# ASP.NET web form

的軟體生命週期（Software Lifecycle）指的是一個 ASP.NET 應用程序從開始建立到結束運行的過程。這個過程包括了開發、部署、運行和維護等各個階段。下面是 ASP.NET 應用的主要生命週期階段：

**1. 開發階段（Development Phase）**

在這一階段，開發人員會使用 Visual Studio 或其他工具進行應用程式的編碼、測試、調試等工作。開發階段包括以下幾個重要的步驟：

* **編寫代碼**：編寫 ASP.NET 的控制器、頁面、服務等代碼。
* **設計 UI**：設計頁面樣式、UI 控件等。
* **測試和調試**：確保代碼無錯誤，進行單元測試和集成測試。
* **配置**：配置應用程序的設定，例如 web.config 或 appsettings.json。

**2. 部署階段（Deployment Phase）**

部署是指將開發完成的應用程序部署到 Web 伺服器上，使其能夠對外提供服務。這一階段通常包括以下步驟：

* **編譯和打包**：將應用程序編譯成可以部署到生產環境的格式，這可能是 DLL 文件、靜態資源、配置文件等。
* **部署到伺服器**：將編譯好的應用程序部署到 Web 伺服器上，這可以是 IIS（Internet Information Services）或其他支持 ASP.NET 的伺服器。
* **配置伺服器**：配置伺服器的相關設定，如應用程序池、端口、憑證等。

**3. 執行階段（Execution Phase）**

在應用程序運行時，ASP.NET 會依照請求的不同，通過處理請求、頁面生命周期、資源管理等多個步驟來生成頁面並返回給用戶。這個過程的主要步驟如下：

* **HTTP 請求到達**：用戶通過瀏覽器發送 HTTP 請求，伺服器收到請求後開始處理。
* **請求處理**：ASP.NET 處理請求並選擇適合的處理程序（如 Web Form、MVC 控制器、Web API 等）。
* **生命週期管理**：ASP.NET 會依照一定的順序執行頁面生命週期中的事件（如頁面初始化、頁面加載、頁面呈現等）。這是 ASP.NET 生命週期的核心部分。
* **頁面渲染**：將服務端的數據或控件渲染成 HTML，並發送回客戶端。
* **HTTP 回應發送**：渲染結果通過 HTTP 回應發送回用戶的瀏覽器。

**4. ASP.NET 頁面生命週期**

這是 ASP.NET 中一個非常重要的概念，指的是從用戶發送請求到收到回應的整個過程。ASP.NET 頁面生命週期主要包括以下幾個階段：

* **Page Request**：當請求到達時，ASP.NET 檢查是否已經存在該頁面的緩存，如果有則直接返回。
* **Page Initialization**（初始化）：在這個階段，頁面的所有控件都會被創建，但還沒有設定任何屬性。
* **Page Load**（加載）：在這個階段，頁面和控件的屬性會被設置為來自用戶請求的值。這是進行數據綁定和其他初始化操作的地方。
* **Postback Event Handling**（回傳事件處理）：如果頁面是來自用戶操作的回傳（如表單提交），則在此階段處理相應的事件（如按鈕點擊事件）。
* **Rendering**（渲染）：在這個階段，頁面會生成最終的 HTML 標記，這些 HTML 標記會傳送到客戶端的瀏覽器。
* **Unload**（卸載）：在頁面生命周期結束時，ASP.NET 會釋放頁面和控件的資源。

這些階段確保了頁面能夠正確地處理來自用戶的請求，並生成相應的回應。

**5. 維護階段（Maintenance Phase）**

維護階段包括了應用程序運行後的更新、升級、修復和性能優化等工作。這些工作通常涉及以下內容：

* **錯誤修復**：修復代碼中的錯誤或漏洞。
* **性能優化**：通過代碼優化、SQL 查詢優化等方式提高應用程序的性能。
* **更新和升級**：添加新功能或升級現有的功能，並確保應用程序與最新的技術或版本兼容。
* **監控**：持續監控應用程序的運行狀況，確保其穩定性和安全性。

**6. 退役階段（Retirement Phase）**

當應用程序不再需要或已經過時時，它會進入退役階段。這一階段通常包括以下幾個步驟：

* **停用服務**：停止應用程序的服務，確保用戶無法再訪問。
* **數據遷移**：將應用程序中的重要數據轉移到新的系統或儲存位置。
* **清理資源**：從伺服器中刪除應用程序的文件、資料庫或其他資源。

**結語**

ASP.NET 的軟體生命週期涵蓋了開發、部署、運行、維護和退役等多個階段。每一個階段都對應著不同的工作流程和技術，開發人員需要充分了解每一個階段的特點，才能有效地開發和管理一個 ASP.NET 應用程序。

