FastAPI

Typed Python APIs with validation and docs built in.

01 Install & run

```
pip install "fastapi[standard]"
fastapi dev main.py   # auto-reload
fastapi run main.py   # production
```

```
from fastapi import FastAPI
app = FastAPI()
@app.get("/")
async def root():
    return {"message": "Hello World"}
```

Interactive docs are built in: Swagger UI at /docs, ReDoc at /redoc. Handlers can be def or async def.

02 Path parameters

```
from fastapi import Path
@app.get("/users/{user_id}")
async def read_user(
    user_id: int = Path(gt=0, le=1000),
):
    return {"user_id": user_id}
```

The type hint converts and validates: /users/abc gets a 422.

03 Query parameters

```
from fastapi import Query
@app.get("/items/")
async def read_items(
    q: str = Query(min_length=3),
    skip: int = 0,
    tags: list[str] = Query(default=[]),
):
    return {"q": q, "tags": tags}
```

No default means required. list[str] reads ?tags=a&tags=b.

04 Request body models

```
from pydantic import BaseModel, Field
class Item(BaseModel):
    name: str
    price: float = Field(gt=0)
    description: str | None = None
@app.post("/items/")
async def create_item(item: Item):
    return item
```

A model parameter is read from the JSON body and validated, including nested models. Bad input gets a 422.

05 Where values come from

`user_id: int`	Path, when in the route
`skip: int = 0`	Query string
`item: Item`	JSON body
`Form() / File()`	Form fields, uploads
`Header() / Cookie()`	Headers, cookies
`Depends(func)`	A dependency result

A JSON body cannot share a request with form fields.

06 Response models & status

```
class ItemOut(BaseModel):
    name: str
    price: float
@app.post("/items/", response_model=ItemOut,
          status_code=201)
async def create_item(item: ItemCreate):
    return item  # extra fields are dropped
```

A return annotation (-> ItemOut) filters the same way.

07 Headers & cookies

```
from fastapi import Cookie, Header
@app.get("/me")
async def me(
    user_agent: str | None = Header(None),
    session_id: str | None = Cookie(None),
):
    return {"agent": user_agent}
```

user_agent reads the User-Agent header. Set cookies on a Response parameter with set_cookie(..., httponly=True).

08 Forms & file uploads

```
from fastapi import File, Form, UploadFile
@app.post("/upload/")
async def upload(
    title: str = Form(),
    file: UploadFile = File(),
):
    data = await file.read()
    return {"size": len(data)}
```

Needs python-multipart, included in fastapi[standard]. UploadFile spools big files to disk.

09 Errors

```
from fastapi import HTTPException
@app.get("/items/{item_id}")
async def read_item(item_id: int):
    if item_id not in items:
        raise HTTPException(status_code=404)
    return items[item_id]
```

Pass detail="..." for a message. @app.exception_handler(MyError) maps custom errors.

FastAPI

Dependencies, auth, data and tests.

10 Dependency injection

```python
def paging(skip: int = 0, limit: int = 100):
    return {"skip": skip, "limit": limit}
@app.get("/items/")
async def read_items(
    page: dict = Depends(paging),
):
    return page
```

Classes work too: page: Pagination = Depends().

11 Token auth (OAuth2)

```python
oauth2_scheme = OAuth2PasswordBearer(
    tokenUrl="token")
async def get_current_user(
    token: str = Depends(oauth2_scheme),
):
    user = decode_token(token)
    if not user:
        raise HTTPException(status_code=401)
    return user
```

POST /token takes OAuth2PasswordRequestForm and returns {"access_token": token, "token_type": "bearer"}. The docs use PyJWT and pwdlib.

12 Protect routes

```python
@app.get("/users/me")
async def read_me(
    user: User = Depends(get_current_user),
):
    return user
```

Chain checks: get_admin can depend on get_current_user. dependencies=[Depends(f)] guards a route, router or app.

13 Database session

```python
engine = create_engine("sqlite:///./app.db")
SessionLocal = sessionmaker(bind=engine)
def get_db():
    db = SessionLocal()
    try:
        yield db
    finally:
        db.close()
```

Cleanup after yield runs once the response is sent.

14 CRUD with SQLAlchemy

```python
@app.post("/items/", status_code=201)
def create_item(
    item: ItemCreate,
    db: Session = Depends(get_db),
):
    db_item = Item(**item.model_dump())
    db.add(db_item)
    db.commit()
    db.refresh(db_item)  # loads the new id
    return db_item
```

Use def, not async def: SQLAlchemy blocks by default.

15 APIRouter & multiple files

```python
router = APIRouter(prefix="/items",
                   tags=["items"])
@router.get("/")
async def read_items(): ...
app.include_router(router)
```

Keep each router in its own module, such as app/routers/items.py. tags group routes in /docs.

16 CORS & middleware

```python
app.add_middleware(
    CORSMiddleware,
    allow_origins=["http://localhost:3000"],
    allow_methods=["*"],
    allow_headers=["*"],
)
```

With allow_credentials=True, list exact origins. @app.middleware("http") adds your own.

17 Background tasks & lifespan

```python
@app.post("/report/")
async def report(tasks: BackgroundTasks):
    tasks.add_task(build_report)
    return {"status": "queued"}
from contextlib import asynccontextmanager
@asynccontextmanager
async def lifespan(app: FastAPI):
    app.state.model = load_model()
    yield  # startup above, shutdown below
    app.state.model.close()
app = FastAPI(lifespan=lifespan)
```

Tasks run after the response. Heavy jobs: Celery or ARQ.

18 Testing

```python
client = TestClient(app)
def test_root():
    response = client.get("/")
    assert response.status_code == 200
app.dependency_overrides[get_db] = test_db
```

TestClient (fastapi.testclient) needs no running server. dependency_overrides swaps in a test database.