Web API client

INF-2300 - Assignment 2

Deadline: Wednesday 02. October 2024

Goals

The goal of this assignment is to familiarise you with web clients and servers and the relation between these. We aim to expose you to some of the important aspects of working against a given server backend API. Any web client must adhere to restrictions and limitations imposed by the server which provides the data we want. These limitations might include authorisation, error-handling, rate limiting etc. This assignment builds on the knowledge of the HTTP protocol acquired from assignment 1.

TODO List - Web Client

You will write a web client which will communicate with the provided server and its API. The web client should feature a user-friendly interface presenting the following functionality to the user:

- A main display of all the items currently on the TODO list.
- A way to delete items from the TODO list.
- A way to mark items as completed without removing it from the TODO list.
- A way to easily clear the whole TODO list.

Server API

The given server exposes a simple Application Programming Interface (API) to handle a TODO list. The API supports JSON only. It runs on *http://localhost:5000/api/* and exposes the following services.

HTTP methods	**URI**	**Action**
GET	/items/	Get a list of items
GET	/items/[item_id]	Get a specific item
POST	/items/	Create an item
PUT	/items/[item_id]	Update an item
DELETE	/items/[item_id]	Delete an item

You should familiarise yourself with how the requests to the server should be formatted. Test the server with unsupported requests, badly formatted requests and so on. Here is an example request with the response from the default server:

```
POST http://localhost:5000/api/items/ HTTP/1.1

{
"name": "Kake"
}
```

Response:

```
HTTP/1.0 200 OK
Content-Type: application/json
<More headers>

{
  "item": {
    "done": false,
    "id": 3,
    "name": "Kake"
  }
}
```

The server maintains a list of JSON-objects to represent the items in the TODO list. It is not persistent, which means any changes to the list is lost if the server is restarted.

Requirements

Your client should expose a user friendly interface as mentioned in the description. In addition to these points, your client should make some attempt at handling errors. The provided server gives verbose feedback on bad requests, this should be integrated into the client. Keep in mind that errors can come from two 'directions', both the user and the server. You should attempt to handle both. In other words, erroneous user input shouldn't crash the client, nor should an error from the server crash your client.
The most straight-forward solution to this task is a static HTML page and JavaScript to handle communication with the API, or a console application. Feel free to use frameworks such as jQuery, React, Angular, Vue, etc.

Resources

- Restlet - Client. Testing an API. Available as a Chrome Extension
  https://restlet.com/
- For manually sending requests from the command line, use *httpie*. Multiplatform support. Easy to use.
  https://httpie.org
- More GUI based API testing tools
  https://insomnia.rest

Hand-In

Hand in your solution and the report by uploading it in a compressed file to Canvas. Your report should be in pdf format in the 'doc' folder. Don't forget to update your README!

Good luck!