

第2次作業題目-作業-QZ2

學號：112111127

姓名：呂喬詠

作業撰寫時間：180 (mins，包含程式撰寫時間)

最後撰寫文件日期：2025/12/29

本份文件包含以下主題：(至少需下面兩項，若是有多者可以自行新增)

- 說明內容
- 個人認為完成作業須具備觀念
- 1. HTTP Status Code 有哪些？怎麼分類？
- 3位數字??
- WHY??
- 2. 在 Express 中，設計基本上可以分成幾層？

Ans:

1. HTTP Status Code 有哪些？怎麼分類？

HTTP 狀態碼 (**Status Code**) 是伺服器回應請求時，用來表示「結果如何」的 3 位數字，用來表示請求處理的結果與狀態，常見分成 5 大類：

3位數字??

第 1 位 (百位數) -- 「分類 (大方向) 」

- 第一位最重要，用來判斷 請求結果的整體狀態
- 只看第一位，就能先判斷誰該負責

第 2 位 (十位數) -- 「子類型 (原因方向) 」

- 第二位數用來表示「錯或成功的類型」
- 第二位幫助「快速定位錯誤性質」

第 3 位 (個位數) -- 「具體狀況 (細節) 」

- 第三位通常是同一類型下的更細分情況
- 第三位讓狀態碼能精準表達「差一點點但很重要」的差異

1xx – 資訊性 (Informational)

表示伺服器已接收到請求，請求仍在處理中，或需要客戶端繼續操作。

- 100 Continue
 - → 伺服器表示可以繼續傳送請求內容 -- 常用於大型請求

- 101 Switching Protocols
 - → 伺服器同意切換通訊協定 -- 例如 **HTTP** 升級為 WebSocket
- 102 Processing (WebDAV)
 - → 伺服器正在處理請求，尚未完成 -- 避免客戶端誤以為逾時

2xx – 成功 (Successful)

表示請求已成功被伺服器接收、理解並處理完成。

- 200 OK
 - → 請求成功，並回傳請求的資料
- 201 Created
 - → 請求成功，且成功建立新資源 -- 常見於 **POST**
- 202 Accepted
 - → 請求已接受，但尚未完成處理 -- 非同步處理
- 203 Non-Authoritative Information
 - → 回傳的資訊來自第三方或非原始來源
- 204 No Content
 - → 請求成功，但不回傳任何內容 -- 常用於 **DELETE**
- 205 Reset Content
 - → 請求成功，要求客戶端重置畫面或表單內容
- 206 Partial Content
 - → 只回傳部分內容 -- 常用於檔案或影音的分段下載

3xx – 重導 (Redirection)

表示請求的資源需要透過重新導向或其他方式取得。

- 300 Multiple Choices
 - → 有多個可用資源，需由客戶端選擇
- 301 Moved Permanently
 - → 資源已永久移動到新的 **URL**
- 302 Found
 - → 資源暫時移動到其他 **URL**

- 303 See Other
 - → 請使用 GET 方法到另一個URL取得結果
- 304 Not Modified
 - → 資源未修改，可直接使用快取內容
- 305 Use Proxy
 - → 必須透過代理伺服器存取資源 -- 已較少使用
- 307 Temporary Redirect
 - → 暫時轉址，且必須保留原本的 HTTP 方法
- 308 Permanent Redirect
 - → 永久轉址，且必須保留原本的 HTTP 方法

4xx – 客戶端錯誤 (Client Error)

表示請求的資源需要透過重新導向或其他方式取得。

- 400 Bad Request
 - → 請求格式錯誤或參數不正確
- 401 Unauthorized
 - → 未通過身份驗證，需登入或提供認證資訊
- 402 Payment Required
 - → 保留狀態碼，預留給未來付款相關機制
- 403 Forbidden
 - → 已通過身份驗證，但沒有存取權限
- 404 Not Found
 - → 找不到請求的資源
- 405 Method Not Allowed
 - → 該資源不支援使用的 HTTP 方法
- 406 Not Acceptable
 - → 伺服器無法提供符合請求格式的回應
- 407 Proxy Authentication Required
 - → 需通過代理伺服器的身份驗證

- 408 Request Timeout
 - → 客戶端請求逾時
- 409 Conflict
 - → 請求與目前資源狀態發生衝突 -- 如重複資料
- 410 Gone
 - → 資源已永久移除，不再提供存取

5xx – 同伺服器錯誤 (Server Error)

表示同伺服器在處理請求時發生錯誤，導致請求無法完成。

- 500 Internal Server Error
 - → 同伺服器內部錯誤，無法完成請求
- 501 Not Implemented
 - → 同伺服器尚未支援請求的功能或方法
- 502 Bad Gateway
 - → 同伺服器從上游同伺服器收到無效回應
- 503 Service Unavailable
 - → 同伺服器暫時無法處理請求 -- 維護或過載
- 504 Gateway Timeout
 - → 同伺服器等待上游同伺服器回應逾時
- 505 HTTP Version Not Supported
 - → 同伺服器不支援使用的 HTTP 版本

WHY??

