P1

大家好今天會由我跟另一位講師Chris跟大家介紹FastAPI網頁架構

P2

這是今天的Agenda

我們會先跟大家簡單介紹FastAPI

再來會帶大家了解非同步以及依賴注入

然後會帶大家感受一下SSE

最後帶大家實作WebSocket應用

P3

那我們就開始講解甚麼是FastAPI

FastAPI是一個Python的API框架，專門用來做快速高效率的API

因為使用非同步設計，適合高併發的需求

舉個例子像是ChatAI這種模型推論服務就很適合

P4

Fast 我們很好理解就是快

那甚麼東西要快?就是我們的API

API全名是Application Programming Interface

目的是在我們的後端系統上建立接口

讓像是第三方開發者、其他系統、或是一些應用程式調用方法

P5

那FastAPI有哪些優勢呢?

因為FastAPI基於輕量型網頁框架Starlette開發的

所以他的效能媲美NodeJS或是Go這種高效能框架

非常適合快速大量請求的API應用

P6

FastAPI結合資料驗證工具Pydantic

自動對型別做驗證

有效減少錯誤並提升穩定性

P7

FastAPI會根據API定義自動生成互動式文件

方便測試、前端溝通

P8

FastAPI強調簡單、可讀性高

適合需要快速迭代的專案

P9

綜合以上特性呢

可以提高開發效率2倍到3倍

P10

為甚麼FastAPI效能可以媲美NodesJS或是Go?

