Quick Introduction to Postman and API

Course notes

Version: November 2021

Introduction

A quick introduction to APIs

- API stands for Application Programming Interface.
- The last word, interface, is the most important one to understand.
- An API is an interface to some data on a server, typically stored in a database. This interface allows a program to communicate and thus exchange data with that server. Without this interface, the server would be inaccessible to the outside world.
- In this course we talk about web APIs, which work over the internet. It is no wonder you are now learning about APIs. They are being used everywhere.
- an API is an interface to some data stored somewhere remotely, on a different server.
- Postman can connect to a server through an API and exchange data.

Your notes

How to install Postman

- Postman uses a freemium pricing model, and for many use-cases, including this course, it is free to use.
- There are two ways of running Postman:
    - in your browser by going to postman.com
    - as a standalone app that you need to install on your computer (Postman is available for Windows, macOS, or Linux)
- DO NOT USE the deprecated Google Chrome extension
- Go to postman.com and create an account.

Resources

- Download Postman
  https://www.postman.com/downloads/
- Postman installation guide
  https://learning.postman.com/docs/getting-started/installation-and-updates/

Your notes

Your first Postman request

- Through the course, we will be using a simple API of a tool rental store which allows us to view tools & place an order
- To know how to use the API, we need to study the API documentation
- Not possible to know how to use an API without some kind of documentation. It is like if you buy a complex machine but get no instruction on how to use it.
- **Tip:** Copy/pasting URLs/params/data from the API documentation ensures you make fewer mistakes.
- **Heads-up!** Make sure you don't add any newlines or spaces when pasting text in Postman

Resources

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

Your notes

The HTTP protocol explained

- Postman is the client and has sent a message to the server running the tool rental API
- We call this message a request, as it is requesting some data.
- The message that the server sends in response is called a response
- To make this communication possible, we have used HTTP
- HTTP is a protocol that enables the client, in this case, Postman, and the server which runs the API to communicate.
- A protocol is essentially a set of rules that both parties need to follow.
- **Heads-up!** It is essential to understand some basics around HTTP to be able to read any API documentation and use APIs.
- HTTPS is the secure & encrypted version of HTTP; use HTTPS whenever possible.

The HTTP request has:

- Request method (sometimes called HTTP verb)
- Address / URL
- Headers
- Body

The HTTP response has:

- Status code
- Headers
- Body

Your notes

Creating requests in Postman

Postman collections

- To save a request, click on save and create or select a collection
- In Postman, we try to avoid having configurations in your requests, just in case something changes.
- Save the baseUrl in a Postman collection variable
    - Works only if you have saved the request in a collection
- The variable name is between curly brackets {{baseUrl}}
- Initial value
    - This is used by Postman when submitting a request
    - Private to you / your Postman account
- Current value
    - Exposed to others when sharing the collection
    - Not used by Postman

Your notes

Query parameters

- **Heads-up!** Postman collection variables are unresolved if the request is not saved in the same collection where the variable is defined.
- Response body: this way of formatting data is called JSON
- Query parameters are a way to send data to the API
- For this API, query parameters in this case are a way to filter data, to get a subset of the data
- Query parameters can be optional or mandatory (as specified by the API documentation)
- Which query parameters are available can only be know by reading the APi
- Mistake that every beginners does: Category vs category
- **Tip**: Always copy / paste names from the documentation to avoid making mistakes
- What are [ ] empty brackets: this is not an error, it is just an empty list

Your notes

Assignment #1

- Add the results parameter to the request
- Try different values
- If you are learning about APIs for work for quality assurance purpose, this is a very important activity
- Can you find any bugs?

Your observations

Path variables

- Path parameters are required if the endpoint mentions them
- The notation for path variables is :variableName (don’t forget the colon!)
- Path variables in Postman are just a placeholder
- The name of the variable is NOT being sent with the request
- You can have both query parameters and path parameters
    - No question mark
- **Tip**: use the Postman console to inspect the requests and responses

Your notes

Query parameters and path variables

Path parameters

- Mandatory
- Sends data to the APIs
- Part of the endpoint / path

Query

- Mandatory or optional
- Sends data to the APIs
- Start after the question mark ?

