# FastAPI a Rest API

Say OL

say_ol@yahoo.com

October 01, 2024

# Overview of HTTP Request Types

- GET: Retrieve data
- POST: Create new data
- PUT: Replace/update existing data
- PATCH: Partially update existing data
- DELETE: Remove data

# Imports

- FastAPI and HTTPException for building the API and handling errors.
- Pydantic's BaseModel for data validation.
- Typing for type hints and list handling.

**Example:**

```
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
from typing import List, Optional
```

# Data Models

- **Item Model**:
  - Represents an item with an ID, name, and optional description.
- **ItemUpdate Model**:
  - Represents fields that can be updated, both name and description are optional.

**Example:**

```
class Item(BaseModel):
    id: int
    name: str
    description: Optional[str] = None

class ItemUpdate(BaseModel):
    name: Optional[str] = None
    description: Optional[str] = None
```

# GET: List of Items

- **Purpose**: Retrieve a list of all items.
- **Characteristics**:
  - Safe and idempotent.
  - Data returned as a list.

**Example:**

```
@app.get("/items/", response_model=List[Item])
def read_items():
    return items_db
```

# GET: Item by ID

- **Purpose**: Retrieve a specific item by its ID.
- **Characteristics**:
  - Safe and idempotent.
  - Returns a single item.

**Example:**

```
@app.get("/items/{item_id}", response_model=Item)
def read_item(item_id: int):
    for item in items_db:
        if item.id == item_id:
            return item
   raise HTTPException(status_code=404, detail="Item not found")
```

# POST

- **Purpose**: Send data to create a new resource.
- **Characteristics**:
  - Data included in the body of the request.
  - Not idempotent.
  - Used for form submissions or file uploads.

**Example:**

```
@app.post("/items/", response_model=Item)
def create_item(item: Item):
    items_db.append(item)
    return item
```

# PUT

- **Purpose**: Update or create a resource.
- **Characteristics**:
  - Sends data in the body, replacing the entire resource.
  - Idempotent.

**Example:**

```
@app.put("/items/{item_id}", response_model=Item)
def update_item(item_id: int, updated_item: Item):
    for index, item in enumerate(items_db):
        if item.id == item_id:
            items_db[index] = updated_item
            return items_db[index]
   raise HTTPException(status_code=404, detail="Item not found")
```

# PATCH

- **Purpose**: Partially update an existing resource.
- **Characteristics**:
  - Sends only the changes in the body.
  - Idempotent.

**Example:**

```
@app.patch("/items/{item_id}", response_model=Item)
def patch_item(item_id: int, item_update: ItemUpdate):
    for index, item in enumerate(items_db):
        if item.id == item_id:
            if item_update.name is not None:
                item.name = item_update.name
            if item_update.description is not None:
                item.description = item_update.description
            items_db[index] = item
            return items_db[index]
   raise HTTPException(status_code=404, detail="Item not found")
```

# DELETE

- **Purpose**: Remove a specified resource.
- **Characteristics**:
  - Idempotent.

**Example:**

```
@app.delete("/items/{item_id}", response_model=dict)
def delete_item(item_id: int):
    for index, item in enumerate(items_db):
        if item.id == item_id:
            del items_db[index]
            return {"message": "Item deleted"}
   raise HTTPException(status_code=404, detail="Item not found")
```