**Postman**

* Postman is a tool that allows us to easily work with APIs.
* Postman is used to build HTTP requests that we send to the server running the API.

Postman.com- <https://sanjanaravi2003-4754094.postman.co/workspace/Sanju's-Workspace~a32385a5-d141-4d7f-83d8-ff05858fda08/request/48485697-2b7033e8-72dd-48b1-84db-62e98dd05fa4>

**1.Your first request with Postman**

* To use an API you need to read the API documentation. We're using Simple Books API whose documentation can be found in the resources section of this lesson below.
* Work in Postman is organized in Workspaces.
* A status code 200 indicates that the request was successful.

**HTTP**

* The API we are using uses the HTTPS protocol.
* HTTPS stands for Secure Hypertext Transfer Protocol.
* HTTPS ensures that the connection is encrypted.
* All APIs should use HTTPS.
* From our point of view HTTP and HTTPS are the same.
* The HTTP request message will contain:
  + URL (address)
  + Request method (GET, POST, PUT, ...)
  + Headers (User-Agent: Postman)
  + Body
* The HTTP response message will contain:
  + Status code (200, 404, 500, ...)
  + Headers
  + Body

**Postman collections and variables**

* You can save requests so that you can re-use them later on.
* All requests need to be added to a Postman collection.
* Typically you will have a Postman collection for each API.
* We are storing the base address of the API in a collection variable called baseUrl.
* Our saved baseUrl will be displayed as {{baseUrl}} in the address bar.
* Variables allow us to avoid repeating the same information.
* Variables allow us to easily make changes.
* A Postman variable has two states
  + INITIAL VALUE - This will be available to others if you share the collection.
  + CURRENT VALUE - This is private to you and this is the value that Postman uses.

**Query parameters**

* JSON is the most popular format that APIs use to send data.
* Query parameters start after the ? in the URL.
  + example : {{baseUrl}}/books?type=fiction
* The format is key=value
* Muliple query parameters are delimited in the URL with an &.
  + example: foo=1&bar=2
* Depending on the API, some query parameters can be optional or mandatory.
* A response status 400 indicates an issue with the request data.
* You can enable and disable parameters by clicking the checkbox associated with each key-value pair.

**Path variables**

* :bookId is a path variable in the URL.
* This endpoint allows us to specify a value that changes all the time, depending on the book.
* :bookId is just a placeholder and does not get sent.
* You can use path variables in combination with query parameters (if the API accepts this).

**POST request / API Authentication**

* A POST request allows you to send data in the request body.
* The endpoint for submitting orders requires authentication.
* Some APIs/endpoints are public and require no authentication.
* Other APIs/endpoints are private and require authentication.
* An access token is temporary password generated by the API.
* To send JSON, select the POST request method and from the Body select Raw and from the list JSON.

**JSON format**

* You need to specify valid JSON, otherwise the server won't understand your request.
* Use double-quotes "" for strings, separate key-value pairs with a comma sign ,
* Numbers, booleans don't need to be between quotes.
* Postman will indicate when your JSON is invalid.

**Random test data**

* You can use a special type of Postman variable to generate random data
  + example: {{$randomFullName}}
* To inspect the request body you can use the Postman console.

**Is Postman the right tool for me?**

* Postman is a tool for dealing with APIs.
* Postman cannot work with User Interfaces, click buttons and fill out forms.
* Postman is not a performance testing tool.
* Postman can be used for security testing but has not been designed for this purpose.

**PATCH request**

* A PATCH request is typically used for updating existing data.
* A PATCH usually does a partial update, by changing only some of the properties.

**DELETE request**

* A DELETE request is used for deleting data.
* If you try to get the same data with a GET request, you will get a 404 Not Found status code.

**Test automation with Postman**

**Your first API tests**

* We are looking at the response to understand if the API is working properly.
* With API tests we want to avoid manually re-testing the API.
* Tests in Postman are written in JavaScript.
* Tests are executed ONLY after the response has arrived from the API.
* Postman uses an assertion library called Chai.js
* Testing the response status code is one of the easiest tests you can write.
* When writing tests, we want to make sure the tests fail.
* To make the assertions on a JSON response, you first need to parse it.
* To see the contents of a JavaScript variable you can use console.log()
* To get a property of an object, you can use this syntax: someobject.someproperty
  + alternative syntax: someobject["someproperty"]

**Postman variables**

* Postman variables are fundamental to automating testing of the API.
* Postman environments (environment variables) are good if you have multiple testing environments (localhost, testing, production)
* Postman collection variables are saved in the collection.
* Postman global variables are available to all collection in a workspace.
* We use Postman global variables as the data we save is not that important after the execution has stopped.

**Extracting data from the response**

* Having hardcoded values in requests can make the API tests fail if the data changes.
* We are using the filter function available on all arrays to remove the books that are not available.
* Always use console.log() to view the data you are trying to set as a variable.

**Collection runner**

* The Collection runner is a built-in functionality of Postman.
* The Collection runner allows you to execute the entire collection with just one click.
* Make sure to check (:white\_check\_mark:) the "Save response" box as this will allow you to inspect the response body.

**Request execution order**

* If you run a Postman collection, the default order is as you have it in the collection.
* In the video, postman.setNextRequest() is used to change the order of execution. However, this method is now deprecated.
* The correct way to set the next request is by using pm.execution.setNextRequest() instead.
* If you wish to stop the execution prematurely, you can do so by running pm.execution.setNextRequest(null)

**Postman monitors**

* Creating a Postman monitor ensures that you can run a Postman collection according to a pre-defined schedule.
* Running the collection will be handled by Postman on their infrastructure, you don't need to keep Postman open.
* If you are not familiar with continuous integration servers like Jenkins, GitLab CI or TeamCity, this is a quick and easy way to access a Postman collection.
* The API needs to be accessible from any network.

**Newman**

* Newman is a CLI tool that can take a Postman collection, run all the tests and generate a report at the end.
* Newman does not have an interface, you need to work with it from the terminal.
* Often Newman is installed on an integration server like Jenkins, GitLab CI or TeamCity.
* To run Newman on your computer, you need to have Node.js installed.
  + you can download Node.js from <https://nodejs.org/> (download the LTS version)
* To install newman, run the command: npm install -g newman
* Check if newman is install with: newman --version
* There are three ways to access a collection from Newman:
  + Export the collection as a JSON file.
  + Share with a public link.
  + Use the Postman API to get the collection.

Newman is particularly useful when you integrate it with a CI server