Your notes

API Authentication

- API can be public or private
- Private API require authentication
- Even public API may require authentication when creating new data or updating existing one
- The purpose of the API client registration is to obtain an access token, which is like a password
- The term client does not refer to a customer (think about client-server)
- For us, the API client is Postman
- when working with APIs, we will not get a login form where we can enter a username and password
- We use tokens which are like a temporary password
- Tokens are usually added to headers or as query parameters (see the API documentation)

Your notes

Troubleshooting HTTP status codes

- Typical errors
    - 404 - check the URL or the HTTP request method
    - 400 - check your request body, ensure that JSON is valid
    - 409 - client registered

Your notes

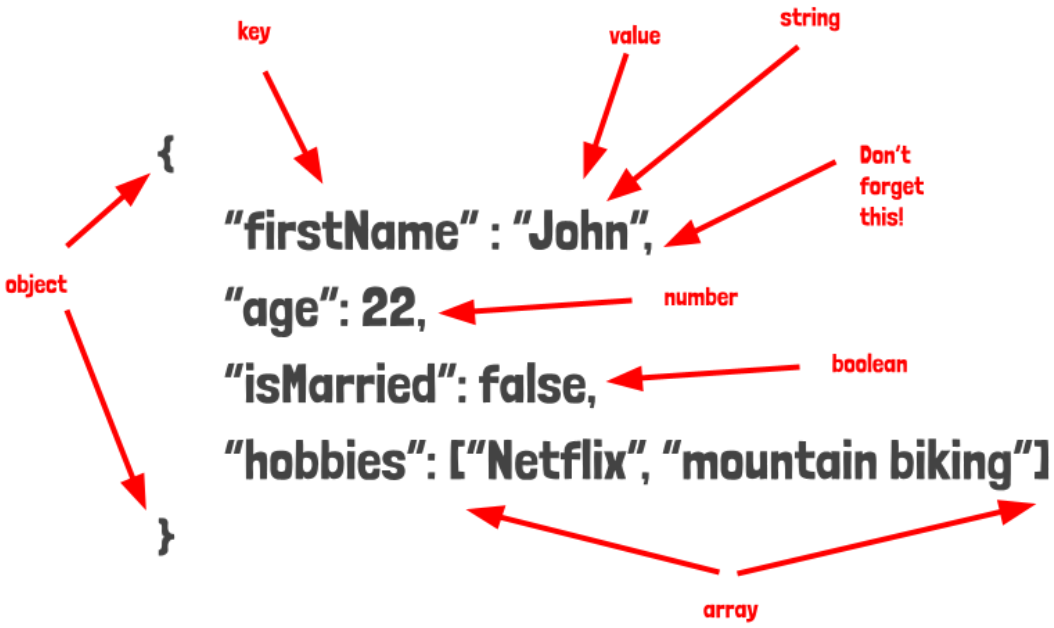
HTTP headers

- HTTP Headers are found in the request and in the response
- Typical request headers
    - Content Type - it is telling the API that the request body is in JSON format
    - Authorization - contains authentication information
- Typical response headers
    - Content Type - it is telling the client (Postman) that the response body is in JSON format

Your notes

JSON format explained

- In practical terms, we use JSON to transfer data from one machine to the other
- JSON has a simple key-value format
- The advantage of JSON is that it is both readable by humans as well as by computers.
- { } - curly brackets denote an object
- [ ] - square brackets denote a list (an array of elements)



- Make sure you write valid JSON, otherwise the API won't understand you.
- Typical JSON errors
  - No quotes for strings
  - Simple quotes
  - No comma between lines
  - Comma at the end

Your notes

Assignment #2

☐ Find a valid tool id.

☐ Create a valid JSON request body for the
POST /orders request. Inspect the response.

☐ Adapt your JSON request body by specifying a tool that is NOT in stock. Inspect the response.

Your observations

GET vs POST

GET

- no data should be changed with GET
- You can call GET multiple times with no effect
- While technically possible, it does not carry a payload.

POST

- Each time you call POST, new data will be created
- Usually has a payload (request body)

Your notes

Using random data in requests (random variables)

- Postman offers a long list of random variables
- Go to any Postman request and start typing {{$ and select an item from the list
- For example: ${{randomFullName}}
- In JSON, keep the double quotes if the returned values is a string
- Show the Postman console to inspect which values has been sent

Resources

- Dynamic variables in Postman
  https://postman-quick-reference-guide.readthedocs.io/en/latest/dynamic-variables.html

Your notes

How not to use Postman

What is Postman NOT?

