* 設置錯誤Response訊息

於Startup.cs -> Configure函式中 -> 加入 app.UseExceptionHandler(“/Error”)

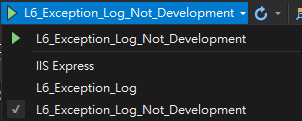
這樣就可以在發生Exception將頁面導向<https://localhost/Error>當中



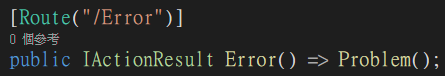
於Properties->launchSettings.json新增一個環境，並設定ASPNETCORE\_ENVIRONMENT為非研發環境



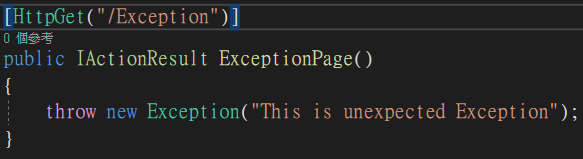
設定Debug環境為新定義的環境



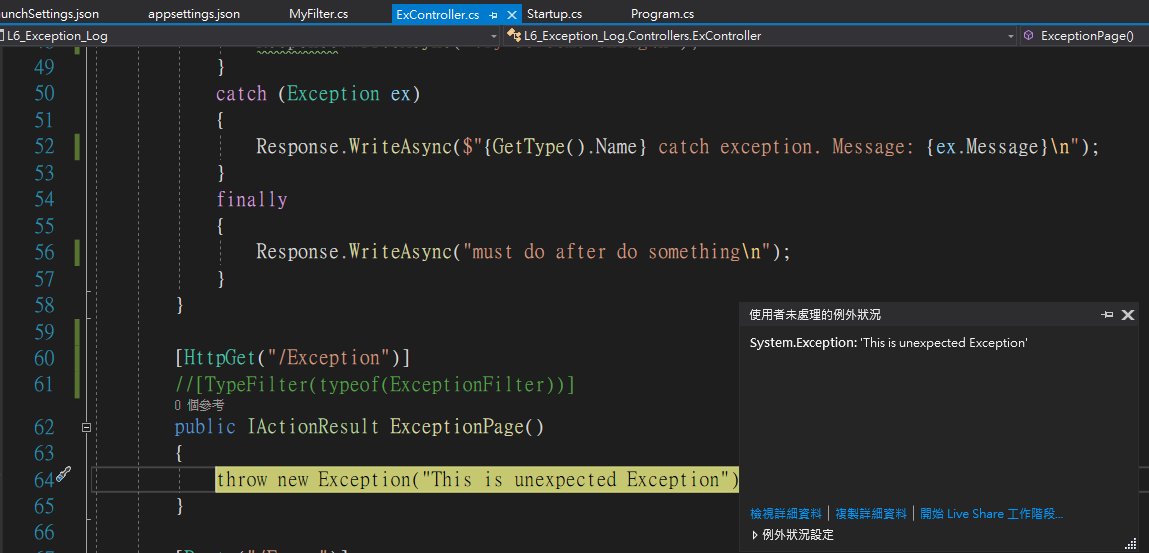
建立一個ExContoller.cs並加入/Error頁面如下 (Error頁面可以自行定義)

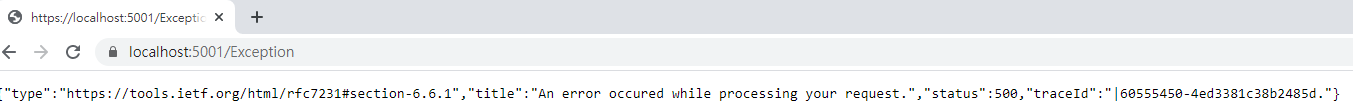


定義產生Exception的頁面如下



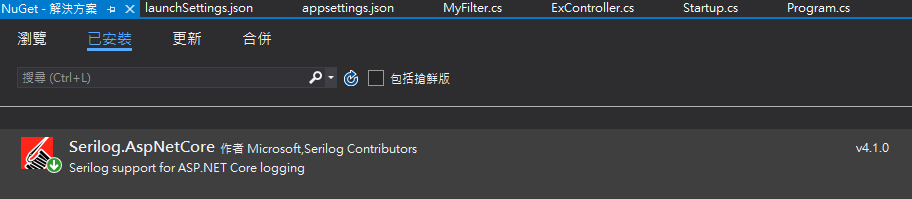
於瀏覽器輸入localhost/Exception測試有沒有成功產生Exception





