排版整理如下：

**抽象方法 vs 虛擬方法：**

* **抽象方法**：
  + 使用 **abstract** 關鍵字聲明，強制衍生類別提供實作。
  + 相當於一個謎題，需要自行找出答案。
* **虛擬方法**：
  + 使用 **virtual** 關鍵字聲明，不強制覆寫。
  + 相當於故事中的選擇，可以決定是否改變它。
  + 讓程式更有彈性，可以根據需要自行調整。

**Cookie vs Session vs JWT：**

* **Cookie**：
  + 以 key-value 的方式儲存在瀏覽器內，儲存在 Client。
  + 包含用戶的識別標識、偏好設置、網站訪問記錄等。
* **Session**：
  + 紀錄 Server 與 Client 狀態的一種機制。
  + 基於 Cookie 產生出 Session 與 SessionId，Session 儲存在 Server，SessionId 儲存在 Client。
  + 伺服器端保存用戶的相關資訊。
* **JWT**：
  + 基於資料的驗證機制，用戶身份驗證資訊被加密後存儲在令牌（token）中，然後在客戶端持久保存。
  + 不需要在伺服器上存儲任何資訊。
  + 包含標頭（Header）、聲明（Payload）和簽名（Signature）。

**HTTP 方法（GET、POST、PUT、PATCH、DELETE）比喻：**

* **GET**：像是寄送明信片，資料附加在 URL 上，容易被人看到，適合獲取資源。
* **POST**：像是寄送密封信，資料藏在請求主體中，安全性較高，適合新增資源或傳送私密資料。
* **PUT**：修改資源，若原本無資料則新增。
* **PATCH**：修改資源的部分內容。
* **DELETE**：刪除資源。

**PUT vs PATCH：**

* 使用 **PUT** 方法更新資料時，即使某些資料沒有要更新，也會一併更新。
* 使用 **PATCH** 方法更新資料時，僅更新指定的資料，未指定的部分不會被修改。

**ASP .NET Core MVC 生命週期：**

1. **Middleware**：包含已註冊的 Middleware，依序執行功能。
2. **Routing**：根據設定好的路由尋找對應的 Controller Action。
3. **Controller Initialization**：處理 Controller 初始化的步驟。
4. **Action Method**：執行 Action 中的邏輯行為。
5. **Result**：回傳結果至 View，顯示在前端頁面上。

**Razor 語法：**

* Razor 語法以 **@** 符號開頭，在 View 中使用，方便而直觀。

**.NET Core DI：**

* .NET Core 使用內建的 DI Framework，命名空間為 **Microsoft.Extensions.DependencyInjection**。
* 在 **startup.cs** 的 **ConfigureServices** 方法中註冊服務。
* 支援 Transient、Scoped、Singleton 三種生命週期。

**ASP.NET Core 簡介：**

* Web 專案透過 **Program.Main()** 中的 **CreateHostBuilder()** 來建立及設定。
* **Startup**：
  + **ConfigureServices**：註冊應用程式所需的服務到 DI 容器中，只執行一次。
  + **Configure**：設定 ASP.NET Core 如何回應用戶端要求，使用 **IApplicationBuilder** 配置 Request Pipeline。

## 如何防範XSS

## 限制Cookie存取

上面的範例也可以看到透過XSS存取Cookies，為了不讓XSS有機會讀到敏感資料並傳送回去攻擊者的伺服器，我們可以設定Cookie存取的限制，使Cookies只能允許在Http傳輸使用  
Startup.ConfigureServices中加入