FastAPI Python web APIs

CONCEPTS

FastAPI reads your type hints: they decide where a parameter comes from and how it is validated.

Request path, query, body
↓ validate
Pydantic types to values
↓ Depends
Path function your code
↓ response_model
Response JSON + OpenAPI

Validation failures return 422 automatically; no manual checks needed.

RUN SERVER

`fastapi dev`
Dev server with reload (auto-detects app)

`fastapi dev <path/to/file.py>`
Run a specific file or package

`fastapi run`
Production mode, no reload, 0.0.0.0

`fastapi dev --port <port>`
Change the port (default 8000)

`fastapi dev --host 0.0.0.0`
Listen on all interfaces

`fastapi dev --app <name>`
Pick the app variable in the module

`fastapi dev -e <module:app>`
Give the import string directly

`fastapi run --workers <n>`
Several worker processes

APP AND ROUTES

`app = FastAPI(title="<name>")`
Create the app

`@app.get("/")`
Handle GET requests

`@app.post("/items")`
Handle POST requests

`@app.put("/items/{item_id}")`
Handle PUT requests

`@app.delete("/items/{item_id}")`
Handle DELETE requests

APP AND ROUTES (cont.)

`router = APIRouter(prefix="/items")`
Group routes in a module

`app.include_router(router)`
Mount a router

`app.add_middleware(CORSMiddleware, allow_origins=[...])`
Enable CORS

PARAMETERS

`def f(item_id: int)`
Path parameter from {item_id}

`def f(skip: int = 0, limit: int = 10)`
Query parameters with defaults

`def f(q: str | None = None)`
Optional query parameter

`q: Annotated[str, Query(max_length=50)]`
Validate a query parameter

`n: Annotated[int, Path(ge=1)]`
Validate a path parameter

`tags: Annotated[list[str], Query()]`
Repeated query values (?tags=a&tags=b)

`x: Annotated[str | None, Header()] = None`
Read a request header

`name: Annotated[str, Form()]`
Read a form field

`f: UploadFile`
Receive an uploaded file

REQUEST BODY

`class Item(BaseModel):`
Define a body schema

`def f(item: Item)`
Parse the JSON body into a model

`price: float = Field(gt=0)`
Constrain a model field

`item.model_dump()`
Model to a dict

`item.model_dump(exclude_unset=True)`
Only fields the client sent (PATCH)

RESPONSES

`@app.get("/x", response_model=Item)`
Filter and document the output

`@app.post("/x", status_code=201)`
Set the success status code

`from fastapi import status`
Named codes: status.HTTP_201_CREATED

`raise HTTPException(status_code=404, detail="...")`
Send an error response

`return JSONResponse(content=<d>, status_code=<n>)`
Build the response yourself

DEPENDENCIES

`def f(db: Annotated[Session, Depends(get_db)])`
Inject a dependency

`CommonParams = Annotated[dict, Depends(common)]`
Reusable dependency alias

`def get_db(): yield db`
Dependency with cleanup after the response

`app.dependency_overrides[<d>] = <f>`
Swap a dependency in tests

TESTING AND DOCS

`client = TestClient(app)`
Test client (needs httpx)

`client.get("/items/1")`
Call a route in a test

`client.post("/items", json={...})`
POST a JSON body

`assert r.status_code == 200`
Check the status

`r.json()`
Decode the JSON response

`/docs`
Swagger UI

`/openapi.json`
OpenAPI schema