* 進行Log運行紀錄

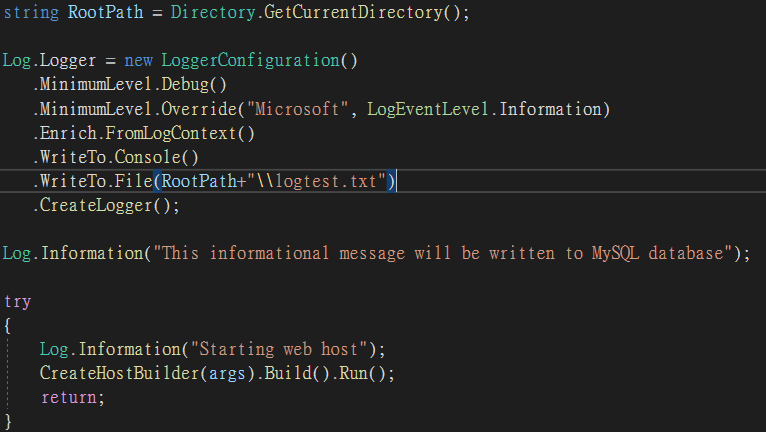
由NuGet安裝Serilog.AspNetCore套件



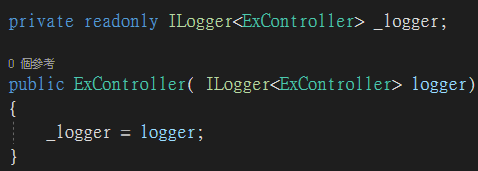
於Program.cs->Main函式中加入以下段落其中

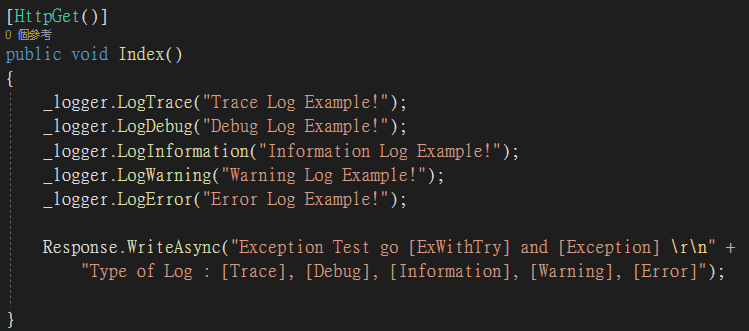
MinimumLevel代表會顯示的Log等級: [Trace]>[Debug]>[Information]>[Warning]>[Error]