為什麼3位數字了，還要分成5大類??

因為三位數本身只是編碼格式，而「5 大類」是用來快速判斷請求結果的意義與責任歸屬，讓人和電腦都能知道發生了什麼事...

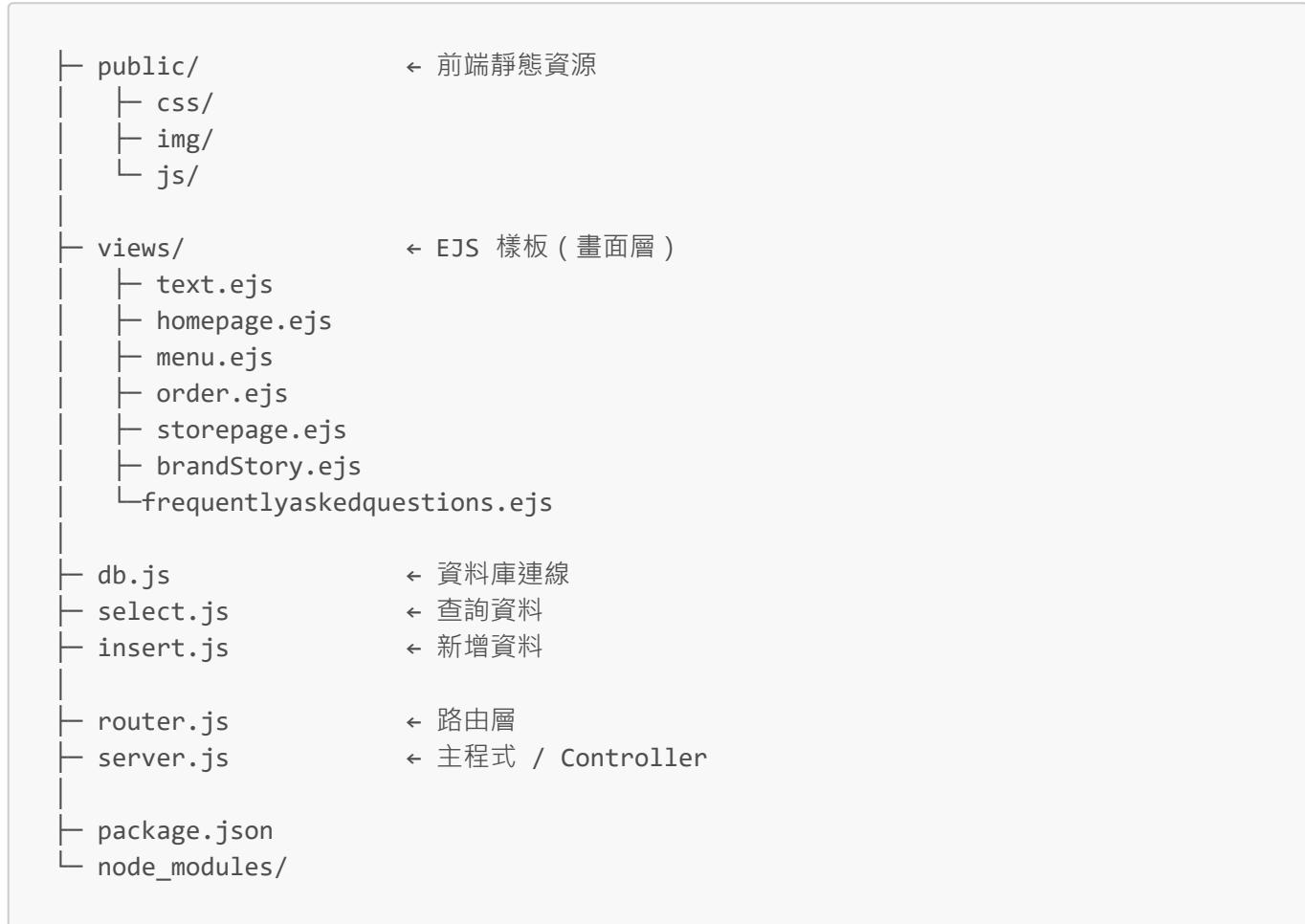
Ans:

