

FASTAPI

What is FastAPI?

- FastAPI is a **modern, fast (high-performance) web framework** for building APIs with Python.
- Built on **Starlette** (async web framework) and **Pydantic** (data validation).
- Claims to be **one of the fastest Python frameworks**, near Go & Node.js.
- Documentation-first & developer-friendly.

Why FastAPI?

- **Super fast** performance
- **Asynchronous support** using `async / await`
- **Automatic validation** using Pydantic
- **Auto Swagger UI & Redoc** documentation
- Easily integrates with:
 - Databases (SQLAlchemy, MongoDB)
 - OAuth/JWT Authentication
 - Background tasks, event loops
- Ideal for **microservices, data science models, ML deployments**

FastAPI Features

- Automatic API documentation
- Path & Query parameters
- Data models using Pydantic
- Dependency injection
- Background tasks
- Middleware
- Async support
- WebSockets
- OAuth2 authentication

Installation

```
pip install fastapi
```

```
pip install uvicorn
```

Your First FastAPI Application

main.py

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def home():
    return {"message": "Welcome to FastAPI!"}
```

Run the Server

```
uvicorn main:app --reload
```

- `--reload`: Auto restarts when code changes.

Built-in API Documentation

Open in browser:

1. Swagger UI:

```
http://127.0.0.1:8000/docs
```

2. Redoc:

```
http://127.0.0.1:8000/redoc
```

PATH PARAMETERS & QUERY PARAMETERS

Path Parameters

```
@app.get("/product/{id}")
def get_product(id: int):
    return {"product_id": id}
```

Query Parameters

```
/search?name=laptop&price=50000
```

```
@app.get("/search")
def search(name: str, price: float):
    return {"name": name, "price": price}
```

Pydantic Models (Data Validation)

Why Pydantic?

- Ensures **automatic validation**
- Converts data types
- Auto-documents API
- Prevents invalid requests

Create a Request Body Model

```
from pydantic import BaseModel

class Product(BaseModel):
    name: str
    price: float
    qty: int
```

POST Example

```
@app.post("/product")
def create_product(p: Product):
    return {"message": "Product added", "data": p}
```

Building CRUD API (FastAPI Example)

In-Memory CRUD Example

```
products = {}

@app.post("/product/{id}")
def add_product(id: int, p: Product):
    products[id] = p
    return {"msg": "Created"}

@app.get("/product/{id}")
def get_product(id: int):
    return products.get(id, {"error": "Not found"})

@app.put("/product/{id}")
def update_product(id: int, p: Product):
    products[id] = p
    return {"msg": "Updated"}

@app.delete("/product/{id}")
def delete_product(id: int):
    products.pop(id, None)
    return {"msg": "Deleted"}
```

Exception Handling

Custom Exception

```
from fastapi import HTTPException

@app.get("/item/{id}")
def get_item(id: int):
    if id not in products:
        raise HTTPException(status_code=404, detail="Item not found")
    return products[id]
```