WriteTo.Console()與WriteTo.Console()則為Log會輸出到終端與指定檔案中



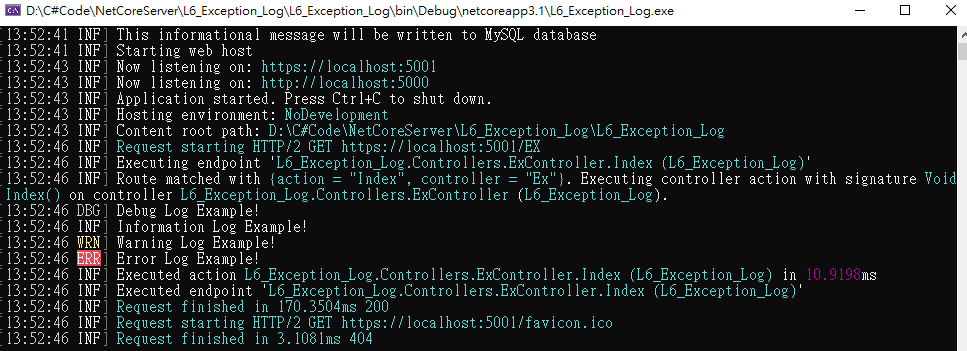
在ExController當中注入Logger實體並定義輸出的Log



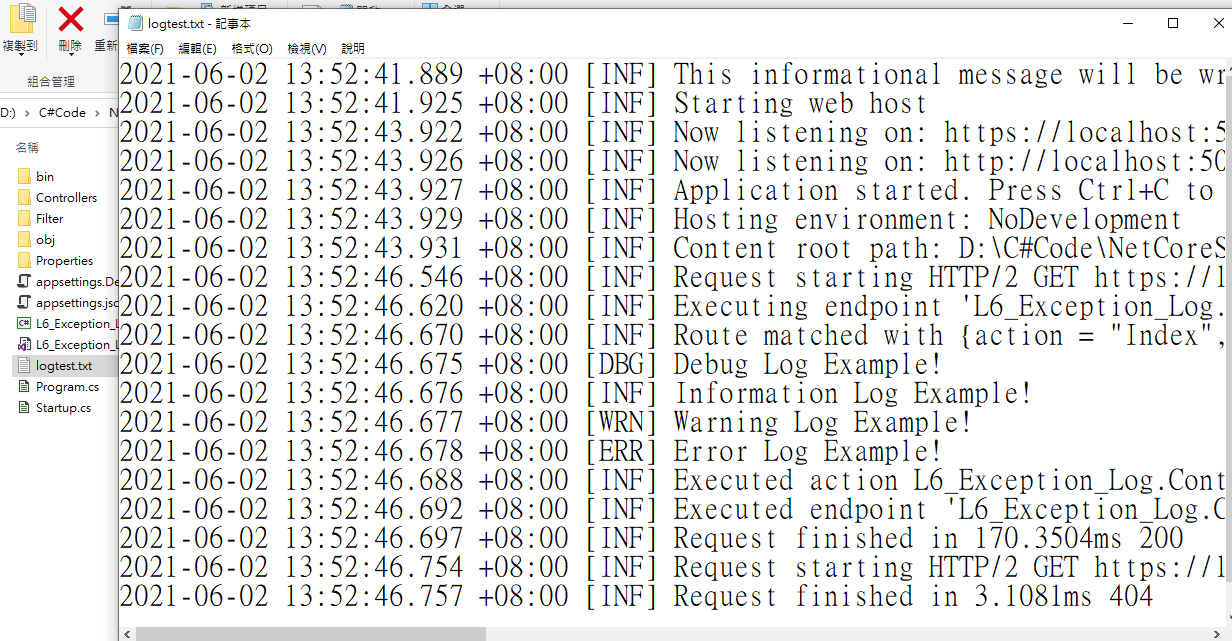


進行運行測試

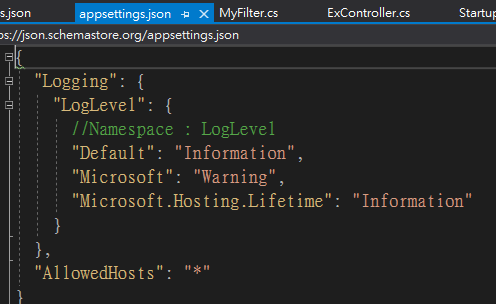
Console:



LogFile檔案

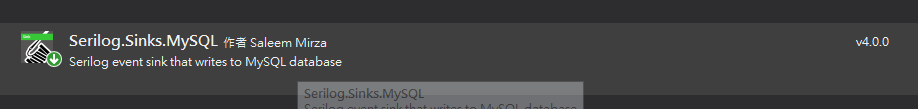


其中Logger相關設定可以在appsetting.json找到可以設定不同namespace的LogLevel

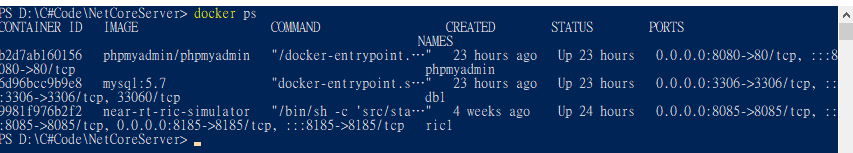


* Logger With Mysql

由NuGet安裝Serilog.Sinks.MySQL套件

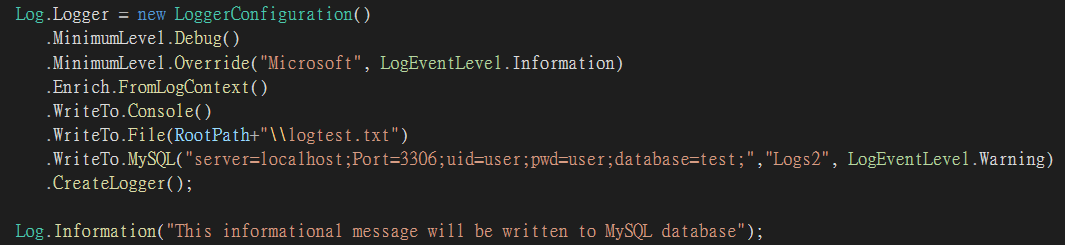


從Docker架好Mysql以及UI



於program.cs當中的new LoggerConfiguration()當中加入

.WriteTo.MySQL(“DatabaseConfiguration”, “Log記錄的Table\_Name”, 記錄的LogLevel)



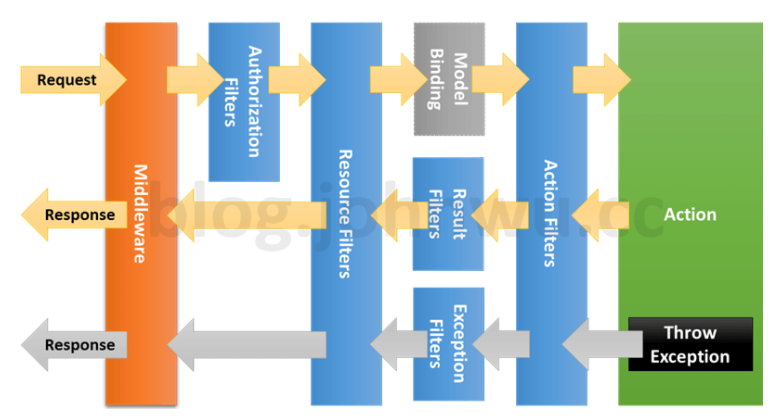
運行並去資料庫UI確認有資料被記錄



* Filter

Filter就是每次運行API (Action)時中間會經過的處理層具體種類與運行如下

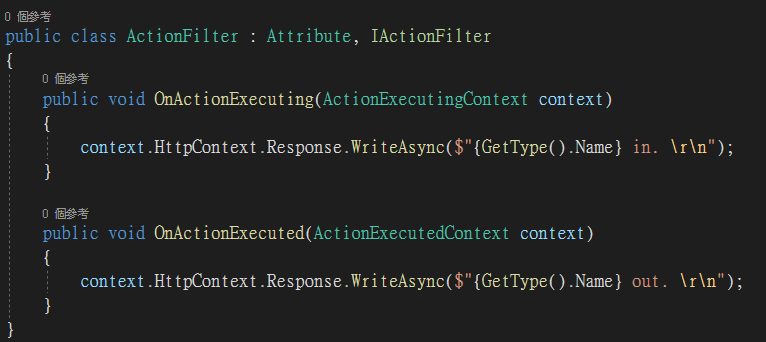
* **Authorization Filter**  
  Authorization 是五種 Filter 中優先序最高的，通常用於驗證 Request 合不合法，不合法後面就直接跳過
* **Resource Filter**  
  Resource 是第二優先，會在 Authorization 之後，Model Binding 之前執行。通常會是需要對 Model 加工處裡才用
* **Action Filter**  
  最容易使用的 Filter，封包進出都會經過它，使用上沒特別需要特別注意的。跟 Resource Filter 很類似，但並不會經過 Model Binding
* **Exception Filter**  
  異常處理的 Exception。
* **Result Filter**  
  當 Action 完成後，最終會經過的 Filter。



