

認識 ASP.NET Core 7.0 的啟動設定檔 (Launch Profile)

由於 ASP.NET Core 7.0 與 ASP.NET Core 6.0 的專案範本中的**啟動設定檔**(Launch Profile)有些許不同，而我最近的 ASP.NET Core 課程剛好有學員提到這個問題，所以我就來整理一下相關知識，幫助大家更好的認識這個 Properties\launchSettings.json 檔案。

先看看 Properties\launchSettings.json 的差異

我先用 dotnet new mvc 在 ASP.NET Core 6.0 與 ASP.NET Core 7.0 產生一樣的專案：

```
mkdir m1
cd m1
dotnet new globaljson --sdk-version 6.0.100 --roll-forward latestFeature
dotnet --version
dotnet new mvc

mkdir m2
cd m2
dotnet new globaljson --sdk-version 7.0.100 --roll-forward latestFeature
dotnet --version
dotnet new mvc
```

由於專案範本不同，專案內的 Properties\launchSettings.json 檔案內容其實不太一樣：

- ASP.NET Core 6.0

```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:22303",
      "sslPort": 44378
    }
  },
  "profiles": {
    "m1": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "applicationUrl": "https://localhost:7007;http://localhost:5183",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}
```

ASP.NET Core 7.0

```

• ASP.NET Core 7.0
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:9601",
      "sslPort": 44330
    }
  },
  "profiles": {
    "http": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "applicationUrl": "http://localhost:5041",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "https": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "applicationUrl": "https://localhost:7068;http://localhost:5041",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  }
}

```

從上述內容你可以很清楚的發現 ASP.NET Core 6.0 與 ASP.NET Core 7.0 有以下 2 點差異：

1. ASP.NET Core 6.0 的 profiles 只有兩個，但是 ASP.NET Core 7.0 的 profiles 卻有三個！
2. ASP.NET Core 6.0 的第一個 Profile 預設名稱為「專案名稱」(m1)，而 ASP.NET Core 7.0 的第一個 Profile 名稱確是 http，而且還多出一個 https 設定檔。

其實在 ASP.NET Core 6.0 以前，所有的專案範本中第一個 Profile 預設名稱都為「專案名稱」，所以其實大家對這個名字沒有什麼感覺，直到 ASP.NET Core 7.0 做出了改變，這才有人發現原來這東西的用途。

不載入啟動設定檔

從 ASP.NET Core 6.0 開始，透過 dotnet new 專案範本所建立的 ASP.NET Core 專案，預設都不會繼續使用早期的 Port 5000 (http) 與 Port 5001 (https)，可能是因為多專案的架構下太容易衝突，而且太多人不清楚啟動設定檔的存在，所以就改成用亂數的 Port 埠號。

但是亂數的 Port 埠號其實很難寫文件，所以我為了讓學員檢測自己是否有成功裝好 .NET SDK 工具，我會這樣寫：

```
# 建立並進入資料夾
mkdir m1 && cd m1
# 建立 ASP.NET Core MVC 專案範本
dotnet new mvc
# 啟動網站並忽略啟動設定檔
dotnet run --no-launch-profile
```

這裡的 `--no-launch-profile` 就是避免 `dotnet new` 去讀取專案中的 `Properties\launchSettings.json` 啟動設定檔，而 .NET Runtime 的 `dotnet run` 命令，在沒有載入啟動設定檔的情況下，在 ASP.NET Core 6.0 與 ASP.NET Core 7.0 有細微的差異：

- 預設 ASP.NET Core 6.0 網站會監聽 Port 5000 (http) 與 Port 5001 (https)
- 預設 ASP.NET Core 7.0 網站會監聽 Port 5000 (http)

由於 ASP.NET Core 的 TLS 自簽憑證預設只有一年效期，每年都要更新，雖然更新很簡單，命令如下：

```
dotnet dev-certs https --clean
dotnet dev-certs https --trust
```

