Fetching Data from API with FastAPI and JavaScript and Inserting into a Table

Tomasz Zadworny

January 24, 2024

1 Introduction

In this exercise, we will explore the process of fetching data from an API using FastAPI on the server side and displaying it on the client side using JavaScript with the `fetch` API.

2 Server-side Implementation with FastAPI

Create a FastAPI application that serves a simple API endpoint returning JSON data. The following Python code snippet demonstrates a basic FastAPI application:

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

origins = [
    "http://localhost:8888",
]

app.add_middleware(
    CORSMiddleware,
    allow_origins=origins,  # Dla wszystkich domen (uwaga: lepiej określić konkretne adresy)
    allow_credentials=True,
    # allow_origins=["*"],  # Dla wszystkich domen (uwaga: lepiej określić konkretne adresy)
    # allow_credentials=False,
    allow_methods=["*"],  # Dla wszystkich metod HTTP
    allow_headers=["*"],  # Dla wszystkich nag␣lówków HTTP
)

data = [
```

```
    {"id": 1, "name": "Example 1"},
    {"id": 2, "name": "Example 2"},
    {"id": 3, "name": "Example 3"},
]

@app.get("/api/data")
def get_data():
    return data
```

2.1 Poetry Configuration

To configure your Python project using Poetry, follow these steps:

1. **Initialize Poetry:** Run the following command to initialize your project with Poetry.

   ```
   poetry init
   ```

   This command will guide you through the project setup process, prompting for information such as the project name, version, and dependencies.

2. **Add FastAPI:** Use the following command to add FastAPI as a dependency to your project.

   ```
   poetry add fastapi
   ```

   This will automatically update your `pyproject.toml` file with the FastAPI dependency.

3. **Add Uvicorn:** Similarly, add Uvicorn as a dependency.

   ```
   poetry add uvicorn
   ```

4. **Install Dependencies:** Execute the following command to install the dependencies.

   ```
   poetry install
   ```

   This command installs the project dependencies, including FastAPI and Uvicorn.

5. **Install (No-Root):** For development purposes, it's recommended to use the following command to install dependencies without including the project itself.

   ```
   poetry install --no-root
   ```

This command installs dependencies without including the main project. It's useful when you're working in a virtual environment.

6. **Run Uvicorn:** To run your FastAPI application using Uvicorn, execute the following command:

   ```
   poetry run uvicorn --reload app.main:app --host 0.0.0.0 --port 9999
   ```

   This command starts the FastAPI application with Uvicorn, enabling hot-reloading and specifying the host and port.

These steps help you configure your Python project using Poetry, adding FastAPI and Uvicorn as dependencies and providing commands to install and run your application.

3 Client-side Implementation with JavaScript

Design an HTML page that uses JavaScript and the `fetch` API to retrieve data from the FastAPI endpoint and display it in a table. The following HTML and JavaScript code can serve as a starting point:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Fetch Data Example</title>
</head>
<body>

<table id="data-table">
    <thead>
        <tr>
            <th>ID</th>
            <th>Name</th>
        </tr>
    </thead>
    <tbody id="data-body"></tbody>
</table>

<script>
    document.addEventListener("DOMContentLoaded", function () {
        fetchData();
    });

    async function fetchData() {
```

```
        try {
            const response = await fetch('http://localhost:9999/api/data');
            const data = await response.json();
            populateTable(data);
        } catch (error) {
            console.error('Error fetching data:', error);
        }
    }

    function populateTable(data) {
        const tableBody = document.getElementById('data-body');

        data.forEach(item => {
            const row = document.createElement('tr');
            row.innerHTML = `<td>${item.id}</td><td>${item.name}</td>`;
            tableBody.appendChild(row);
        });
    }
</script>

</body>
</html>
```

4 Conclusion

This exercise introduces the basics of using FastAPI to create a simple API and using JavaScript with the `fetch` API to retrieve and display data on the client side.