noun (like [HEAD](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/HEAD) or [OPTIONS](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS)), that describes the action to be performed. For example, GET indicates that a resource should be fetched or POST means that data is pushed to the server (creating or modifying a resource, or generating a temporary document to send back).
2. The request target, usually a [URL](https://developer.mozilla.org/en-US/docs/Glossary/URL), or the absolute path of the protocol, port, and domain are usually characterized by the request context. The format of this request target varies between different HTTP methods. It can be
   * An absolute path, ultimately followed by a '?' and query string. This is the most common form, known as the origin form, and is used with GET, POST, HEAD, and OPTIONS methods.
     + POST / HTTP/1.1
     + GET /background.png HTTP/1.0
     + HEAD /test.html?query=alibaba HTTP/1.1
     + OPTIONS /anypage.html HTTP/1.0
   * A complete URL, known as the absolute form, is mostly used with GET when connected to a proxy. GET https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages HTTP/1.1
   * The authority component of a URL, consisting of the domain name and optionally the port (prefixed by a ':'), is called the authority form. It is only used with CONNECT when setting up an HTTP tunnel. CONNECT developer.mozilla.org:80 HTTP/1.1
   * The asterisk form, a simple asterisk ('\*') is used with OPTIONS, representing the server as a whole. OPTIONS \* HTTP/1.1
3. The HTTP version, which defines the structure of the remaining message, acting as an indicator of the expected version to use for the response.

##### Headers

[HTTP headers](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers) from a request follow the same basic structure of an HTTP header: a case-insensitive string followed by a colon (':') and a value whose structure depends upon the header. The whole header, including the value, consist of one single line, which can be quite long.

Many different headers can appear in requests. They can be divided in several groups:

* [General headers](https://developer.mozilla.org/en-US/docs/Glossary/General_header), like [Via](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Via), apply to the message as a whole.
* [Request headers](https://developer.mozilla.org/en-US/docs/Glossary/Request_header), like [User-Agent](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent) or [Accept](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept), modify the request by specifying it further (like [Accept-Language](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept-Language)), by giving context (like [Referer](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Referer)), or by conditionally restricting it (like [If-None](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/If-None)).
* [Representation headers](https://developer.mozilla.org/en-US/docs/Glossary/Representation_header) like [Content-Type](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type) that describe the original format of the message data and any encoding applied (only present if the message has a body).



##### Body

The final part of the request is its body. Not all requests have one: requests fetching resources, like GET, HEAD, DELETE, or OPTIONS, usually don't need one. Some requests send data to the server in order to update it: as often the case with POST requests (containing HTML form data).

Bodies can be broadly divided into two categories:

* Single-resource bodies, consisting of one single file, defined by the two headers: [Content-Type](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type) and [Content-Length](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Length).
* [Multiple-resource bodies](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types#multipartform-data), consisting of a multipart body, each containing a different bit of information. This is typically associated with [HTML Forms](https://developer.mozilla.org/en-US/docs/Learn/Forms).

#### Response

##### Status line

The start line of an HTTP response, called the *status line*, contains the following information:

1. The *protocol version*, usually HTTP/1.1.
2. A *status code*, indicating success or failure of the request. Common status codes are [200](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/200), [404](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/404), or [302](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/302)
3. A *status text*. A brief, purely informational, textual description of the status code to help a human understand the HTTP message.

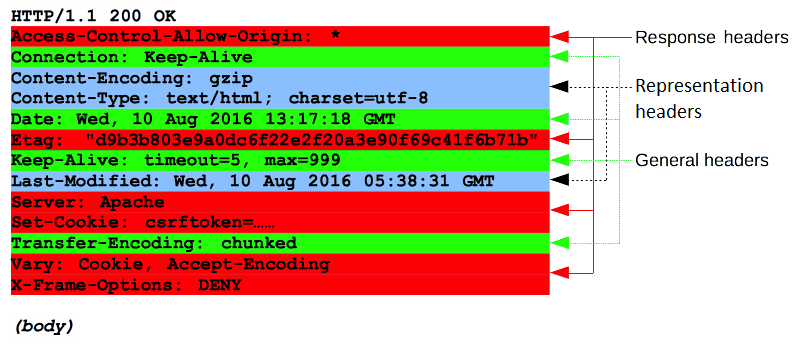
A typical status line looks like: HTTP/1.1 404 Not Found.