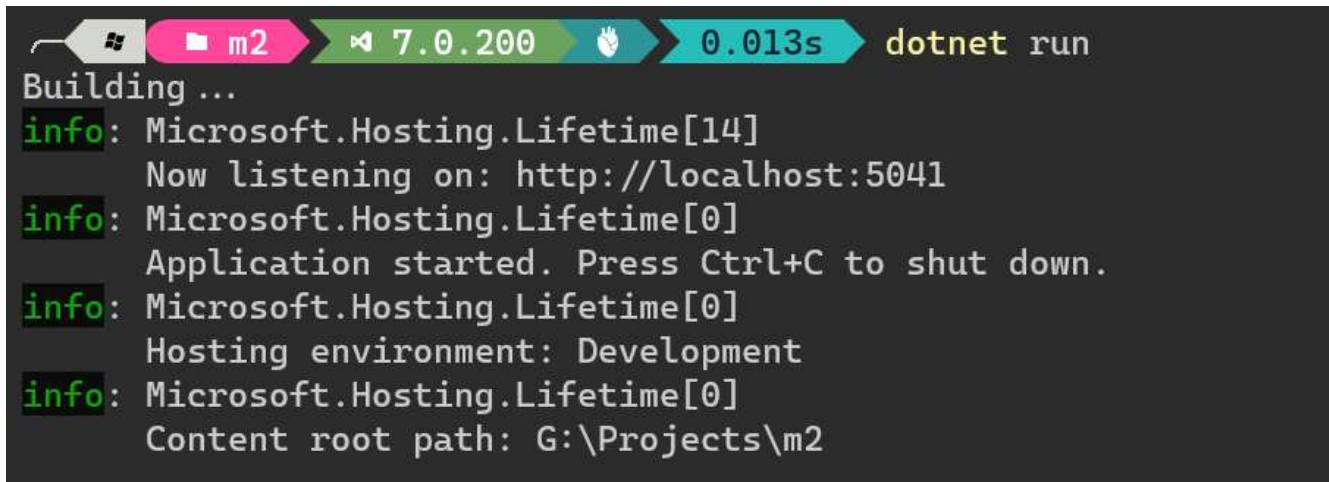
我覺得應該是有很多人搞不定 https 問題，才讓微軟決定降低開發門檻，直接把預設 https 的 binding 移除！

啟動設定檔的用法

我們在執行 `dotnet run` 的時候，只要沒特別指定**啟動設定檔**，預設就會自動載入 `Properties\launchSettings.json` 設定檔中的第一個 `profiles` 中的設定，由於 ASP.NET Core 7.0 的「第一個」是 http 這個名字：

```
"http": {
  "commandName": "Project",
  "dotnetRunMessages": true,
  "launchBrowser": true,
  "applicationUrl": "http://localhost:5041",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Development"
  }
}
```

所以當網站啟動時，我們的 `applicationUrl` 預設就只有 http 的網址而已，因此網站啟動時並不會有 https 可以測試！



```
m2 7.0.200 0.013s dotnet run
Building ...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5041
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: G:\Projects\m2
```

那如果我們也想測試 https 網址怎麼辦，這時你就可以選擇 `Properties\launchSettings.json` 設定檔中的第二個 `profiles` 中的設定：

