Day 4: FastAPI Query Parameters

---

1. What are Query Parameters

- Dynamic parameters sent after `?` in URL
- Example: `/items/?id=10&name=apple`

---

2. Path vs Query Parameters

- **Path Parameter:** part of URL, e.g., `/users/{id}`
- **Query Parameter:** after `?` in URL, e.g., `/users/?id=10`

---

3. Optional vs Required Query Parameters

- **Required:** no default value; must be provided
- **Optional:** default value provided; can be skipped

Example:

```
from fastapi import FastAPI
from typing import Optional

app = FastAPI()

@app.get("/products/")
def get_products(limit: Optional[int] = 10):
    return {"limit": limit}
```

---

4. Type Hints & Automatic Validation

- FastAPI validates parameter types automatically
- Example:

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

- Invalid input returns `422 Unprocessable Entity`

---

5. Multiple Query Parameters

- Accept more than one parameter in a function

- Example:

```
@app.get("/search/")
def search_items(name: str = None, price: int = None):
    return {"name": name, "price": price}
```

---

6. Real-World Examples

Search Filter:

```
products = [
    {"id": 1, "name": "Laptop"},
    {"id": 2, "name": "Phone"},
    {"id": 3, "name": "Tablet"},
]

@app.get("/search/")
def search(name: str = None):
    if name:
        result = [p for p in products if name.lower() in p["name"].lower()]
        return {"results": result}
    return {"results": products}
```

Filtering:

```
items = [
    {"name": "Laptop", "category": "electronics"},
    {"name": "Shirt", "category": "clothing"},
    {"name": "Phone", "category": "electronics"},
]

@app.get("/filter/")
def filter_items(category: str = None):
    if category:
        return {"results": [item for item in items if
item["category"].lower() == category.lower()]}
    return {"results": items}
```

Sorting:

```
items = [
    {"name": "Laptop", "price": 50000},
    {"name": "Phone", "price": 20000},
    {"name": "Tablet", "price": 30000},
]
```

```python
@app.get("/sort/")
def sort_items(order: str = "asc"):
    if order == "desc":
        items.sort(key=lambda x: x["price"], reverse=True)
    else:
        items.sort(key=lambda x: x["price"])
    return {"results": items}
```

Pagination:

```python
items = [
    {"id": 1, "name": "Laptop"},
    {"id": 2, "name": "Phone"},
    {"id": 3, "name": "Tablet"},
    {"id": 4, "name": "Shirt"},
    {"id": 5, "name": "Shoes"},
]

@app.get("/items/")
def get_items(skip: int = 0, limit: int = 2):
    return {"items": items[skip: skip + limit]}
```

---

7. Error Handling for Query Parameters

- Invalid type → FastAPI automatically returns `422`
- Example:

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

- `/items/?id=abc` → returns 422 error with details

---

8. Combination of Path + Query Parameters

- Path parameter identifies resource, query parameter adds extra info
- Example:

```python
@app.get("/users/{user_id}")
def get_user(user_id: int, active: bool = True):
    return {"user_id": user_id, "active": active}
```

- URL: `/users/1?active=false`

✅Summary

- Query parameters are widely used for **searching, filtering, sorting, and pagination**
- Type hints provide **automatic validation**
- FastAPI handles **errors gracefully**
- Combining path + query parameters makes APIs more flexible