2. 在 Express 中，設計基本上可以分成幾層？

一、以本期末專案為例：4 層（後端 3 層 + 前端 1 層）

層級	名稱	對應資料夾 / 檔案
第 1 層	使用者介面層 (View)	views/、public/
第 2 層	Router (路由層)	router.js
第 3 層	Controller (控制邏輯層)	server.js
第 4 層	Model (資料層)	db.js、select.js、insert.js

二、專案結構



三、各層詳細說明

(一) View / 前端介面層

views/

```

text.ejs
homepage.ejs
menu.ejs
order.ejs
storepage.ejs
brandStory.ejs
frequentlyaskedquestions.ejs

```

📁 public/

```
css/  
img/  
js/
```

- 負責畫面顯示
- 使用 **EJS** 樣板引擎

(二) Router 層 (路由分派)

📄 router.js

```
// router.js
const express = require('express');
const router = express.Router();
const { addOrder } = require('./insert');

// 前端頁面路由
router.get('/', (req, res) => res.render('test', { title: '開頭動畫' }));
router.get('/homepage', (req, res) => res.render('homepage', { title: '首頁' }));
router.get('/menu', (req, res) => res.render('menu', { title: '菜單' }));
router.get('/order', (req, res) => res.render('order', { title: '訂購' }));
router.get('/storepage', (req, res) => res.render('storepage', { title: '門市資訊' }));
router.get('/brandstory', (req, res) => res.render('brandStory', { title: '品牌故事' }));
router.get('/frequentlyaskedquestions', (req, res) =>
res.render('frequentlyaskedquestions', { title: '常見問題' }));

// API: 新增訂單
router.post('/api/order', async (req, res) => {
  const { customer_name, phone, pickup_method, cart } = req.body;

  if (!customer_name || !phone || !pickup_method || !cart || !cart.length) {
    return res.status(400).json({ error: '資料不完整' });
  }

  try {
    const order_id = await addOrder(customer_name, phone, pickup_method, cart);
    res.json({ message: '訂單新增成功', order_id });
  } catch (err) {
    console.error("新增訂單失敗(完整錯誤)：" + err);

    // ✅ 除錯用：把 MySQL/程式錯誤回傳給前端 (之後上線再改回簡短訊息)
    res.status(500).json({
      error: '新增訂單失敗',
      code: err.code,
      errno: err(errno),
    });
  }
})
```

```
    sqlMessage: err.sqlMessage,
    sqlState: err.sqlState,
    message: err.message
  });
}

});

module.exports = router;
```

- `router.js` 負責定義系統所有前端頁面與API的路由
- 接收使用者請求並進行基本驗證後，將資料轉交給資料存取層處理
- 本系統中負責「請求分派與流程轉交」的**路由層核心**。

(三) Controller 層 (控制邏輯)

server.js

```
const express = require('express');
const path = require('path');
const app = express();
const port = 3000;

// 解析 JSON
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// 設定 EJS
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

// 靜態檔案
app.use(express.static(path.join(__dirname, 'public')));

// 引入路由
const routes = require('./router');
app.use('/', routes); // 所有路由交給 router.js 管理

// 啟動伺服器
app.listen(port, () => {
  console.log(`伺服器已啟動！請訪問 http://localhost:${port}`);
});
```

```
server.js
├─ 初始化 Express
├─ 設定 Middleware
└─ 設定 View Engine
```

```
└ 提供靜態資源  
  └ 將請求交由 router.js 處理
```

- 初始化**Express**應用程式
 - 建立**Express**應用實例
- 設定**Middleware** -- 請求資料處理
 - 解析前端送來**JSON**與表單資料
 - 確保**API**能正確接收前端資料
- 設定**View**引擎 -- 畫面控制
 - 指定使用**EJS**作為模板引擎
 - 定義**View**檔案存放位置
 - 此設定讓後端能將資料傳遞至前端畫面顯示。
- 管理靜態資源 -- 前端資源控制
 - 提供 CSS、JavaScript、圖片等靜態檔案
 - 分離畫面樣式與後端邏輯
 - 提升系統結構清楚度與維護性
- 掛載路由模組 -- 流程轉交
 - 將所有**HTTP**請求交由**router.js**統一管理
 - 達成「控制層」與「路由層」的職責分離
- 啟動伺服器並對外提供服務
 - 啟動 Web Server
 - 開始監聽指定 Port
 - 提供前後端整合服務

(四) **Model** 層 (資料庫存取)

📄 db.js

- 管理連線

📄 select.js

- 查詢資料

📄 insert.js

- 新增資料