|  |
| --- |
| 速達三代Application Framework 說明 |
| V 1.0 |
| 陳鈺杰 |

[一、 Framework架構說明 2](#_Toc369271217)

[1、 MVC分層 2](#_Toc369271218)

[2、 APF元件專案分層 3](#_Toc369271219)

[3、 APF網頁專案分層 3](#_Toc369271220)

[二、 資料夾結構說明 4](#_Toc369271221)

[1、 PIC.HOBBIT.CORE 4](#_Toc369271222)

[2、 SUDA3G.APF 5](#_Toc369271223)

[3、 SUDA3G.APF.WEB 5](#_Toc369271224)

[三、 Database Table Schema說明 7](#_Toc369271225)

[四、 共用API 呼叫說明 8](#_Toc369271226)

[1、 PIC.HOBBIT.CORE 8](#_Toc369271227)

[2、 SUDA.APF 8](#_Toc369271228)

[五、 SSO介接說明 9](#_Toc369271229)

[1、 CAS Client for ASP.NET元件建置 9](#_Toc369271230)

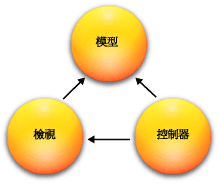
[2、 ASP.NET MVC與CAS Server設定 15](#_Toc369271231)

[3、 子模組Server取得SSO認證資訊 21](#_Toc369271232)

[六、 版本控管使用說明 22](#_Toc369271233)

# Framework架構說明

## MVC分層



模型-檢視-控制器 (Model-View-Controller，MVC) 三層式架構模式可將一個應用程式劃分成三個主要元件：模型、檢視和控制器。 ASP.NET MVC 架構為 ASP.NET Web Form 模式提供建立 Web 應用程式的替代方式。

Model：模型物件屬於實作應用程式資料網域邏輯之應用程式的一部分。 通常，模型物件會擷取和儲存資料庫中的模型狀態。 例如，Product 物件可能從資料庫擷取資訊、操作該資訊，然後將更新的資訊寫回 SQL Server 資料庫的 Products 資料表。

Controller：控制器就是元件，可以處理使用者互動、使用模型並且在最後選擇可以轉譯要顯示 UI 的檢視。 在 MVC 應用程式中，檢視只會顯示資訊；控制器則會處理和回應使用者輸入和互動。 例如，控制器會處理查詢字串值，並且將這些值傳遞至模型，接著模型可能會使用這些值查詢資料庫。

View：檢視是顯示應用程式中使用者介面 (UI) 的元件。 通常此 UI 是從模型資料建立。 例如 Products 資料表的編輯檢視，它會根據 Product 物件目前的狀態顯示文字方塊、下拉式清單和核取方塊。

## APF元件專案分層

有鑑於MVC應用於商業開發之需求，故將Model層多建了一層方法層(Repository)以提供資料存取層作用，並在其上延伸增加商業邏輯層(BIZ)以實作商業應用之邏輯程式；故APF元件模組會分為如下分層：

