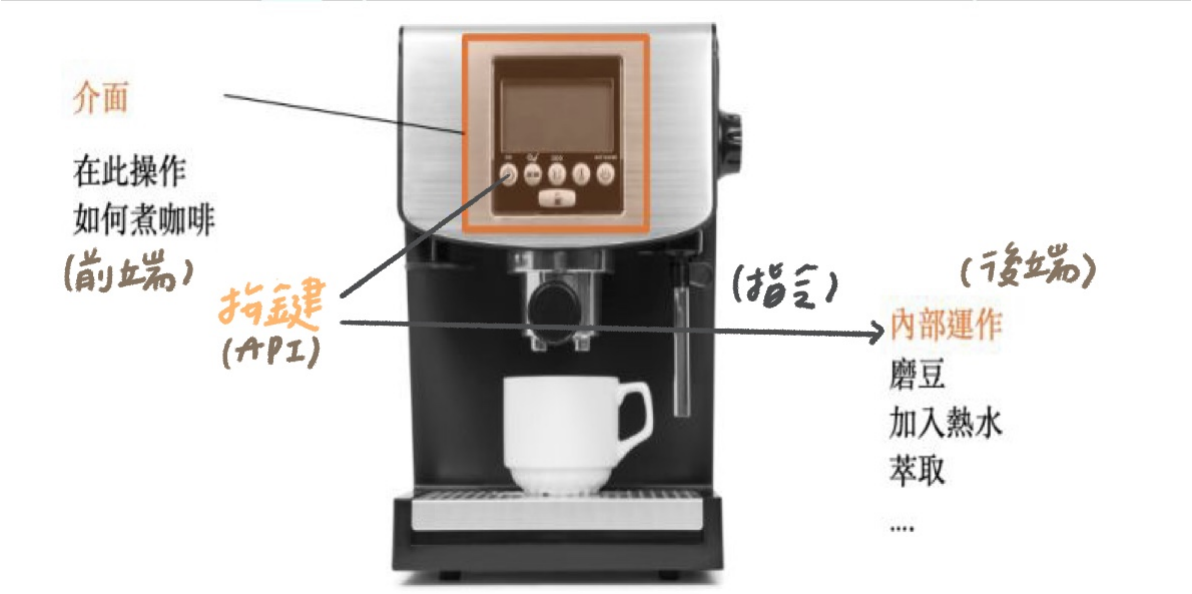
網頁開發大揭秘——後端架構介紹與實例分析

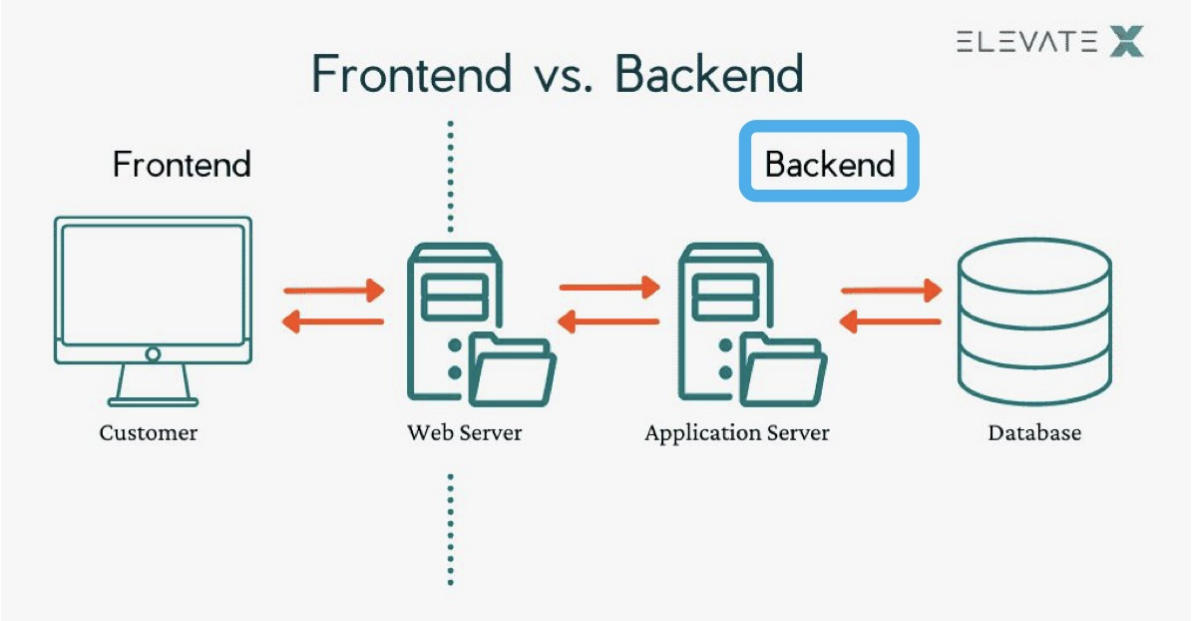
一、網頁前端？後端？差在哪

- 前端開發者
  - 負責建立流暢的使用者介面(可能會由 UI/UX 設計師所提供)
  - 開發網頁的互動功能、特效(e.g. 下拉式選單、彈跳視窗…等)
  - 所需技能: HTML、CSS、Javascript …等等
- 後端開發者
  - 負責資料處理、提高數據處理的效能
  - 依照不同專案的商業邏輯，與資料庫進行互動
  - 所需技能: 程式語言(e.g. Java、C#、Node.js…)、資料庫(e.g. MSSQL、MySQL、MongoDB)…等

網站的前端與後端間通常透過API (應用程式介面, Application Programming Interface)進行溝通



以咖啡機為例：螢幕就是介面，也是前端所顯示的部分，按鈕即是API，前端與後端溝通的橋樑，使用者按下按鈕後會送出指令給後端，咖啡機開始運作，螢幕上也會顯示當前選擇。



二、網站HTTP協定與HTTPS之差異

- HTTP (超文本傳輸協定, HyperText Transfer Protocol)
  - 規範 request、response 物件的內容標準
  - 明文傳遞，有不安全的問題
  - 網址開頭：http://…
- HTTPS (HTTP Secure): 採用 SSL 協定
  - 採用 SSL 協定**加密** HTTP 的傳輸內容，比HTTP協定安全，資料不易洩漏
  - 可以驗證傳輸資料雙方的身份，確保傳輸內容未被篡改
  - 網址開頭：https://…

Request、Response物件

- 根據 HTTP/HTTPS 傳輸協定，Client & Server 雙方會互相傳遞資料，來達成訊息溝通
- Request 物件
  - 由 Client 端發出
  - 包含 HTTP Method、PATH、Headers…等
- Response 物件
  - 由 Server 端回應
  - 包含 Status Code、Status message、Headers…等