- Not for dealing with user-interactions, like filling out forms, clicking buttons.
- Not a good tool for performance testing or any other kinds of tests where you send to send a lot of requests in a short time-frame

It can be used for security testing, but this is not the primary focus of the tool.

Your notes

Assignment #3

☐ Find the endpoint for getting all orders and create a request.

☐ Find the endpoint for getting a single order and create a request.

Your observations

PATCH request method

- The PATCH request method (if supported by the API), allows you to change existing data
- With a PATCH request, you don’t need to provide the entire data object, only the properties that need to be changed.

Your notes

DELETE request method

- Use DELETE to remove data (if the API supports this)
- Typically no request body is required
- Use a GET request to see if the delete was successful.

Your notes

Preparing for automation

Automation basics

- Manually testing an API is a lot of work
- When someone makes a change to the API, we have to manually test all endpoints and parameters to see if the API is working as before.
- We can let Postman test the API by writing API tests.
- Automation means that we let Postman do the testing work, and we only step in if something goes wrong.

Your notes

Your first API test

- Postman has no idea if a request was successful or not
- We typically write tests to assert if the response contains something that we expect
- We already know the request, so it is less interesting to write tests for that
- Postman uses JavaScript for writing scripts
- The most basic test is to check for the status code

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

- Heads-up! Always make your test fail

- To make assertions on the response body, you need to parse the JSON response

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

- This is a typical test structure:

```
pm.test("Basic test structure", () => {
    pm.expect(1).to.eql(1);
});
```

- Use Postman console to show the value of the property
- Use both response.status and response['status']

```
pm.test("Status is UP", () => {
    const response = pm.response.json();
    pm.expect(response.status).to.eql("OK");
});
```

Your notes

Assignment #4 - Status code for all other requests

- Go through all the requests in the collection and create a status code test for each of them.
- Make sure the tests will fail, if needed

Your observations

Postman variables

- copy/pasting data from one request to the other is annoying and time-consuming.
- Postman allows you to create different variable types:
    - Collection variables
        - Available only for a collection
    - Environment variables
        - Available only for an environment
        - Useful when you wish to reuse the same collection against different servers running the API, like localhost, testing, production.
    - Global variables
        - Available for the entire workspace

Your notes

Working with Postman variables from scripts

- You can define or set a variable value manually, through the Postman UI or from scripts.
- Getting a collection variable:

```
pm.collectionVariables.get("apiToken")
```

- If the Postman variable does not exist, the value of the expression above will be `undefined`.
- To set a collection variable, you can use an expression like the following:

```
pm.collectionVariables.set("firstName", "John")
```

- **Heads-up!** Don't confuse set with get!
- To get or set a global variable, just replace `collectionVariables` with `globals` in the expressions above.

Your notes

Extracting data from the response

- Setting variables from scripts is most useful when we use data from the response, instead of hard-coding a value

```
const response = pm.response.json();
const tools = response.filter((tool)=> tool.available === true);
pm.globals.set("toolId", tools[0].id);
```

Resources

- JavaScript filter function https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter

Your notes

Assignment #5

- Go to the Create order request and dynamically set the order id from the response body.

Your observations

Assertions on objects

- If no book is being found, it makes sense to write a test to ensure the request will fail

```
pm.globals.set("toolId", tool.id);

pm.test("Tool found", () => {
   pm.expect(tool).to.be.an('object');
   pm.expect(tool.inStock).to.be.true;
   pm.expect(tool.inStock).to.eql(true);
});
```

Your notes

Assignment #6

- Check that the category of the tool is *electric-generators*
- Store the value *electric-generators* as a collection variable called *category*
- Replace all occurrences of *electric-generators* with the Postman variable.

Your observations

Assignment #7

- Use the toolId global variable in all requests.

Your observations

Assignment #8

For the “Get single tool” endpoint write the following assertions:

- check if the id of the tool in the response matches the request
- check if current-stock is greater than 0

- Hints:

```
pm.expect(1).to.be.above(2);
```

Your observations

Automated collection runs

Running the collection manually

- We have all elements in place that ensure we can do proper test automation:
    - We have tests that ensure the API works as expected
    - We have variables that ensure we don't need to copy/paste data
- Go from request to request and run the collection
- If some tests are still failing, I recommend you pause the video and take a minute to fix them.

