



FastAPI

Version 0.115.8 · verified 2026-10-17

High-performance Python web framework built on Starlette and Pydantic with automatic OpenAPI generation.

Video walkthrough (4 min) — scan the code to watch.

Contents

Mental model	2
Application & routing	2
Path & query parameters	3
Request bodies & Pydantic	5
Dependency injection	6
Middleware & CORS	8
Testing & AsyncClient	9
Further reading	11

Mental model

FastAPI runs on Starlette for web routing and Pydantic for data validation. Python type hints define path parameters, query parameters, headers, and request bodies. FastAPI automatically validates incoming payloads against your types, converts inputs, returns JSON responses, and generates interactive OpenAPI (/docs) schemas at runtime.

Application & routing

Concept	Usage
Instantiation	<code>app = FastAPI()</code>
Operation	<code>@app.get("/items")</code>
Router	<code>router = APIRouter()</code>
Include router	<code>app.include_router(router)</code>

An application instance mounts routes via HTTP method decorators. Use `APIRouter` to split endpoints across modules.

```
from fastapi import FastAPI, APIRouter

app = FastAPI(title="Store API")
router = APIRouter(prefix="/users")

@router.get("/")
async def list_users():
    return [{"id": 1, "name": "Ada"}]

app.include_router(router)
```

Tip: Pass `tags=["users"]` to `APIRouter` to organize route groups in the generated interactive `/docs` UI.

Path & query parameters

Parameter	Syntax
Path parameter	<code>def f(item_id: int):</code>
Query parameter	<code>def f(skip: int = 0):</code>
Path validation	<code>item_id: int = Path(gt=0)</code>

Parameter	Syntax
Query validation	<code>q: str None = Query(None)</code>

Function parameters in route path templates are path parameters. Other parameters are inferred as query parameters.

```
from fastapi import FastAPI, Path, Query

app = FastAPI()

@app.get("/items/{item_id}")
async def read_item(
    item_id: int = Path(..., gt=0),
    q: str | None = Query(None, max_length=50)
):
    return {"item_id": item_id, "q": q}
```

Gotcha: A parameter with no default value is required. To make a query parameter optional, set a default like `None`.

Request bodies & Pydantic

Feature	Code
Request body	<code>def f(item: ItemModel):</code>
Pydantic model	<code>class ItemModel(BaseModel):</code>
Field rules	<code>price: float = Field(gt=0)</code>
Nested models	<code>tags: list[TagModel]</code>

Declare request payloads as Pydantic models. FastAPI parses JSON request bodies, validates fields, and injects model instances.

```
from fastapi import FastAPI
from pydantic import BaseModel, Field

class Item(BaseModel):
```

```
name: str

price: float = Field(..., gt=0)

tax: float | None = None


app = FastAPI()


@app.post("/items/")
async def create_item(item: Item):
    return {"name": item.name, "total": item.price}
```

Warning: Returning raw ORM objects without Pydantic models or `response_model` can accidentally expose internal fields.

Dependency injection

Construct	Purpose
<code>Depends(fn)</code>	Declare a function dependency
Shared logic	DB session, auth, pagination

Construct	Purpose
Sub-dependencies	Dependencies depending on dependencies

Dependencies are callable objects or generators executed before route handlers. FastAPI caches dependency results per request.

```
from typing import Annotated
from fastapi import FastAPI, Depends

def common_params(
    q: str | None = None, page: int = 1
):
    return {"q": q, "page": page}

app = FastAPI()

@app.get("/items/")
async def list_items(
```

```
params: Annotated[dict, Depends(common_params)]
):
    return params
```

Tip: Yielding inside a dependency (`yield db`) executes cleanup logic after the route handler completes.

Middleware & CORS

Middleware	Function
HTTP middleware	<code>@app.middleware("http")</code>
CORS	<code>CORSMiddleware</code>
Custom processing	Inspect/modify requests & responses

Middlewares wrap request handling. `CORSMiddleware` configures cross-origin resource sharing headers for web clients.

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware

app = FastAPI()

app.add_middleware(
    CORSMiddleware,
    allow_origins=["https://example.com"],
    allow_methods=["*"],
    allow_headers=["*"],
)
```

Gotcha: `allow_origins=["*"]` cannot be used alongside `allow_credentials=True` due to browser CORS security restrictions.

Testing & AsyncClient

Tool	Usage
TestClient	Synchronous testing with httpx

Tool	Usage
<code>AsyncClient</code>	Asynchronous test requests
Override dep	<code>app.dependency_overrides</code>

Use `httpx.AsyncClient` or `TestClient` to test route endpoints. Override dependencies to swap mock services or test DBs.

```
from httpx import ASGITransport, AsyncClient
from fastapi import FastAPI

app = FastAPI()

@app.get("/ping")
async def ping():
    return {"msg": "pong"}

async def test_ping():
    tr = ASGITransport(app=app)
    url = "http://testserver"
```

```
async with AsyncClient(tr, base_url=url) as c:  
    res = await c.get("/ping")  
    assert res.status_code == 200
```

Tip: Reset `app.dependency_overrides = {}` after tests to prevent state leakage between test cases.

Further reading

- [FastAPI Documentation](#)
- [Starlette Documentation](#)
- [Pydantic Documentation](#)