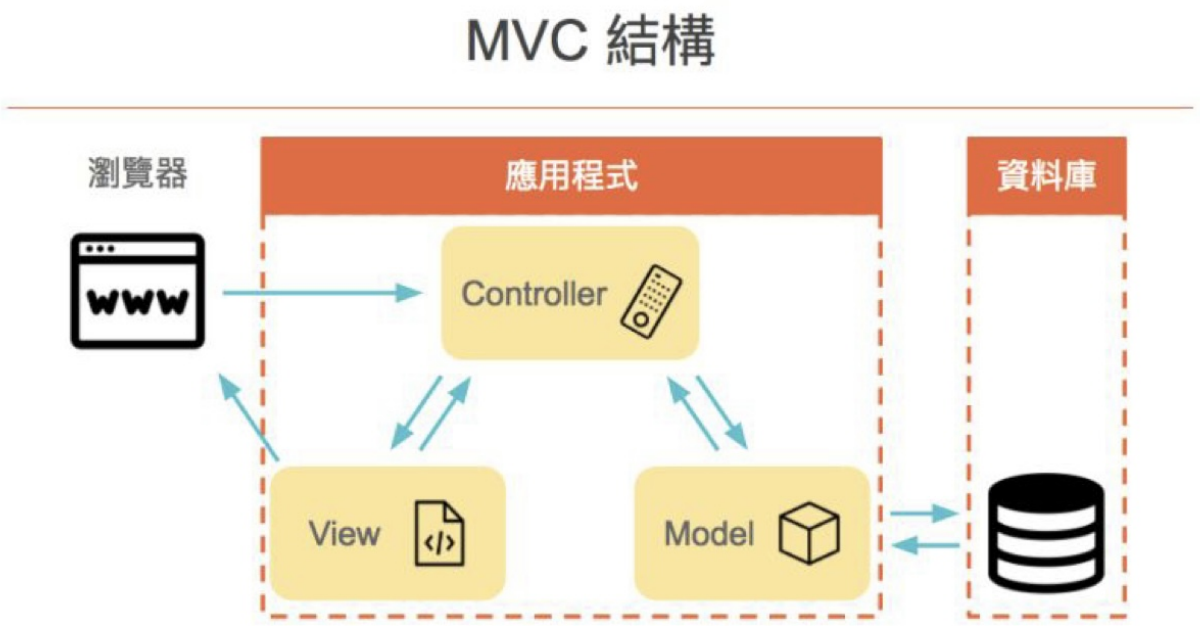
三、路由（Routing）

- Routing(路由): 是把網址 (URL) 轉給特定程序處理的過程，幫助伺服器解析 URL、判斷 URL 路徑與相應的執行動作
- 對 Client 端而言: 就是網址(URL)，它告訴使用者如何從伺服器拿到資料的路徑

- 對 Server 端而言: 它會告訴應用程式什麼時候要做出什麼動作?

四、網頁設計架構——MVC模式

- 將不同意義的程式碼按照功能作分類，使其便於開發、維運
- M (Model): 負責和資料庫溝通
- V (View): 負責管理畫面的呈現
- C (Controller):
  - 掌握使用者互動邏輯，也是應用程式收/發 request、response 物件的核心
  - 為整個MVC架構的中介人，決定應用程式的工作流程，並且搜集不同元件的工作結果，再統一回傳給使用者
- 除了MVC模式外，另有MVP、MVVM模式



五、模組化（Modularize）

- 通常我們會將一個專案拆分成許多功能模組、元件，並且會把一些共用的功能模組化，提供給其他元件使用（這樣不用一直重複寫同樣的程式碼）
- Node.js 有三大類模組
  - Core Module (原生模組)Local Module (本地模組)Third Party Modules (第三方模組)
  - require: 引入模組
  - exports: 匯出模組

六、RESTful APIs

- 一種軟體架構風格
- 功能：定義了一個標準，使在世界各地的軟體、程式能在網際網路中互相傳遞訊息
- 每一個網站都可以視為一個資源(resource)，使用者可以透過 URL 取得這些資源，並在瀏覽器上使用
- 常見的 REST 指令
  - GET (讀取資源)

    例子：使用者想獲取一個資源，發起get請求
  - POST (新增資源)

    例子：在電商網站上註冊、新增一個會員資料時
  - DELETE (刪除資源)
  - …等等

七、View Engine

- **樣板引擎（樣板構造）**
- 可動態產出 HTML 文件
  - 門檻較低
  - 讓程式碼可動態產生你要的畫面
- Header、Footer 都可以用 View Engine 做，並且和網頁互通
- EJS是View Engines的一個函式庫

八、案例講解

▼ 步驟一：設定環境

- 安裝**Node.js**

  https://nodejs.org/zh-tw/download/

  - 下載: 點選 LTS → 自己電腦的作業系統圖示(系統會判斷對應的安裝檔)。或是 Windows 下載 `.msi`，MacOS 下載 `.pkg`
  - 安裝: 打開下載好的安裝包，一直按下一步就好
  - 檢查是否正確安裝(打開終端機):
    - 輸入 `node --version`
    - 正確: 有回應版本號
    - 錯誤: 其它情況
  - 註: 安裝 Node.js 會自動幫忙安裝 npm，所以不用特別去安裝 npm
  - 安裝完成後，開啟終端機(cmd)檢查是否已正確安裝 Node.js, npm 套件管理包工具

- npm 套件管理包工具是參考 `package.json` 這個檔案的內容來下載所有需要使用到的套件 ⇒ `npm install`

> 💡 Node.js基本概念
>
> - 免費、開源、跨平台的 Javascript 執行環境,讓開發者可以在瀏覽器以外也能使用 Javascript；Javascript通常在前端做，但這讓後端也可以寫Javascript
> - 特色:
>   - 單執行緒(single-thread)
>   - 非阻塞 I/O
>   - 一次可以不只執行一個Thread
>   - 廣大的開源函式庫 (npm)
> - 適用情境:
>   - 處理較大吞吐量(throughput)的系統，例如: 影音串流平台(Netflix)
>   - 能同時處理高併發連接(concurrent)的服務，例如: 直播聊天室

> 💡 NPM套件管理包工具
>
> - Node.js 的標準套件管理包工具
> - 用來下載、管理 Node.js 套件與其之間的相依性(dependency)