這邊要帶大家了解WSGI、ASGI

WSGI 是一種通信標準

他定義我們的應用程式要怎麼跟Web兼容

像是Gunicorn、uWSGI、mod\_wsgi都是走WSGI的通信標準

P11

我們來看WSGI的整個流程

User在訪問我們的網站的時候 會發送Http請求

伺服器會接收到請求像是Gunicorn

那伺服器就會透過WSGI把請求傳給應用程式像是Flask

應用程式處理完後 會把http響應丟給伺服器

最後伺服器把http響應發送回去給User

P12

從這張圖可以看到

WSGI在處裡請求的時候

同一個時間只能處理一個請求

並且在處理完後

才會把資源釋放出來

在最初設計的時候

也沒有考慮到大量請求或是持續傳輸的問題

P13

在看完WSGI可以發現

WSGI已經沒有辦法應付現在的網路需求

所以就出現了ASGI

全名是Asynchronous Server Gateway Interface

最大的改進就是支援非同步的請求處理

支援WebScocket、SSE、長輪詢

P14

假設User發出HTTP 或 WebSocket 請求

ASGI伺服器會接收請求，像是Uvicorn

伺服器會再交給應用程式執行異步處理像是:FastAPI、Starlette

最後返回結果給使用者

P15

相較WSGI

ASGI可處理傳統 HTTP 請求

也可以處理 WebSocket、SSE、背景任務

適合高並發應用（如聊天應用、即時通知）

P16

這邊幫大家整理成簡單的比較表

可以看到ASGI可以支援異步執行也支援WebScocket

WSGI就只適合一般HTTP服務

ASGI除了一般HTTP服務還可以用在高併發應用

像是Uvicorn、Daphne 使用ASGI

Gunicorn、uWSGI 使用WSGI

框架的話ASGI有FastAPI、Django Channels

WSGI有Flask、Django

P17

在選擇Python的後端開發框架時

FastAPI、Django、Flask 是三個主要的選擇

FastAPI是一個比較新的框架

注重高效能和非同步處理

非常適合架設高效能應用像是推論模型服務

Flask 是一個輕量級框架，設計簡潔且高度靈活

適合簡單且客製化需求高的應用

Django是一個功能完整的框架

內建很多功能，比較適合全功能的應用程式開發

FastAPI上手非常容易

P18

Uvicorn 是一個ASGI Python Web 伺服器

他有兩個特性

一是支援非同步 I/O

二是啟動非常快速

所以常用來部署支援非同步的 Web 框架

像是FastAPI或是Starlette這類支持高併發的 Web 應用。

P19

我們先安裝fastapi uvicorn

我們引入fastapi uvicorn套件

將app指定為一個FastAPI實例

定義get路由 路由網址是resource

裡面會回傳ok

使用uvicorn伺服器

注意FastAPI:app是FastAPI.py中的app

Host 使用127.0.0.1給本機使用的Localhost

0.0.0.0 是本機上的所有IPV4地址

Port設定可以自己決定

P20

Pydantic主要功能是做資料驗證和資料的解析

透過定義型態來做資料驗證和序列化

Python中最快的資料驗證library

可以生成JSON Schema

或從JSON、字典或其他結構化資料格式中

解析並建立資料模型

支援Lax、Strict模式

Lax 寬鬆模式 Pydantic 會「自動幫你轉型別」

**Strict 嚴格模式**必須提供「**正確的資料型別**」

Pydantic **不會幫你轉換**，只要型別錯就報錯。

支援多種型態的驗證

支援客製的驗證

P21

http協定就是客戶端跟伺服器端

大家一起講好要用甚麼方式溝通

P22

http的請求方法有8種

這邊列出比較常見的5種

Get就是我們CRUD的Read

用來讀取資料

Post對應到Create

用來新增資料

Put Patch對應到Update

都是用來更新資料

Patch只更新部分資料

Delete刪除資料

Server回傳的狀態碼有四種

2開頭是OK

3開頭是重新定向

4開頭是客戶端錯誤

5開頭是伺服器錯誤

P23

以往設計API 的時候

每進行一項資料的操作時都要設計一個 URL 網址

如果依照RESTful API

只需要一個URL網址

依照你的http method

即可滿足所有資料的操作。

P24

Swagger是一間名為SmartBear Software

開發出的REST API 的工具

後來貢獻給OpenAPI Initiative

並公開讓所有人都能夠使用

為什麼要用Swagger

以前可能會用像是Word、Excel或是HackMD寫文件

通常會有幾個問題

API更新文件沒更新

寫的時候寫錯

工程師也不是很願意寫

Swagger能夠自動生成API文件

而且還可以在線上進行測試

FastAPI還會自動生成Swagger

P25

這邊我們多引入一些套件

然後定義一個水果的類別

P26

然後我們建立一個假DB

然後我們建立一個get方法

P27

我們再建立Get ByID 方法

如果找不到資料的話回傳404

P28

建立Post

新增資料在資料庫有重複Name的話

回傳400表示用戶端的錯誤

P29

新增Put

這邊比較暴力一點

會整個資料做更新

P30

最後實作Delete

找不到資料回傳404

P31

在這個章節

我會帶大家把剛剛的專案改成非同步方法

然後我們會使用SQL連線資料庫

P32

依賴注入是在路由聲明會使用到的依賴項目

常用於共享業務邏輯、共享數據庫連接、安全性驗證

使用依賴注入，將程式碼重複最小化。

P33

ORM 全名是 Object-Relational Mapping

可以把Python物件跟資料庫做映射

就不用在程式裡寫SQL語句

它的優點有

自動轉換成對應的SQL操作

不是直接下SQL指令，所以可以避免掉SQL Injection

可以讓整體的程式碼較好理解，比較容易維護

缺點

中間需要多一層轉譯，會損失一點效能

P34

SQLAlchemy是一款Python開源軟體

提供SQL工具包及ORM的功能

可以讓我們使用Python的class定義數據庫物件

並自動映射到資料庫

支援多種資料庫

對API Framework相容性也很好

切換資料庫也很方便

但對於比較小的項目來說可能會覺得有點複雜

ORM在處理大量數據的時候

可能會有較大的性能開銷

因為它需要轉換Python對象和數據庫表格之間的映射

P35

到Extension安裝SQLite

P36

我們使用SQLite

建立資料庫test.db

他會去建立資料表

Id會是自增PK

Name 不為Null text

Description text

Price 不為Null Real

On\_offer 是 T F 預設是F

P37

這邊要請大家安裝sqlalchemy aiosqlite

P38

Depends：依賴注入

Column, Integer, String 等：定義資料表欄位型別。

create\_async\_engine：建立 async 用的資料庫引擎。

AsyncSession：非同步版的 Session。

declarative\_base()：用來定義 model（等於宣告資料表用的基礎類別）。

sessionmaker：建立 Session 。

select：非同步 SQL 查詢語法（比傳統 query() 更推薦）。

P39

設定資料庫的連線字串

sqlite+aiosqlite 表示使用 SQLite 資料庫，並透過 aiosqlite 這個非同步的驅動來操作它。

建立一個非同步的引擎，SQLAlchemy 會透過這個引擎跟資料庫連線。

echo=True 表示會在終端機印出 SQL 語句（方便除錯用）。

建立SessionLocal，每次要存取資料庫就從這裡產生一個 Session。

class\_=AsyncSession：表示用非同步的 Session（配合 async/await 用法）。

expire\_on\_commit=False：讓資料在 commit() 之後不會自動過期，這樣你還能存取資料內容。

P40

建立ORM類別

\_\_tablename\_\_ = "fruit"告訴 SQLAlchemy：這個模型對應到資料庫裡的 fruit 這張表。

P41

建立Create Read pydantic 類別

用來驗證使用者送進來的資料

P42

FastAPI 中常見的依賴注入寫法，用來在路由中自動取得資料庫的 Session。

使用 async with 是為了確保 session 用完會自動關閉（避免資源浪費）。

yield 把 session 傳出去，給使用它的路由函式使用。

P43

建立Get Get ByID

回傳使用FruitRead

Tags=Fruit是Swagger分群標籤（不影響功能）

db: AsyncSession = Depends(get\_db)

使用 FastAPI依賴注入

get\_db() 會傳回一個 async 資料庫 session

result = await db.execute(select(Fruit))

執行 SQL 查詢語句

fruits = result.scalars().all()

把所有查詢結果轉成模型物件

P44

實作Post

Refresh 取得寫進資料庫後的完整資料

P45

理論上要建立FruitUpdate Model  
class FruitUpdate(BaseModel):

name: Optional[str]

description: Optional[str]

price: Optional[float]

on\_offer: Optional[bool]

class Config:

orm\_mode = True

P46

204 No Content 是 RESTful 標準的「刪除成功」回應狀態碼。

P47

Postman是後端人員常用的API測試工具