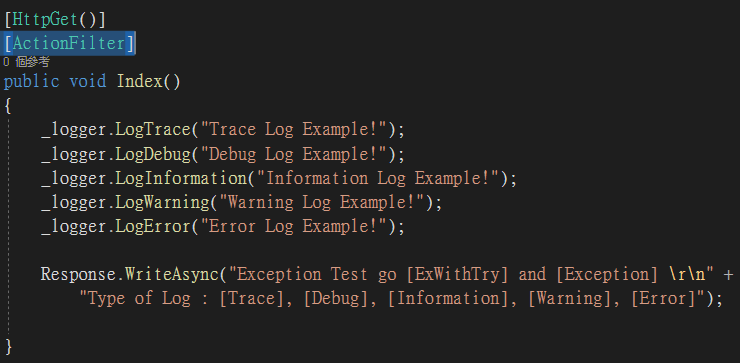
新增Filter資料夾並加入MyFilter.cs



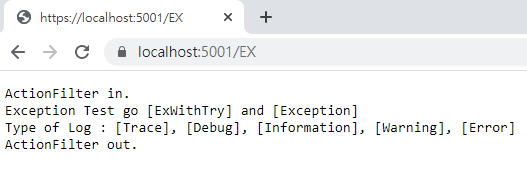
撰寫ActionFilter如下



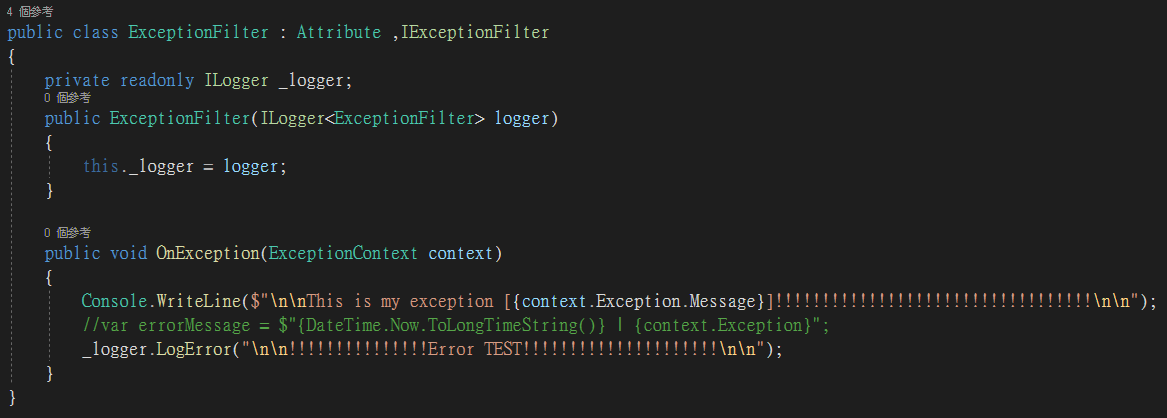
於ExController.cs加入ActionFilter標籤



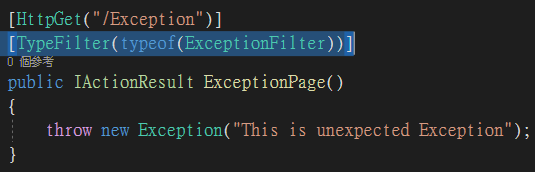
運行該頁面時就可以看到ActionFilter有被運行



撰寫ExceptionFilter如下



加入ExceptionFilter的標籤



運行時使用瀏覽器連入localhost /Exception頁面

就可以在Console與資料庫看到該Filter有確實被運行