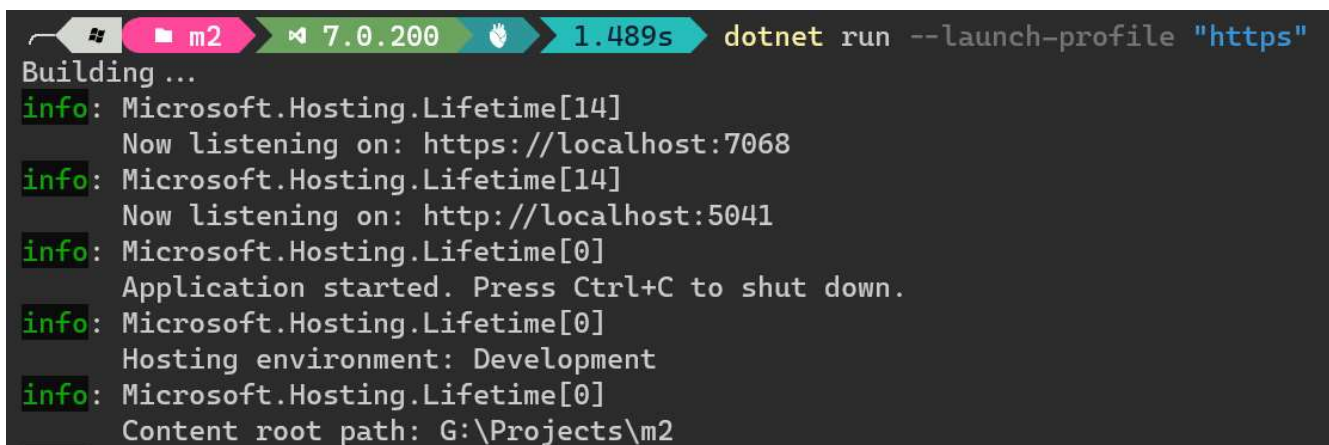
```
"https": {
  "commandName": "Project",
  "dotnetRunMessages": true,
  "launchBrowser": true,
  "applicationUrl": "https://localhost:7068;http://localhost:5041",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Development"
  }
}
```

其啟動的方式如下：

```
dotnet run --launch-profile "https"
```

或者可以簡寫成這樣：

```
dotnet run -lp "https"
```



```
m2 7.0.200 1.489s dotnet run --launch-profile "https"
Building ...
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: https://localhost:7068
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://localhost:5041
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Development
info: Microsoft.Hosting.Lifetime[0]
      Content root path: G:\Projects\m2
```

