# ✅ FastAPI: Dependency Injection & Database Connectivity (Part 1)

## 1. What is Dependency Injection in FastAPI?

Dependency Injection (DI) in FastAPI is a way to manage and share common logic like database connections, authentication, or services. It allows you to define functions that can be reused across multiple routes without duplicating code.

### Key Points:

* ✔ Reduces code duplication by reusing logic.
* ✔ Improves code modularity and organization.
* ✔ Simplifies managing common tasks (e.g., DB connections, authentication).
* ✔ Use the 'Depends' function to inject dependencies.

### Example of Dependency Injection in FastAPI:

from fastapi import FastAPI, Depends  
  
app = FastAPI()  
  
# Dependency Function  
def common\_dependency():  
 return {"message": "Hello from dependency"}  
  
@app.get("/items/")  
def get\_items(data: dict = Depends(common\_dependency)):  
 return data

## 2. Database Connectivity in FastAPI

FastAPI supports database connectivity using popular ORMs like SQLAlchemy, Tortoise-ORM, and databases. You can manage database sessions and perform CRUD operations efficiently.

### Key Points:

* ✔ FastAPI supports SQL databases (e.g., PostgreSQL, MySQL) and NoSQL (e.g., MongoDB).
* ✔ Use SQLAlchemy for SQL databases (most common ORM).
* ✔ Dependency Injection is used to manage database sessions.
* ✔ Use async/await for handling I/O-bound database operations.

### Example of Database Connection using SQLAlchemy:

from fastapi import FastAPI, Depends  
from sqlalchemy import create\_engine, Column, Integer, String  
from sqlalchemy.ext.declarative import declarative\_base  
from sqlalchemy.orm import sessionmaker, Session  
  
DATABASE\_URL = "sqlite:///./test.db"  
  
# Set up SQLAlchemy  
engine = create\_engine(DATABASE\_URL)  
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)  
Base = declarative\_base()  
  
# Model Definition  
class User(Base):  
 \_\_tablename\_\_ = "users"  
 id = Column(Integer, primary\_key=True, index=True)  
 name = Column(String, index=True)  
  
Base.metadata.create\_all(bind=engine)  
  
# Dependency for DB Session  
def get\_db():  
 db = SessionLocal()  
 try:  
 yield db  
 finally:  
 db.close()  
  
app = FastAPI()  
  
@app.post("/users/")  
def create\_user(name: str, db: Session = Depends(get\_db)):  
 user = User(name=name)  
 db.add(user)  
 db.commit()  
 return {"user": user.name}