FastAPI Application Documentation

Overview

This document outlines the structure and functionality of a FastAPI application designed for handling file operations, data processing, and API interactions which returns JSON data. It details the Python modules used, the setup of logging, the initialization of the FastAPI application, data model definitions, and endpoint configurations. Below is a detailed explanation of the code, including descriptions of each component and endpoint.

The endpoints are:

1. "validationbody"
2. "databasevaluesbody"
3. "tablebody"

Prerequisites:

1. Python: Ensure you have Python 3.6+ installed. You can download it from python.org.
2. Packages: Install required packages using pip:

   ```
   pip install fastapi pandas uvicorn
   ```

Steps to Run:

1. Save the Code: Save the provided code in a file named FastAPI_EDX.py.
2. Run the Service: Open a terminal and navigate to the directory where FastAPI_EDX.py is saved.
3. Start the FastAPI Server: Run the following command to start the FastAPI server:

   ```
   uvicorn FastAPI_EDX:app --reload
   ```

   The --reload flag automatically reloads the server when you make changes to the code.

4. Access the API: Open a web browser and go to http://127.0.0.1:8000.

Code Explanation:

Modules and Imports:

- **FastAPI:** A modern, fast web framework for building APIs.
- **Request:** Represents an HTTP request in FastAPI, providing access to various parts of the HTTP request.
- **HTTPException**: Allows for raising exceptions that the FastAPI app can catch to return standardized error responses.
- **typing (List, Optional)**: Provides runtime support for type hints.
- **os**: Provides a portable way of using operating system-dependent functionality like reading or writing to the filesystem.
- **tempfile**: Generates temporary files and directories.
- **datetime as dt**: Manipulates dates and times.

- **logging**: Logs events for a library or application.
- **uvicorn**: A lightning-fast ASGI server for Python.
- **pandas as pd**: Offers data structures and operations for manipulating numerical tables and time series.

Setup and Configuration:

- Log files are stored in ./Loggs. The directory is created if it does not exist using os.makedirs.
- Log filename is based on the current date, structured as YYYY-MM-DD.log.
- Logging is set to capture info-level messages with a specific format that includes timestamps.

FastAPI App Initialization:

- A FastAPI instance is created with app = FastAPI().
- API documentation is accessible at the root URL (/).

API Endpoints:

- **Validation body:**

Description: This endpoint is designed to handle the validation of incoming TXT files by reading the file content, storing it temporarily, validating the file, and returning a response indicating the validation status. This method accepts a binary TXT file and if the file is valid, it returns a JSON object with Code: 0. Otherwise, when it is invalid or an error occurs, it logs the error and returns a JSON object with Code: 500 and the error message.

Request:

- Method: POST
- Body: Binary data in string format

Validation Response:

```
{
  "Code": 0,
  "Message": null,
  "Warning": null
}
```

In case of occurring an error:

```
{
  "Code": 500,
  "Message": "System.Exception column 0: MA (unknown column name)",
  "Warning": null
}
```

- **Databasevaluesbody**

Description: This endpoint processes incoming requests and returns a predefined EDX template data. The response is automatically validated and serialized according to the EDXTemplate model. FastAPI ensures that the response data matches the structure and types defined in the model.

Request:

- Method: POST
- Body: Expects binary data

Database Values Response:

```
{
  "Compositions": [
    {
      "CompositionElements": [
        {"CompoundIndex": 0, "ElementName": "V", "ValueAbsolute": null, "ValuePercent": 31.299999237906547},
        {"CompoundIndex": 0, "ElementName": "Mn", "ValueAbsolute": null, "ValuePercent": 2.700000047683716},
        {"CompoundIndex": 0, "ElementName": "Co", "ValueAbsolute": null, "ValuePercent": 22.600000381469727},
        {"CompoundIndex": 0, "ElementName": "Ni", "ValueAbsolute": null, "ValuePercent": 6},
        {"CompoundIndex": 0, "ElementName": "Ho", "ValueAbsolute": null, "ValuePercent": 37.400001525878906}
      ],
      "DeletePreviousProperties": true,
      "Properties": [
        {
          "PropertyID": 0,
          "Type": 2,
          "Name": "Measurement Area",
          "Value": 1,
          "ValueEpsilon": null,
          "SortCode": 10,
          "Row": null,
          "Comment": null
        }
      ]
    }
  ]
}
```

The JSON structure for "Database Values Response" is a nested object used to represent the composition of elements within a database.

Compositions: This key contains an array of composition objects, each representing a distinct composition entity.

Each composition object can contain several fields:

- **CompositionElements**: This is a list of the elements that make up the composition of the sample.

- **CompoundIndex**: An identifier that could be used to reference a specific compound within a database. The same index across different elements indicates they are part of the same compound or mixture.
- **ElementName**: The name of the element (e.g., V for Vanadium, Mn for Manganese, etc.).
- **ValueAbsolute**: The absolute quantity or concentration of the element.
- **ValuePercent**: The percentage concentration of the element within the sample. This shows the proportion of each element relative to the entire composition.

- **DeletePreviousProperties**: A Boolean (true or false) indicating whether previously stored properties of this composition in the database should be deleted before updating with the new data. true means the old data should be removed.

- **Properties**: Additional properties associated with the composition.
  - **PropertyID:** A unique identifier for the property.
  - **Type:** An integer that represent type of the property, referring to different types of measurements which is here an integer.
  - **Name**: The name of the property, which here is "Measurement Area".
  - **Value:** The value associated with the property. For "Measurement Area", the value is 1 which is a float.
  - **ValueEpsilon:** This is intended to represent a margin of error or uncertainty in the measured value, set to null here indicating no such data is provided.
  - **SortCode**: A numerical code used for sorting or categorization in reports or database queries.
  - **Row**: This is used to reference a specific row in a database or dataset.
  - **Comment**: Any additional comments or notes related to the property, null in this sample suggesting no comments are provided.

➢ **Tablebody**

Description: This endpoint is designed to handle the processing of incoming TXT files by reading the file content, and returning a predefined table data structure as a response.

Request:
- Method: POST
- Body: Binary data in string format

Response:

Table Body Response in JSON format:
```
table_data = {
    "DataTable": [
      {
        "Spectrum": "Spektrum 1 {1}",
        "In stats.": "Yes",
        "X (mm)": -44.6,
        "Y (mm)": -40.3,
                "C": 80.36562,
                "O": 17.7799,
                "Si": 0.331076,
                "V": -0.3328189,
                "Mn": 0.09434319,
                "Co": 1.12593,
                "Ni": 1.070188,
                "Ho": -0.4342265
            }
        ]
    }
```

This block sets up table_data as a dictionary containing a list of dictionaries, each representing a row in a data table.