Your notes

Collection Runner

- The collection runner is a tool built-in Postman that allows us to execute the entire collection with just one click, instead of going through each request.
- You can drag and drop requests to change the execution order
- You can disable requests from the execution
- Iterations: how man times to run the collection (default 1)

Your notes

Request execution order

- The default execution order is the one given by the collection
- We can skip the request “Register API client” by using:

```
postman.setNextRequest("Create order");
```

- **Heads-up!** Use `postman.setNextRequest` and not `pm.setNextRequest` (the latter won’t work).

- Alternative: move Register API client at the end

```
postman.setNextRequest(null);
```

- DON’t create an endless loop!

```
postman.setNextRequest("Status");
```

Your notes

Postman monitors

- With the collection runner, we still need to open Postman and manually run the collection
- Another easy way to automate your collection run is by setting up a Postman monitor. You can create a monitor from the context menu of the collection
- Postman monitors are decoupled from your Postman installation
- Postman collection are executed in the Postman cloud infrastructure
- The Postman monitor will use the INITIAL VALUE of any variables you have defined.

Your notes

Newman - the Postman CLI tool

- Newman is a CLI tool that can run a Postman collection.
- To use newman on your computer, you need to have Node.js installed.
- How to install Node.js?
- Go to https://nodejs.org/en/download/
- Always use the LTS (Long time support) version
- Next, you need to install newman from the terminal

```
npm install -g newman
```

- Getting *WARN deprecated* messages in your logs is normal

```
newman --version
```

- How to get the collection in Newman?
    - File export
    - Public link
    - Postman API (not covered in the tutorial)

Resources

- Newman CLI documentation
  https://github.com/postmanlabs/newman
- Postman API
  https://learning.postman.com/docs/developer/intro-api/
- Postman API Public workspace
  https://www.postman.com/postman/workspace/postman-public-workspace/documentation/12959542-c8142d51-e97c-46b6-bd77-52bb66712c9a

Your notes

HTML reports

- Quite often we wish to generate reports for the collection run.
- Htmlextra reporter is loved by the Postman community

npm install -g newman-reporter-htmlextra

- Warnings during the installation are normal and you can ignore them
- We will use --reporters to specify additional reporters
- **Heads-Up!**
    - Is not reporters = ,
    - no space before or after reporters in the comma-separated list

Resources

- Htmlextra documentation
  https://www.npmjs.com/package/newman-reporter-htmlextra

Automation overview

- We have used Postman to manually test the API and write API tests.
- With a tool like the collection runner we can check with a single click if your collection can run without any manual intervention.
- With newman you can use a professional server that deals with building and testing software, like Jenkins, GitLab CI, Circle CI, TeamCity or anything else you wish to use.

Resources

- Gitlab CI pipeline tutorial for beginners
  https://www.youtube.com/watch?v=Jav4vbUrqII

Conclusion

Conclusion and next steps

- If you want to get a certificate for completing this course, **ensure that all lectures have a marked checkbox**.
- The certificate will be automatically generated by Udemy

- Take the **free JavaScript programming course** for Postman
  https://www.youtube.com/watch?v=juuhb3W8xT4

- Learn about **data-driven testing** where you use an external CSV or JSON file to feed different data sets in your request
  Part I - https://www.youtube.com/watch?v=fr7UpFNQbLw
  Part II - https://www.youtube.com/watch?v=MOdMKrjTOi4

- **Schema validation** where you essentially test the structure of the response in one go, instead of doing property by property
  Part I - https://www.youtube.com/watch?v=haDQBmQii2g
  Part II - https://www.youtube.com/watch?v=P_So0vpNJCQ

- Authentication with OAuth2
  https://www.youtube.com/watch?v=YpmEkNJubHA

- Feel free to reach out anytime you have questions. I am still there to help you, even after completing the course.
  - Allow me to keep you up-to-date by email:
    https://sendfox.com/lp/1dv56d
  - Connect on LinkedIn (please introduce yourself in the note):
    https://www.linkedin.com/in/vdespa/
  - Subscribe on YouTube:
    http://www.youtube.com/channel/UCUUl_HXJjU--iYjUkIgEcTw?sub_confirmation=1
  - Follow me on Twitter:
    https://twitter.com/vdespa

Take care and bye-bye!