

# Module 2: Getting Started with FastAPI

## Overview

- FastAPI project structure
- Creating GET API endpoints
- Path and query parameters
- Pydantic models
- Creating POST API endpoints

# Why FastAPI?

- High performance, built on **Starlette** and **Pydantic**
- Automatic **data validation** and **type checking**
- Automatic **API docs** (Swagger & Redoc)
- First-class support for `async` and `await`
- FastAPI = Starlette (web) + Pydantic (data)

# FastAPI Project Structure

Common structure:

- `main.py` – entry point
- `routers/` – modular routes
- `models/` – Pydantic or DB models
- `services/` – business logic
- `config.py` – env variables, settings

# Creating Your First Endpoint

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"message": "Hello, FastAPI"}
```

# Running FastAPI with Uvicorn

## Install:

```
pip install fastapi uvicorn
```

## Run the app:

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

- `--reload` enables auto-reload on file changes

# Pydantic model

```
from pydantic import BaseModel
```

```
class Item(BaseModel):  
    name: str  
    price: float  
    in_stock: bool = True
```

```
items: list = list(  
    Item(  
        name="Sample Item " + str(i+1),  
        price=(i+1) * 10.0,  
        in_stock=True)  
    for i in range(10))
```

# Path Parameters

## Define variables in the URL:

```
@app.get("/items/{item_id}")  
def read_item(item_id: int):  
    return {"item_id": item_id}
```

## Examples:

```
curl http://localhost:8000/items/2  
# → {"item_id": 2}
```

```
curl http://localhost:8000/items  
# → 404 Not Found
```

# Query Parameters

## Optional inputs:

```
@app.get("/items/")  
def read_item(skip: int = 0, limit: int = 10):  
    return {"skip": skip, "limit": limit}
```

## Examples:

```
curl http://localhost:8000/items  
# → {"skip": 0, "limit": 10}
```



## Examples (cont):

```
curl http://localhost:8000/items?skip=5&limit=3  
# → {"skip": 5, "limit": 3}
```

```
curl http://localhost:8000/items?skip=5  
# → {"skip": 5, "limit": 10}
```

```
curl http://localhost:8000/items?limit=3  
# → {"skip": 0, "limit": 3}
```

# Using Pydantic Models

```
from pydantic import BaseModel

class Item(BaseModel):
    name: str
    price: float
    in_stock: bool = True
```

# Creating POST Endpoints with Pydantic

```
items = list()

@app.post("/items/")
def create_item(item: Item):
    items.append(item)
    return {"item_name": item.name, "price": item.price}
```

- Example:

```
curl -X POST "http://localhost:8000/items/" \
-H "Content-Type: application/json" \
-d '{"name": "Book", "price": 12.99}'
```

# FastAPI Docs Interface

FastAPI auto-generates API docs:

- Swagger UI: <http://127.0.0.1:8000/docs>
- Redoc: <http://127.0.0.1:8000/redoc>

# Live Coding: Build a Small API

Live demo of:

- Creating routes
- Using path/query parameters
- Validating request body with Pydantic
- Viewing Swagger docs

# Homework

- Build a FastAPI app with at least 3 endpoints
- Use path and query parameters
- Create one POST endpoint with a Pydantic model
- Push your code to GitHub

# Remember

- FastAPI project structure
- GET endpoints
- Path parameters
- Query parameters
- Pydantic request/response models
- POST endpoints
- OpenAPI/Swagger documentation