##### Headers

[HTTP headers](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers) for responses follow the same structure as any other header: a case-insensitive string followed by a colon (':') and a value whose structure depends upon the type of the header. The whole header, including its value, presents as a single line.

Many different headers can appear in responses. These can be divided into several groups:

* [General headers](https://developer.mozilla.org/en-US/docs/Glossary/General_header), like [Via](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Via), apply to the whole message.
* [Response headers](https://developer.mozilla.org/en-US/docs/Glossary/Response_header), like [Vary](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Vary) and [Accept-Ranges](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Accept-Ranges), give additional information about the server which doesn't fit in the status line.
* [Representation headers](https://developer.mozilla.org/en-US/docs/Glossary/Representation_header) like [Content-Type](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type) that describe the original format of the message data and any encoding applied (only present if the message has a body).



##### Body

The last part of a response is the body. Not all responses have one: responses with a status code that sufficiently answers the request without the need for corresponding payload (like [201](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/201) **Created** or [204](https://developer.mozilla.org/en-US/docs/Web/HTTP/Status/204) **No Content**) usually don't.

Bodies can be broadly divided into three categories:

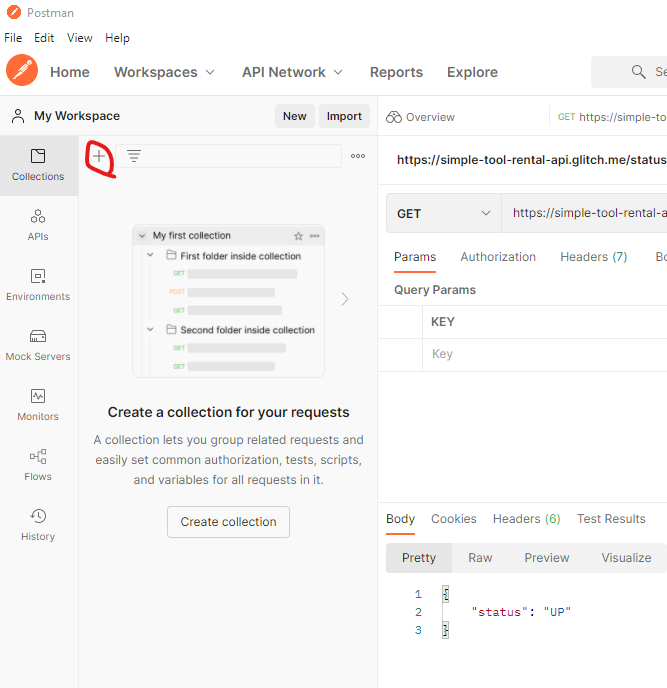
* Single-resource bodies, consisting of a single file of known length, defined by the two headers: [Content-Type](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Type) and [Content-Length](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Length).
* Single-resource bodies, consisting of a single file of unknown length, encoded by chunks with [Transfer-Encoding](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Transfer-Encoding) set to chunked.
* [Multiple-resource bodies](https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types#multipartform-data), consisting of a multipart body, each containing a different section of information. These are relatively rare