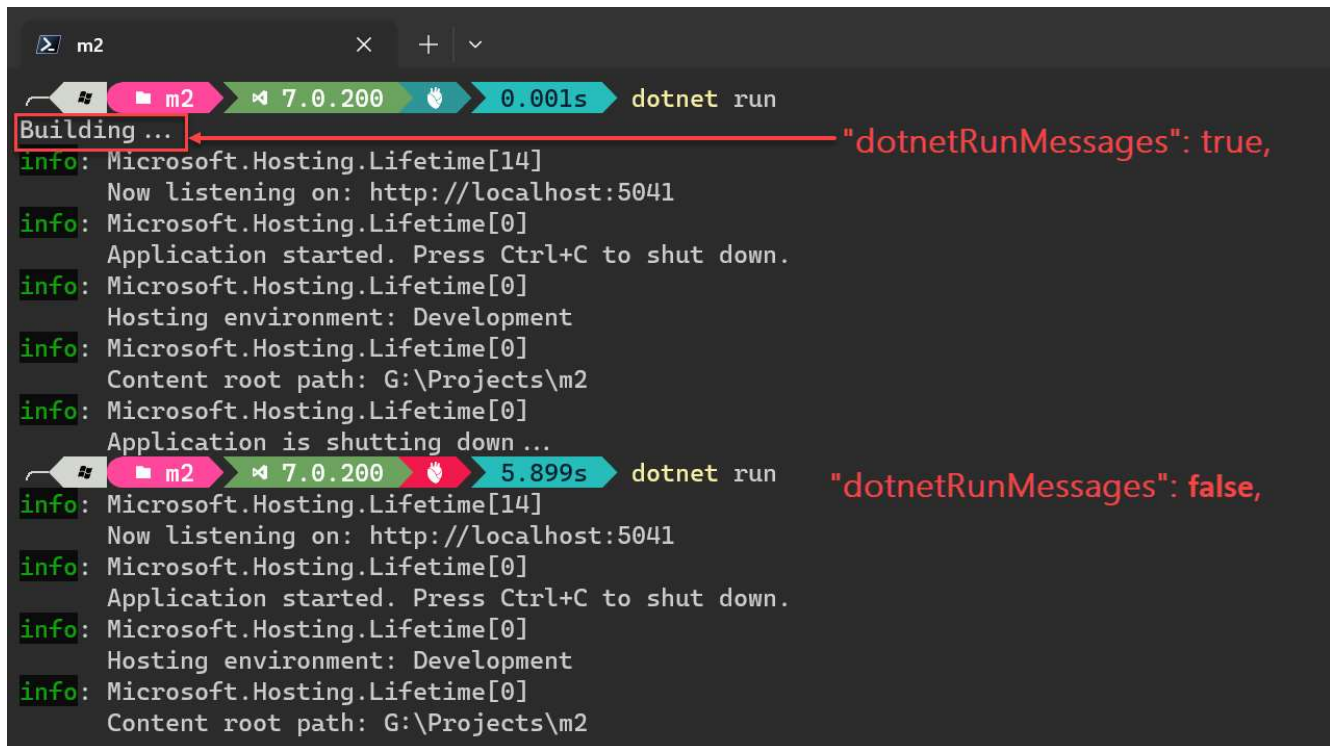
深入瞭解啟動設定檔

在啟動設定檔中有個 `launchBrowser` 屬性，預設為 `true`，從字面上來看，就是「啟動瀏覽器」的意思。但其實我們用 `dotnet run` 並不會因為 `launchBrowser` 為 `true` 就開啟網頁，而是要透過 `dotnet watch` 命令啟動網站才會真的啟動瀏覽器並開啟網頁！

另一方面，你可能會發現我們在 Visual Studio 2022 裡面每次啟動網站都會開啟網頁，這時我其實也會覺得這個設計很煩，每次都一直不斷的開啟新頁籤，其實蠻惱人的。而關掉的方法，就是調整啟動設定檔中的 `launchBrowser` 屬性，調整為 `false` 即可！

```
"https": {
  "commandName": "Project",
  "dotnetRunMessages": true,
  "launchBrowser": false,
  "applicationUrl": "https://localhost:7068;http://localhost:5041",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Development"
  }
}
```

另外還有一個 `dotnetRunMessages` 也是個謎一般的存在，完全沒有文件說明，而這個設定也確實沒太多存在感！各位可以看下圖差異之處，只有在 `dotnet run` 啟動的時候有非常細微的差異而已。我們在透過 `dotnet run` 啟動網站時，由於要先 `dotnet build` 過才能真正 Run 起來，而這段時間大概會跑個 2~3 秒左右，如果在 `dotnetRunMessages` 為 `true` 時，在 Console 會優先出現一個 `Building...` 字樣，讓你知道 `dotnet run` 沒掛掉，他只是還在跑而已，否則你會有 2~3 秒的空白，有人會覺得好像程式沒啟動，感覺怪怪的！😬



最後，還有個 `commandName` 也是謎一般的存在，依然完全沒有文件說明，但在 Stack Overflow [有篇文章](#)有做詳細說明，其設定值有可能是以下四種：

- Project

將當前的這個應用程式作為控制台應用程式運行，這的設定跟你用 `dotnet run` 執行是相同的。

- IIS Express

預設使用 IIS Express 開發伺服器來執行 ASP.NET Core 應用程式。

這個選項只能用在 Visual Studio 而已，.NET CLI 是不支援的！

- IIS

當你在本機安裝 IIS 後，您可以讓您的應用程式直接在 IIS 執行。

這個選項只能用在 Visual Studio 而已，.NET CLI 是不支援的！

- Executable

這讓你可以執行任意執行檔、傳入任意參數，非常靈活的設定！

這個選項只能用在 Visual Studio 而已，.NET CLI 是不支援的！

例如你不想用 Visual Studio 2022 內建難用的 Hot Reload 的話，可以這樣設定：

```
"Watch": {  
  "commandName": "Executable",  
  "dotnetRunMessages": true,  
  "executablePath": "dotnet",  
  "commandLineArgs": "watch run",  
  "workingDirectory": "${ProjectDir}",  
  "launchBrowser": true,  
  "applicationUrl": "https://localhost:5001;http://localhost:5000",  
  "environmentVariables": {  
    "ASPNETCORE_ENVIRONMENT": "Development"  
  }  
}
```

相關連結