# Asp.net mvc

按下執行應用程式(發出第一次請求)=>

Web伺服器(IIS)會為應用程式創建一個應用程序域並啟動(Web.config為中心), 當應用程緒終止, 相關的應用程序都會被卸載釋放

1. **請求到達 (Request Arrival)**

**客戶端發送請求**

**2. 路由處理 (Routing)**

在請求到達應用程式後，ASP.NET MVC 會使用路由來解析 URL 並決定哪個控制器（Controller）和動作方法（Action）來處理這個請求。

* **路由表（Route Table）**：在應用啟動時（通常是在 Global.asax 的 Application\_Start 方法中），MVC 設置了一個路由表，這個路由表定義了 URL 模式和對應的控制器、動作方法。例如：

RouteTable.Routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

* **路由匹配**：當請求進來時，路由會檢查請求 URL 是否符合路由表中的模式，並且根據匹配的路由信息選擇相應的控制器和動作方法。

**3. 控制器實例化 (Controller Instantiation)**

一旦路由決定了哪個控制器和動作方法來處理請求，ASP.NET MVC 會創建該控制器的實例，並準備執行對應的動作方法。

* 控制器通常是由 **控制器工廠**（IControllerFactory）創建的。MVC 默認使用 DefaultControllerFactory，會根據請求 URL 中的控制器名稱來實例化控制器。

## Action Execution

**4. Model Binding**

控制器被創建後，Model Binding會根據動作方法的參數，將來自 HTTP 請求的數據自動映射到這些參數上。

* **動作方法的參數綁定**：在動作方法執行之前，ASP.NET MVC 會根據請求的參數（如查詢字符串、表單數據、路由參數等）自動將這些數據綁定到動作方法的參數中。例如：

public ActionResult Index(int id)

{

// id 會自動從路由或查詢字符串中綁定

}

* **模型綁定**：如果動作方法的參數是複雜類型（例如，模型對象），MVC 會嘗試自動將請求的數據綁定到模型的屬性。

**5. 過濾器 (Filters)**

在動作方法執行之前或之後，ASP.NET MVC 會根據配置執行一些過濾器（Filters）。過濾器是可以在不同階段插入的邏輯，通常包括以下幾種類型：

* **Authorization Filters**：用來處理授權邏輯，檢查用戶是否有權訪問該控制器或動作方法。
* **Action Filters**：在動作方法執行之前或之後執行。可以用來做一些額外的處理，如記錄日誌、檢查足夠的授權　或　修改請求數據、處理異常。
* **Result Filters**：在動作結果（Action Result）返回給用戶之前執行。可以用來修改或處理即將發送給用戶的結果。
* **Exception Filters**：用來捕獲動作方法中的異常並處理。

**6. 動作方法的返回結果 (Action Result Execution)**

當動作方法執行完畢後，通常會返回一個 **ActionResult** 物件。這個物件表示該請求的處理結果。

* **常見的 ActionResult 類型**：
  + ViewResult：返回一個視圖（即頁面）。
  + RedirectResult：重定向到另一個 URL。
  + JsonResult：返回 JSON 格式的數據。
  + ContentResult：返回純文本內容。
  + FileResult：返回文件下載。
  + EmptyResult：表示無返回結果。

## 返回view

**7. 視圖渲染 (View Rendering)**

如果動作方法返回的是 ViewResult，ASP.NET MVC 會將控制器的資料傳遞到對應的視圖（View），並渲染視圖。這個過程會包含以下步驟：

* **查找視圖**：MVC 會根據路由和控制器的名稱查找對應的視圖文件。默認情況下，視圖文件放置在 Views/{ControllerName}/{ActionName}.cshtml。
* **渲染視圖**：視圖會被渲染成 HTML，並與控制器提供的數據一起生成最終的 HTML 頁面。

**8. 返回結果給客戶端 (Returning Response to Client)**

當視圖渲染完成後，ASP.NET MVC 將最終生成的 HTML、JSON、檔案或其他結果返回給客戶端（通常是瀏覽器）。這會結束請求的處理過程，並將結果顯示給用戶。

**9. 結束請求 (Request End)**

請求的處理結束，控制權會返回給 Web 伺服器（如 IIS），並且該請求的所有處理（包括所有中介階段，如過濾器等）都已完成。

**ASP.NET MVC 的頁面生命週期簡要總結：**

1. **請求到達** - 用戶發送請求，Web 伺服器接收請求。
2. **路由處理** - 根據 URL 配置的路由，決定哪個控制器和動作方法處理請求。
3. **控制器實例化** - 根據路由選擇並創建控制器實例。
4. **動作方法執行** - 執行控制器中的動作方法，並進行參數綁定。
5. **過濾器執行** - 按順序執行過濾器，進行授權檢查、日誌記錄、錯誤處理等。
6. **動作結果處理** - 動作方法返回結果，如視圖、重定向、JSON 等。
7. **視圖渲染** - 如果是視圖結果，則渲染相應的視圖。
8. **返回結果給客戶端** - 將結果（通常是 HTML）返回給瀏覽器或其他客戶端。
9. **請求結束** - 請求的生命周期結束。

搜尋引擎最佳化：如果第一次請求是由後端生成前端畫面

那麼搜尋引擎一開始就找不到你的網頁　因為點進去之前是一片空白