## Exercise 1 - Issuing an HTTP Request

Install Postman and get familiar with the interface.

Perform basic GET requests to resources on the internet (e.g. https://google.com or an API like https://www.googleapis.com/books/v1/volumes?q=query). What responses do you get? Where to find the response body? Where to find the response headers?

## Exercise 2 - Using a Fake API

Using the todos resource of the JSONPlaceholder API, query the API with Postman so that you see all todo items.

Can you get all items by the user with id 3?

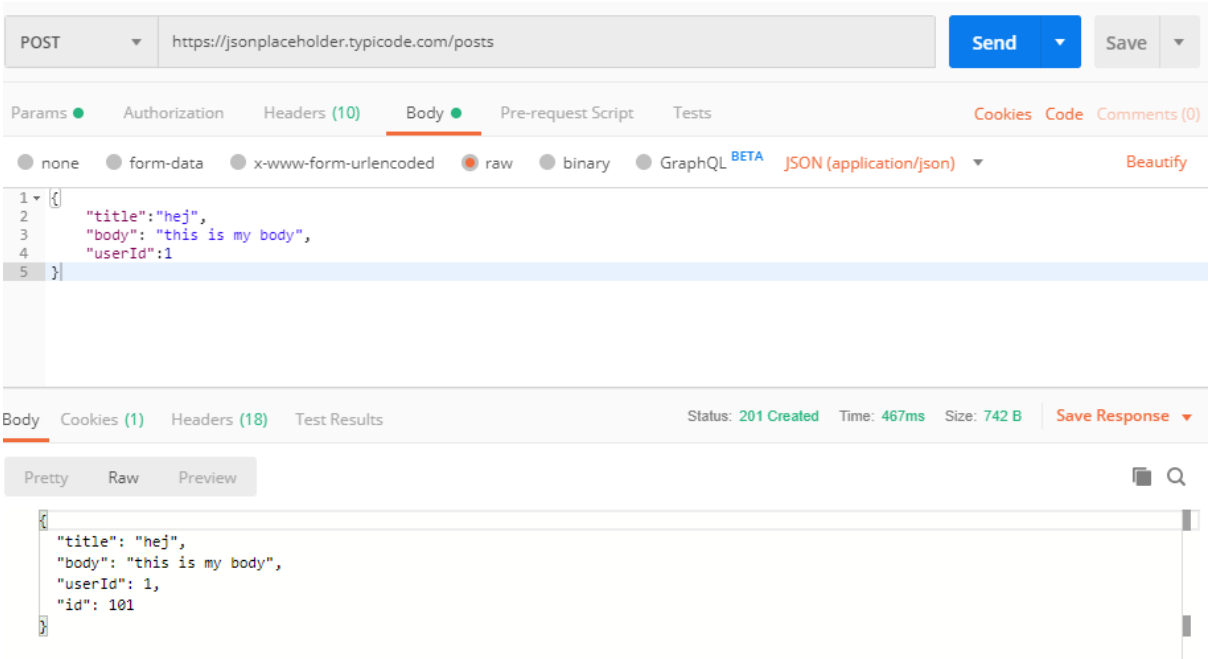
Can you get all items, which are completed?

Query the API so that only todo items that are not completed and have a userId of 5 are shown. What is the name of this user? *(hint: you cannot use the* todos *resource to find this information, but there is a* users *resource available on the website as well)*.

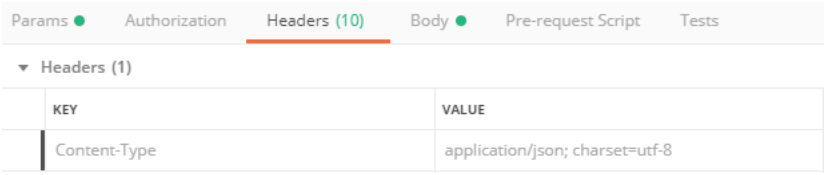
Looking at the documentation of the API, try performing POST, PUT, PATCH and DELETE requests as well (be aware that changes aren't persisted between calls).

**Posting**

You can POST a post to the posts resources like this:




And in the headers have:




Notice the response code: 201, which means my post was created. Yay!

Try to POST something to the TODOs resource in a similar way.

## Exercise 3 - Using a Real API

Scout the internet for public APIs that you can perform queries on using Postman. Read through any available documentation of the APIs and see if you can use it as intended.

