




Week 5

Dependency Injection and Modularization in FastAPI

Overview

- FastAPI's Dependency Injection System
- Reusable Dependencies (DB session, auth)
- Modular App Structure with Routers

Routers vs routes

Term	Context	Correct?
router	When defining modular endpoints using <code>APIRouter</code>	 Yes
routes	When referring to individual endpoints or the list of them	 Yes
<code>routes.py</code>	Common filename for defining routes (not wrong, but <code>routers.py</code> is clearer in modular design)	 Acceptable

FastAPI's Dependency Injection System

What is Dependency Injection (DI)?

- A design pattern for managing dependencies
- Promotes separation of concerns and testability

FastAPI's Approach

- Based on Python's `Depends` function
- Automatically resolves dependencies at runtime

Example of Dependency Injection

```
from fastapi import Depends

def get_query(q: str = None):
    return q

@app.get("/items/")
def read_items(query: str = Depends(get_query)):
    return {"q": query}
```

Why Use Dependency Injection?

- **Decouples components:** Makes it easier to change or replace components without affecting others.
- **Improves testability:** Allows for easier mocking of dependencies in tests.
- **Promotes reusability:** Common dependencies can be reused across different endpoints.

Reusable Dependencies: Database Session

```
from fastapi import Depends
from sqlalchemy.orm import Session
from .database import SessionLocal

def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()

@app.get("/users/")
def read_users(db: Session = Depends(get_db)):
    return db.query(User).all()
```

Database Session - Explanation

- `get_db` function creates a new database session
- Uses `SessionLocal` (Session Factory) to create a session
- Uses `yield` to provide the session to the endpoint
- A function with `yield` pauses at the `yield` statement and can resume later.
- Ensures the session is closed after use
- `Depends(get_db)` injects the session into the endpoint function

Reusable Dependencies: Authentication

```
from fastapi import Depends, HTTPException

def get_current_user(token: str = Depends(oauth2_scheme)):
    user = verify_token(token)
    if not user:
        raise HTTPException(status_code=401)
    return user
```


Modular App Structure with Routers

Why Modularize?

- Scalable codebase
- Easier to maintain
- Logical grouping of features

Creating a Router

```
# routers/users.py
from fastapi import APIRouter

router = APIRouter()

@router.get("/")
def get_users():
    return [{"name": "Alice"}, {"name": "Bob"}]
```

```
# main.py
from fastapi import FastAPI
from routers.users import router as users_router

app = FastAPI()
app.include_router(users_router, prefix="/users")
```

Recommended Project Structure

```
app/  
├── main.py  
├── routers/  
│   ├── users.py  
│   └── items.py  
├── dependencies/  
│   └── auth.py  
├── models/  
│   └── user.py  
└── database.py
```

Live Coding Example (1)

Basic Dependency

- Create a FastAPI application with:
 - A dependency function `get_query_param` that extracts a query parameter `q` (string) from the request.
 - A search endpoint `/search` that returns `{"query": q}` using the dependency.

Live Coding Example (2)

Reusable Database Session

- Simulate a fake database by:
 - Creating a dependency `get_fake_db()` that yields a dictionary `{"users": ["alice", "bob"]}`.
 - Creating an endpoint `/users` that returns the list of users using that dependency.

Live Coding Example (3)

Simulated Auth Dependency

- Write a dependency `get_current_user` that checks if a query parameter `token` equals "secret".
- If the `token` is valid, return "authenticated_user", otherwise, raise an `HTTPException` (401).
- Create an endpoint `/profile` that requires this dependency and returns the current user.

Homework

Mini Project – Modular App

Using `APIRouter`, create a modular FastAPI project with the following:

Folder structure:

```
app/  
├── main.py  
├── routers/  
│   └── users.py  
├── dependencies/  
│   └── auth.py
```

Homework (continued)

Features to implement:

- A `/users/me` endpoint in `users.py` that depends on an `auth` dependency from `auth.py`.

The `auth` dependency (`get_current_user`) should be imported and injected using `Depends`.

Remember

- FastAPI uses `Depends` for DI
- Dependencies improve reusability and testability
- Modular structure with APIRouter