## Creating requests in Postman

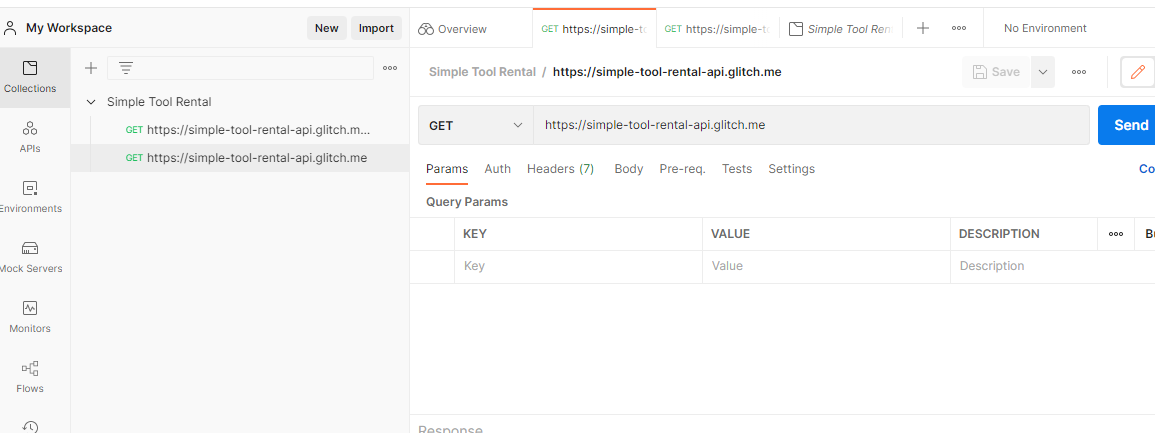
### Postman collections

Collection is basically a folder for

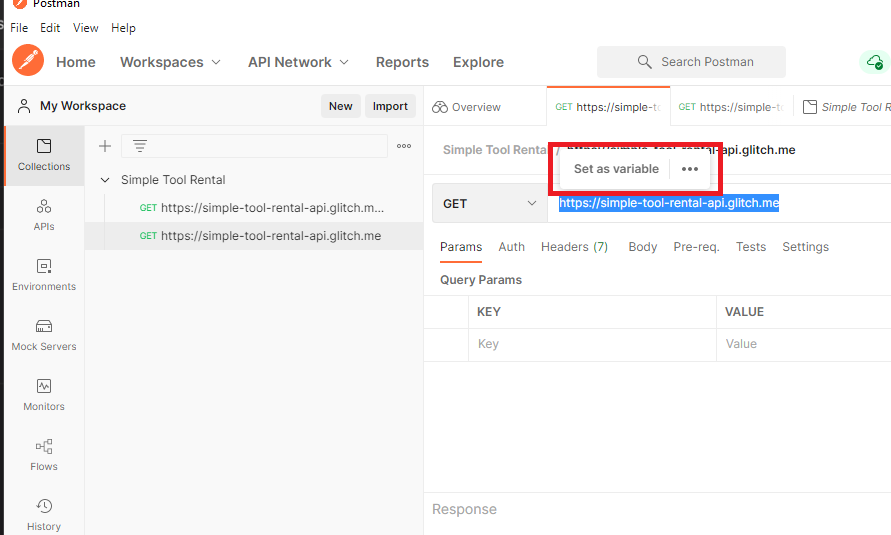
Create a collection



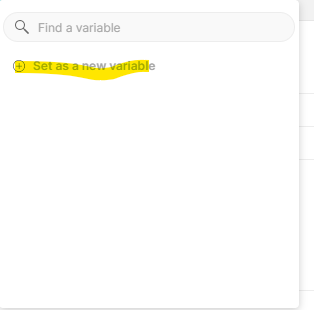
Name it and save the two requests that we have created into it. Result:

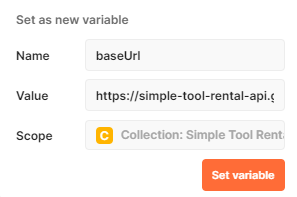


Create a new variable called **baseUrl** by selecting the address in the request and wait for menu to pop up