Here are some examples if you can't find any APIs yourself:

https://api.github.com

https://earthquake.usgs.gov/fdsnws/event/1/

https://xkcd.com/json.html

You can look up more examples on https://any-api.com (many APIs require an API key before they can be used).

## Exercise 4 - A REST Client

Create a basic console application that consumes the API from exercise 2. Your application should display a list of users (by name and email) in the console using the https://jsonplaceholder.typicode.com/users endpoint.

You may need a refresh from last time.

*Hint: You will need to process the JSON response so that only the name and email of each user is displayed in the console.*

*Hint: this tutorial goes through creating a client for a RESTful webservice.*

## Exercise 5 – Hangman game

You're going to create a Hangman game, i.e. a game about guessing a word.

You should use the following piece of code to get a random word:

```
public async Task<string> GetWord() {
    client.DefaultRequestHeaders.Accept.Clear();
    client.DefaultRequestHeaders.Accept.Add(
        new MediaTypeWithQualityHeaderValue("application/vnd.github.v3+json"));
    client.DefaultRequestHeaders.Add("User-Agent", ".NET Foundation Repository Reporter");
    var streamTask = client.GetStringAsync("https://random-word-api.herokuapp.com/word?key=ULGTRBDR&number=1");
    string result = await streamTask;
    return result;
}
```

It first clears the Headers, then sets the expected type as json.

We get a random word from the random-word-api.

The key was generated from the web-site and is just some kind of security. You have to go get your own here:

**https://random-word-api.herokuapp.com/**
Just replace the key given with the key in the above example.

The “number=1” indicates I want 1 word. I could retrieve more

Test the method, print it out. It’s json format, so you’ll have to do a little work yourself.

Playing the game could look like this in the console:

Guess a word:
_ _ _ _ _
Your letter?
e
Guessed letters so far e,
Result:
_ e_ _ _
Your letter?
r
Guessed letters so far e, r,
Result:
_ e_ r_
Your letter?
t
Guessed letters so far e, r, t,
Result:
_ e_ r_
Your letter?
d
Guessed letters so far e, r, t, d,
Result:
_ e_ r_
Your letter?
w
Guessed letters so far e, r, t, d, w,
Result:
_ e_ r_
Your letter?
y
Guessed letters so far e, r, t, d, w, y,
Result:
_ e_ r_
Your letter?
a
Guessed letters so far e, r, t, d, w, y, a,
Result:
_ ear_
Your letter?
d
Guessed letters so far e, r, t, d, w, y, a, d,
Result:

_ ear_
Your letter?
h
Guessed letters so far e, r, t, d, w, y, a, d, h,
Result:
_ ear_
Your letter?
l
Guessed letters so far e, r, t, d, w, y, a, d, h, l,
Result:
_ ear_
Your letter?
n
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n,
Result:
_ ear_
Your letter?
y
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n, y,
Result:
_ ear_
Your letter?
t
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n, y, t,
Result:
_ ear_
Your letter?
p
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n, y, t, p,
Result:
_ ear_
Your letter?
k
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n, y, t, p, k,
Result:
_ ear_
Your letter?
j
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n, y, t, p, k, j,
Result:
_ ear_
Your letter?
s
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n, y, t, p, k, j, s,
Result:
_ ears
Your letter?
f
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n, y, t, p, k, j, s, f,
Result:
_ ears
Your letter?
g
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n, y, t, p, k, j, s, f, g,
Result:
_ ears
Your letter?
h
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n, y, t, p, k, j, s, f, g, h,
Result:
_ ears
Your letter?
v
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n, y, t, p, k, j, s, f, g, h, v,
Result:
_ ears
Your letter?
b
Guessed letters so far e, r, t, d, w, y, a, d, h, l, n, y, t, p, k, j, s, f, g, h, v, b,
Result:
bears

Congratulations, you won.
Number of guesses 22

## Exercise 6 - Downloading Content

Create a program that asynchronously downloads all contents of https://www.via.dk/ and displays it in the console (that is, all of the HTML in the response). The program should measure the download time and display this in the console after the content.

## Exercise 7 - A Client for a Service

Create a small application using a web service that you decide yourself. For inspiration, a few websites is listed below with access to the documentation of various APIs.

- https://www.programmableweb.com/apis/directory
- https://www.data.gov
- https://developers.google.com/apis-explorer
- http://portal.opendata.dk