- 安裝 **Visual Studio Code**

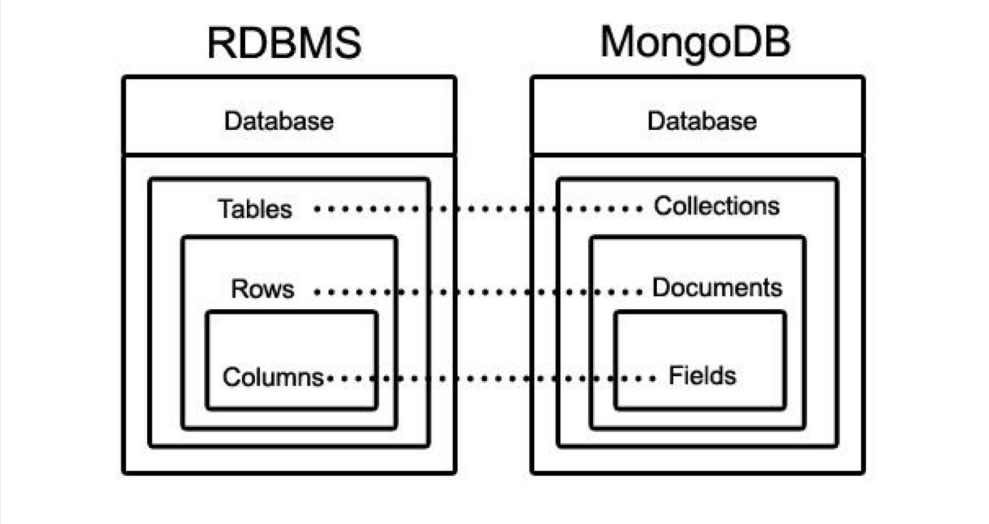
  https://code.visualstudio.com/Download

- 註冊**MongoDB Atlas**

  https://www.mongodb.com/

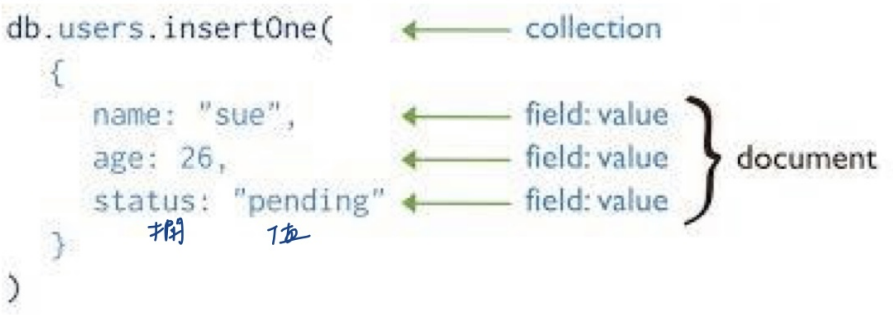
- 註冊一個 MongoDB Atlas 的帳戶
- 建立資料庫
  - Deployment → Database → Build a Database → 選擇 M0 Free 方案 → 選擇 Provider: AWS → 選擇 Region: Hong Kong → Name: `NodeJSCrashCourse` → 點選 Create
- 建立使用者
  - 建立使用者名稱(username)，請牢記!
  - 建立使用者密碼(password)，請牢記!
  - 點選 Create User
- 設定可以存取這個資料庫的 IP 白名單
  - IP Address: **0.0.0.0/0**
  - description: **for testing**
- 點選 Finish and close
- 點選 Go to database

關聯式資料庫 v.s. 非關聯式資料庫



- 關聯式資料庫（RDBMS）
  - 像是表格一樣，有Tables、Rows（列）、Columns（欄）
- 非關聯式資料庫
  - MongoDB是一種常見的非關聯式資料庫
  - 底下是巢狀結構

雖然架構不同，但結構可以相互對應



- 非關聯資料庫：每個大括弧都是一個 Document，裡面有「鍵」跟「值」

💡 Model、ODM介紹

- **Model：會去跟資料庫溝通**
- **ODM（Object Data Modeling）**
  - 與資料庫對接的方式有兩種
    - 原生 SQL 語言
    - 使用 ORM/ODM，將網站資料轉換成 Javascript 物件，再對映到資料庫
  - 舉例：ORM（Sequelize）、ODM（Mongoose）
- Mongoose：ODM 的函式庫
- 我們操作 Model → 要寫對應的 User
  - 對應真實的 Table，間接進行控制
- **Schema（綱要）**
  - 類似骨架
  - 定義汽車的綱要：要有哪些零件來做出這台車？
- **Model（模型）**
  - 由 Schema 所產生的模型
  - 類似汽車的模具
- **Entity（實體）**
  - 利用 Model 創建的實作
  - 真的做出一台實體的汽車