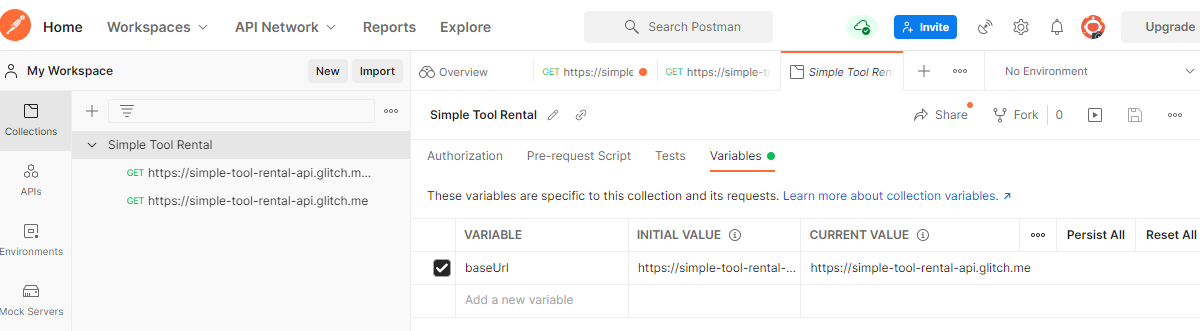
v

**Select Set as variable** then **Set as new variable** and set scope to be the collection



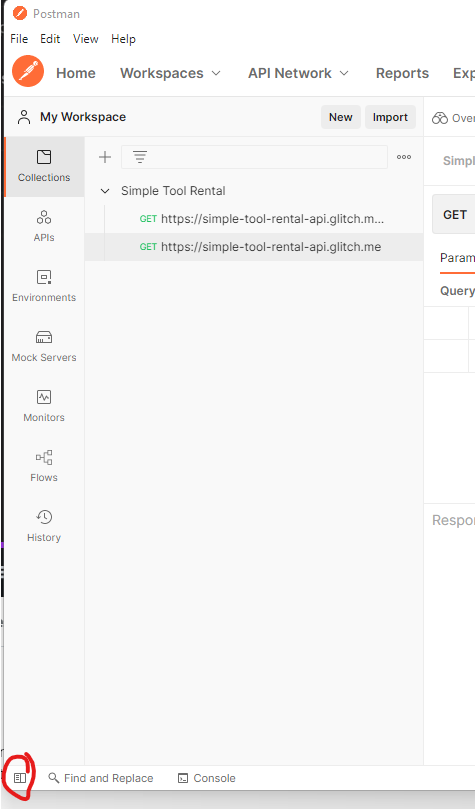


Here are the variables. These are private to your set up. When sharing the value at the time of sharing is shared with them, but you can change your variables independently



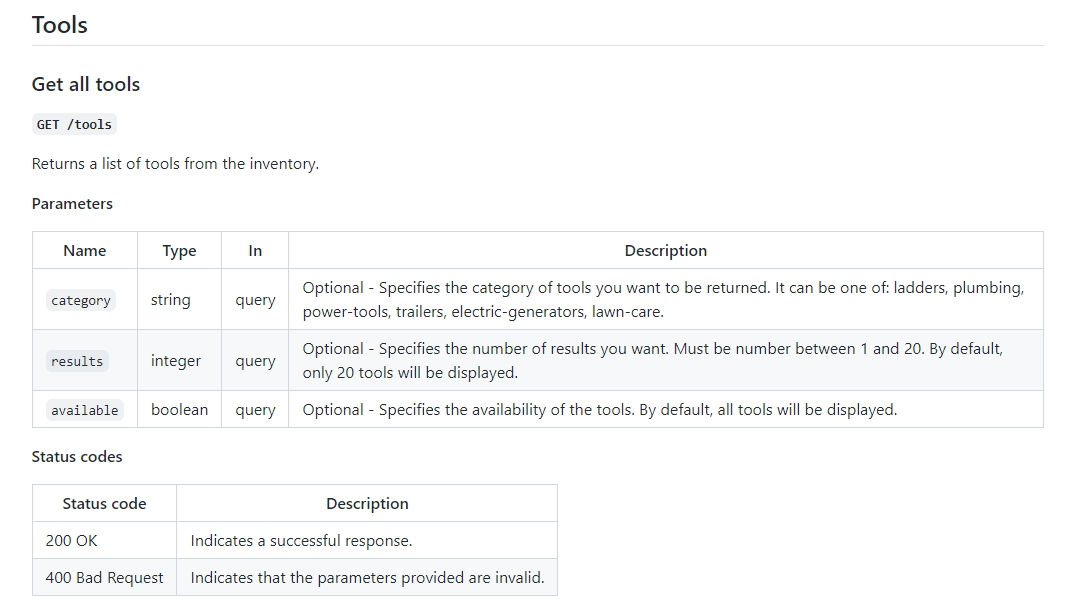
BE CAREFUL ABOUT SHARING A COLLECTION WITH PASSWORDS AS VARIABLES

Button at lower left is to show/hide sidebar



### Query parameters

Back to the course website, the next category of messages is **tools**



### Assignment#1 – Query parameters

### Assignment#1 – Solution

### Path variables

### Query parameters and path variables

### API Authentication

### Troubleshooting HTTP status code

### HTTP headers

### JSON format explained

### Assignment#2

### Assignment#2-Solution

### GET vs POST

### Using random data in requests

### How not to use Postman

### Assignment#3 – Adding more requests

### Assignment#3-Solution

### PATCH request method

### DELETE request method

## Preparing for automation