Day 3 - FastAPI Path Parameters (Complete Guide)

Introduction to Path Parameters

- Path Parameters are dynamic parts of the URL used to pass data to endpoints.
- Example: /users/{user_id} where user_id changes depending on the user.
- Path parameters are captured using {} in the route.
- FastAPI automatically parses and validates them using type hints.

Uses of Path Parameters

1. Fetch specific resources: /users/1
2. Fetch nested resources: /users/1/orders/101
3. Dynamic API endpoints for mobile apps, e-commerce, restaurant apps, etc.

Running FastAPI Server

1. Install FastAPI & Uvicorn:

pip install fastapi uvicorn

2. Run the server:

uvicorn main:app --reload

- main = python file name
- app = FastAPI instance
- --reload = auto-reload on code change

Topics Covered Today

1. Single Path Parameter

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/user/{user_id}")
def get_data(user_id: int):
    return {"details": f"Your user id is {user_id}"}
```

2. Multiple Path Parameters

```
@app.get("/user/{user_id}/order/{order_id}")
def get_data(user_id: int, order_id: int):
    return {
        "user": user_id,
        "order": order_id,
        "details": f"User id is {user_id} and order id is {order_id}"
    }
```

3. Case-Insensitive Strings

```
@app.get("/user/{user_name}")
def get_data(user_name: str):
    user = user_name.lower()
    if user == "hello":
        return {"name": "hello"}
    elif user == "guru":
        return {"name": "guru"}
    else:
        return {"name": "not found"}
```

4. Dictionary Lookup for Real-World Data

```
user = {
    1: {"name": "hello", "orders": {101: {"name": "laptop", "amount": 3700000}}},
    2: {"name": "welcome", "orders": {201: {"name": "mouse", "amount": 250}}}
}

@app.get("/users/{user_id}/order/{order_id}")
def get_data(user_id: int, order_id: int):
    if user_id not in user:
        raise HTTPException(status_code=404, detail="User not found")
    if order_id not in user[user_id]["orders"]:
        raise HTTPException(status_code=404, detail="Order not found")
    order_data = user[user_id]["orders"][order_id]
    user_name = user[user_id]["name"]
    return {
        "user_id": user_id,
        "user_name": user_name,
        "order_id": order_id,
        "order": order_data
    }
```

5. Error Handling

- HTTPException is used to handle missing data.
- Returns proper HTTP status code and JSON message.

6. Validation Basics

- Using type hints (int, str) ensures FastAPI validates input automatically.
- Optional advanced validation using Path() can set min/max values (covered later).

Test Cases / Examples

1. Single parameter: /user/10 -> {"details": "Your user id is 10"}
2. Multiple parameters: /user/1/order/101 -> returns user and order details.
3. Case-insensitive: /user/Hello -> {"name": "hello"}
4. Invalid user: /users/3/order/101 -> 404 "User not found"
5. Invalid order: /users/1/order/999 -> 404 "Order not found"

Topics Not Covered Today (Advanced/Optional)

1. Path Parameter Validation with Path() (min/max, regex)
2. Path Parameter with Enum (restricted string values)
3. Combining Path and Query Parameters
4. Pydantic Models for response schema

End of Day 3 - Path Parameters

All examples tested, explained, and ready for real-world application.