Understanding the Core Concepts of FastAPI

This document explains the fundamental concepts used in the fastapi_basics.py file.

1. What is FastAPI?

FastAPI is a modern, high-performance web framework for building APIs with Python. Its key features are:

- **Fast:** It's one of the fastest Python frameworks available, on par with NodeJS and Go.
- **Easy to code:** It's designed to be simple and intuitive, which speeds up development.
- **Automatic Docs:** It automatically generates interactive API documentation (using Swagger UI and ReDoc), which is incredibly useful for testing and sharing your API.
- **Type Hints & Validation:** It heavily uses Python type hints, which provides great editor support (autocompletion) and powerful data validation at runtime.

2. The FastAPI Instance

```
from fastapi import FastAPI
app = FastAPI()
```

The app object is the heart of your application. You use it to declare all of your API's routes (or "path operations").

3. Path Operations and Decorators

A "path operation" refers to handling an HTTP request at a specific URL path. You declare them using decorators.

- **Decorator:** A decorator in Python is a special function that adds functionality to another function. In FastAPI, decorators like @app.get() or @app.post() link a URL path and an HTTP method to your Python function.
- **HTTP Methods:**
  - @app.get("/"): Handles GET requests. Used for retrieving data.
  - @app.post("/items/"): Handles POST requests. Used for creating new data.
  - @app.put("/items/{item_id}"): Handles PUT requests. Used for updating existing data.
  - @app.delete("/items/{item_id}"): Handles DELETE requests. Used for deleting data.

4. Path and Query Parameters

You can pass variables to your API through the URL.

- **Path Parameters:** These are parts of the URL path itself. You define them using curly

braces {}.
    - **Example:** In the path /items/{item_id}, item_id is a path parameter. FastAPI uses Python type hints (item_id: int) to convert the value from the URL into the correct type.
- **Query Parameters:** These are key-value pairs that come after a ? in the URL.
    - **Example:** In the URL /items/?skip=0&limit=10, skip and limit are query parameters.
    - Any function parameter in your path operation function that is *not* a path parameter is automatically treated as a query parameter. You can provide default values to make them optional (skip: int = 0).

5. Request Body and Pydantic Models

When a client needs to send data to your API (e.g., when creating a new item), it sends it in the **request body**.

- **Pydantic BaseModel:** To define the structure of the data you expect, you create a class that inherits from pydantic.BaseModel.
```
from pydantic import BaseModel

class Item(BaseModel):
    name: str
    price: float
```
- **How it works:** When you declare a parameter with this Pydantic model type (def create_item(item: Item)), FastAPI will automatically:
    1. Read the request body as JSON.
    2. Validate that the JSON has the required fields (name and price).
    3. Convert the data to the specified types (e.g., str, float).
    4. If the data is invalid, it returns a clear JSON error message.
    5. Make the data available in your function as a Python object (item.name, item.price).

6. Automatic Interactive Documentation

This is one of FastAPI's best features. You don't have to do anything extra. Once your app is running, just go to these URLs in your browser:

- **/docs**: Provides the interactive Swagger UI. You can see all your endpoints and even test them directly from the browser.
- **/redoc**: Provides an alternative documentation style with ReDoc.

7. Running the Application with Uvicorn

FastAPI is a framework, but it needs a server to run it. **Uvicorn** is an "ASGI" (Asynchronous Server Gateway Interface) server that is recommended for FastAPI.

To run the app, you use the command:

```
uvicorn fastapi_basics:app --reload
```

- fastapi_basics: The name of your Python file (the module).
- app: The FastAPI() instance you created inside the file.
- --reload: This tells Uvicorn to automatically restart the server whenever you save changes to your code, which is very helpful during development.