不存在的路由，轉移的頁面要修改

Session.Clear跟Session.Abandon 清除資料跟清除SessionId的時機點跟差異

實現角色的搜尋功能權限時 有兩種方式

第一個是只要有分項目功能就給予此權限

第二個是在功能判斷只要有以下權限就可以使用

Set @www = (select .. from..)時如果資料是多筆會返回甚麼?

技術債?破窗效應?

session 欄位設定

uuid是甚麼? UID要改一下名稱(最好都全稱)=>UserId

導覽頁面的大項目轉導到子項目的功能 可能不需要=>改成類似(必須展開大項目 再點開子項目)

細項目名稱 跟內容標題一致

創建使用者失敗 下拉選單有bug

查詢的排序功能 (根據uid, 狀態, 角色)

刪除功能 跳提示窗

防護快速點擊按鈕(ex:新增..)

角色禁用之後 必須檢查 帳號是否可以繼續使用(依據狀態檢查)

禁用跟刪除 之間的關係 (設計的想法) (有可能再刪除帳號的時候, 其他table有連結此帳號的資料時, 可能會出問題)

被踢掉之後需要提示為何被踢掉

下次Demo：修改密碼、角色的刪除、更改功能（細修其他東西）

不存在的路由處理

web.config配置

優點：易於管理且配置簡單

全局異常處理 (Global.asax)

捕獲未處理異常(包括404)

優點：可以捕獲所有異常並個別處理，還可使用記錄錯誤等等

在 RouteConfig.cs 設置

直接自定義所有無效路由的處理邏輯

Session.Clear跟Session.Abandon 清除資料跟清除SessionId的時機點跟差異

Clear清除資料時機：執行當下就清除

Abandon廢棄資料時機：session跟資料隨著下個生命週期開始才會清除(因為發現對應sessionId是無效或廢棄)

Abandon廢棄sessionId時機：執行時會標記sessionId失效，而新的sessionId在下次請求時分配

技術債

技術債指的是開發人員為了加速軟體開發，忽略長期效益及品質，改以短期能加速開發的方案，導致未來要花更多時間、技術、資源來修補額外開發負擔，就像一筆債務一樣，看似得到好處但未來得償還。

**EX：**當需要改一個功能，發現無法只改一個功能不影響其他，這代表耦合性過高，程式碼太複雜，需降低耦合性

**避免：具備測試概念、不定期檢查清除技術債（EX：不適用於新環境的功能）、避免過早「最佳化」**

破窗效應

結論：為了修補破窗，應該盡量避免加速開發軟體導致產生技術債

描述寫程式的五個階段：

1. 程式開始出現小問題，因不影響程式運行所以未解決，但部分正常的程式與問題有輕微的關聯。
2. 未解決的小問題被放置，且開發者不關心，正常的程式也暫時不受影響。
3. 問題越來越多，開發者也越來越隨便，開始出現幾個小問題聯合形成的中問題。
4. 越來越多正常的程式跟問題產生連結，程式越來越難修改。
5. 問題沒有解決並且持續變多到最終影響功能，卻無從下手修改。

禁用跟刪除之間的關係

1. 在刪除帳號時，應該確認刪除該帳號的所有相關資料，或者進行資料初始化

2. 因帳號是一個非永久標示符，應根據唯一識別符識別資料而不是帳號

　3. 若無刪除，須隔離舊帳號資料

禁用帳號就可以避免以上這些問題，並且保留客戶資料

實現角色的權限(搜尋連動性)