Part 1: SWAPI

Instructions

For each of the following use the SWAPI docs, to figure out the complete URL(s) (including params or queries) that you need to go to in order to reach the following data:

1. the height of Darth Vader:

https://swapi.dev/api/people/?search=vader `"height": "202"`

`"https://swapi.dev/api/people/4/"`

2. the population of the planet Alderaan

https://swapi.dev/api/planets/?search=Alderaan `"population": "2000000000"`

`"https://swapi.dev/api/planets/2/"`

3. the name of the manufacturer of the Millennium Falcon

https://swapi.dev/api/starships/?search=Millennium `"manufacturer": "Corellian Engineering Corporation"`

`"https://swapi.dev/api/starships/10/"`

4. the name of the species that C-3PO belongs to (multiple URLs)

https://swapi.dev/api/people/?search=C-3PO

`"https://swapi.dev/api/species/2/"`

`"https://swapi.dev/api/people/2/"`

5. the title of each film that Obi-Wan Kenobi is in (multiple URLs)

https://swapi.dev/api/people/?search=Obi-Wan


```
"films": [
                "https://swapi.dev/api/films/1/",
                "https://swapi.dev/api/films/2/",
                "https://swapi.dev/api/films/3/",
                "https://swapi.dev/api/films/4/",
                "https://swapi.dev/api/films/5/",
                "https://swapi.dev/api/films/6/"
            ]
```


6. use the search query (the how to on the search query is at the bottom of the Getting Started section of the documentation) to get the information about the Millennium Falcon, it's a starship

https://swapi.dev/api/starships/?search=Millennium


```
 "count": 1,
    "next": null,
    "previous": null,
    "results": [
        {
            "name": "Millennium Falcon",
            "model": "YT-1300 light freighter",
            "manufacturer": "Corellian Engineering Corporation",
            "cost_in_credits": "100000",
            "length": "34.37",
```

```
        "max_atmosphering_speed": "1050",
        "crew": "4",
        "passengers": "6",
        "cargo_capacity": "100000",
        "consumables": "2 months",
        "hyperdrive_rating": "0.5",
        "MGLT": "75",
        "starship_class": "Light freighter",
        "pilots": [
            "https://swapi.dev/api/people/13/",
            "https://swapi.dev/api/people/14/",
            "https://swapi.dev/api/people/25/",
            "https://swapi.dev/api/people/31/"
        ],
        "films": [
            "https://swapi.dev/api/films/1/",
            "https://swapi.dev/api/films/2/",
            "https://swapi.dev/api/films/3/"
        ],
        "created": "2014-12-10T16:59:45.094000Z",
        "edited": "2014-12-20T21:23:49.880000Z",
        "url": "https://swapi.dev/api/starships/10/"
    }
]}
```

Part 2: Social Mountain

Summary

In this section, you'll be looking through the documentation for the Social Mountain API and answering questions. You'll also be making requests and recording the URLs and some information about the responses. Run the requests in Postman. **Note: this API is live and viewable by your classmates and staff. Keep things appropriate for class.**

You can view the documentation for the Social Mountain API here

The base URL of your requests is: https://practiceapi.devmountain.com/api (make sure to have the "s" in "https")

1. Check if the POST request accept params, queries, and/or a body. Which one(s) and what information is it expecting to be sent?

2. What data type does the GET request return?

https://practiceapi.devmountain.com/api/posts

It returns Objects.

3. What would the URL look like for deleting the post with the id 555? (This post does not exist anymore, but the syntax is the same for existing posts,)

DELETE: https://practiceapi.devmountain.com/api/posts/555

[{"Id": 555,

"text": " Hello world!",

"date": "11 Jan 2018"

}]

4. List the possible response codes from the GET request at '/posts/filter'

200: Returns the array of filtered posts. 400: Request query is missing required text property.

5. Create a post whose text is your name, record the URL and body here:

POST: https://practiceapi.devmountain.com/api/posts

```
{

  "text": "Antonio"

}
```

6. What would the URL and body object be to update the post you just made to contain your favorite color instead of your name?

Post: https://practiceapi.devmountain.com/api/posts

```
{

  "text": "Red and Black"

}
```

7. What is the URL to get posts that contain the text “blue”?

https://practiceapi.devmountain.com/api/posts/filter?text=blue

8. Make a request to GET all the posts. What are the content type and charset of the response? (Hint: look on the Headers)

content-type: application/json; charset=utf-8

9. What would cause a PUT request to return a 409 status?

If you were trying to update something that isn’t within the req.query.id or req.body.text. So either a typo or something that does not currently exist.

10. What happens if you try to send a query in the GET request URL? Why do you get that response?

When i try to get the query in the get request URL i get this message:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Error</title>
</head>

<body>
    <pre>Cannot GET /api/</pre>
</body>

</html>
```

I think its because it isn't set up to receive a query yet. It needs to be setup in order to receive a query