▼ 步驟二：從Git clone檔案過去

- 檢查 Git 是否已正確安裝: `git --version`

    - 正確: 有回應版本號
    - 錯誤: 其它情況
- 若之前沒有下載 Git，可以到官方網站下載
    - 網址: https://git-scm.com/downloads
    - 下載: 點選自己對應的作業系統
    - 安裝: 一直按下一步就好
- 使用 `git clone` 把典翰 GitHub 的專案拉下來

> 指令: `git clone https://github.com/Hans-Tsai/Node.js-crash-course.git`

▼ 切至Ch07

- `git checkout ch07`：切換到 ch07 這個 branch
- `app.js` 中的 `app.set('view engine', 'ejs');` ⇒ 設定 view engine 為 ejs 模板引擎
- `./views/` 資料夾中的內容就是前端畫面要顯示的頁面
- `./views/partials/` 資料夾中的頁面就是在這個部落格中，可以共同使用的部分，例如: Header, Footer

▼ Ch08

- `git checkout ch08`：切換到 ch08 這個 branch
- `app.js` 中的第 15 ~ 31 行，就是 Node.js 採用 Express 框架，執行中介函式的寫法

▼ Ch09

- `git checkout ch09`：切換到 ch09 這個 branch

blog.js在實作model

- 主要在做 Model
- 引入 `mongoose` 套件
- 用模具建立 Blog 的實體
    - 建立完實體後要匯出

💡 Model 是間接操作資料庫

▼ CH11

- `git checkout ch11`：切換到 ch11 這個 branch

Controller 和 Routes

- 引入 router 後(`express.Router()`)，宣告路由器的名稱叫做 `router`
  - router 導流到對應的 path

> 💡 `router.get('/create', blogControllers.blog_create_get);`
> 舉例說明: 當 client 端從瀏覽器的網址列輸入 http://localhost:3000/create，並發出 GET 請求的 Request 到 Server 端時，Server 端要到 `blogControllers` 找出 `blog_create_get` 這個函式來處理這個請求，再回覆給 Client 端

  - router 最後也需要匯出
- blogController.js
  - 引入 models / blog
  - blog_details 的概念就是從前端傳過來的 id (`req.params.id`)，透過間接操作

▼ Wrap up

- `git checkout main`：切換到 main這個 branch
- 修改 `app.js` 的 `dbURI`

修改欲連線的資料庫位址
Step 1: 到 MongoDB Atlas，點選左側 Deployment → Database → Connect → Connect your application → 找到中間有一段 `mongodb+srv://` 開頭的那段，將整段複製
Step 2: 貼到專案資料夾中 `app.js` 中 `dbURI` ，修改成自己的資料庫帳號、密碼
例: `const dbURI = 'mongodb+srv://hans:LlcUxUUgvc4Hy26S@nodejscrashcourse.yukds.mongodb.net/node-tuts?retryWrites=true&w=majority';`

- 安裝所有需要用到的套件: `npm install`
- 在 Visual Studio Code 開啟終端機
  - 從上排導覽列 ⇒ 新增終端機 (待會所有關於終端機的指令都在這邊輸入)
  - 可以使用 `pwd`（macOS） / `chdir`（Windows）查看目前資料夾位置，目前所在位置需位於 `Node.js-crash-cour` `e/` 資料夾中，才是正確的
  - 若不是位於正確位置，找出自己下載 `Node.js-crash-cour` `e/` 的所在位置(e.g. 下載 資料夾中)，再於 Visual Studio Code 終端機中，前往該正確路徑
    例如: `cd /Users/hans/Node.js-crash-course/`
- 啟動 Node.js 後端伺服器
  - 指令: `node app.js`
- 檢視部落格
  - 打開瀏覽器，在網址列輸入 `http://localhost:3000`
- 終止 Node.js 後端伺服器
  - 在 Visual Studio Code 的終端機輸入 `ctrl + c`

補充

- app.js：應用程式的起點
  - 在這邊匯入 express , mongoose , blogRoutes 模組
  - `express.static` ：把靜態檔案放在特定的地方
- `redirect（頁面路徑）` 可以讓我們導引到一個新的連結
  - 因為我們在 views 資料夾裡寫了 about page，因此可以 render 它
  - 可以在 router 的斜線後面打變數，Ex：`/blog:id`
- 使用 `app.listen(3000)` 讓網頁幫你開啟 3000 port