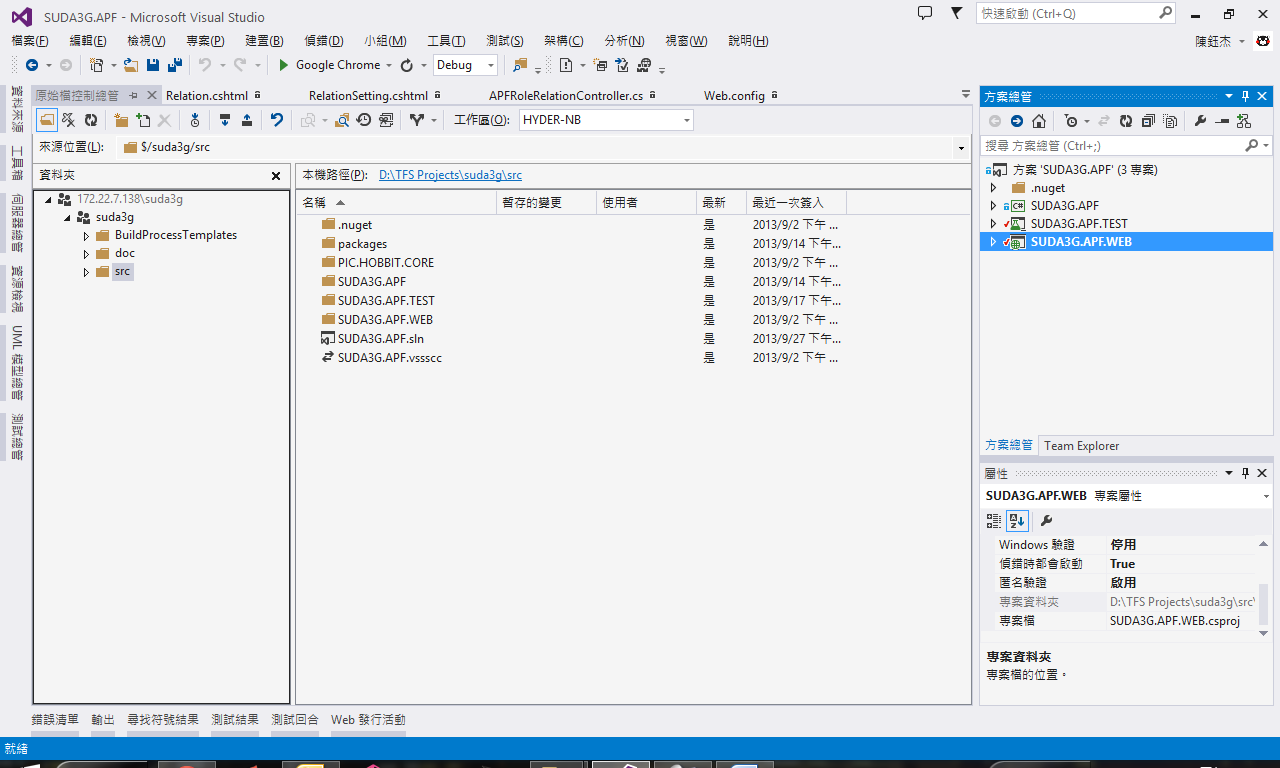


## APF網頁專案分層

為了讓前端頁面的資料模型符合前端需求，並能實現前端與後端平行開發，故在APF網頁專案多新增一層ViewModel(檢視資料模型層)供給前端使用：



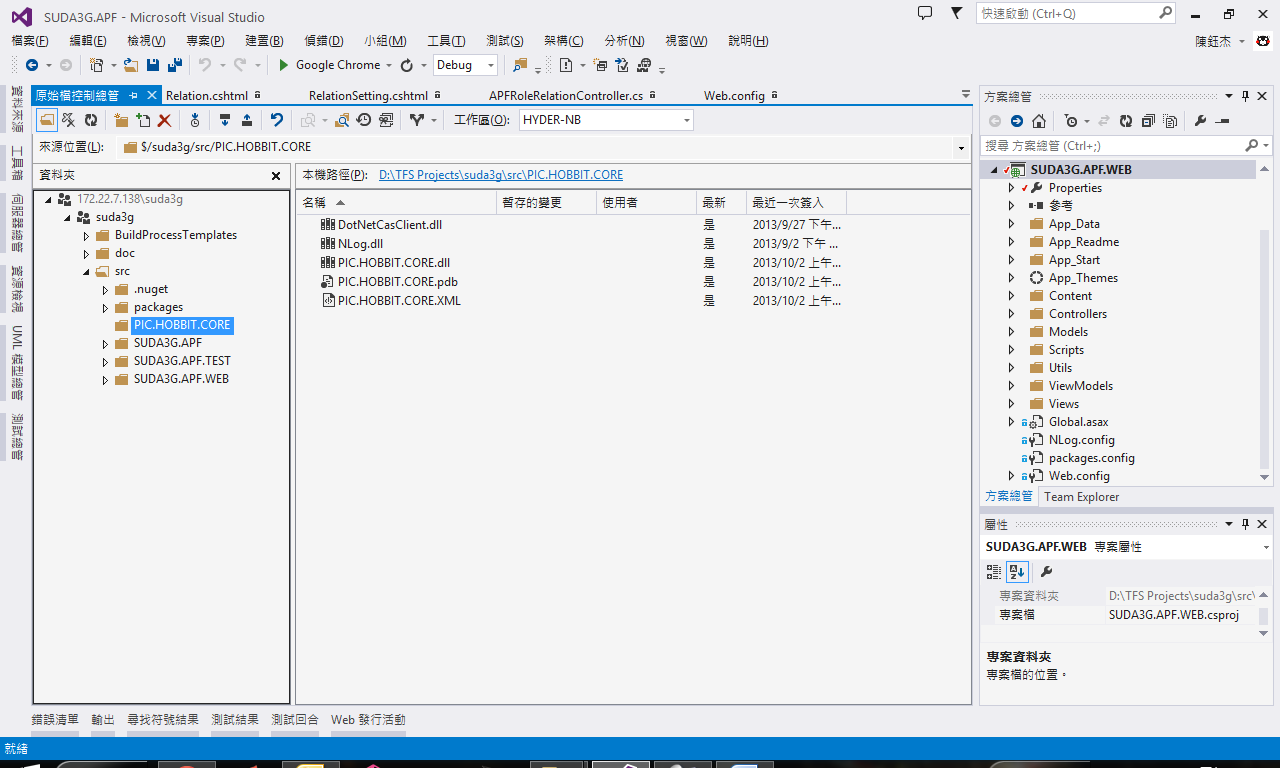
# 資料夾結構說明



如上圖所示，方案資料夾分為：

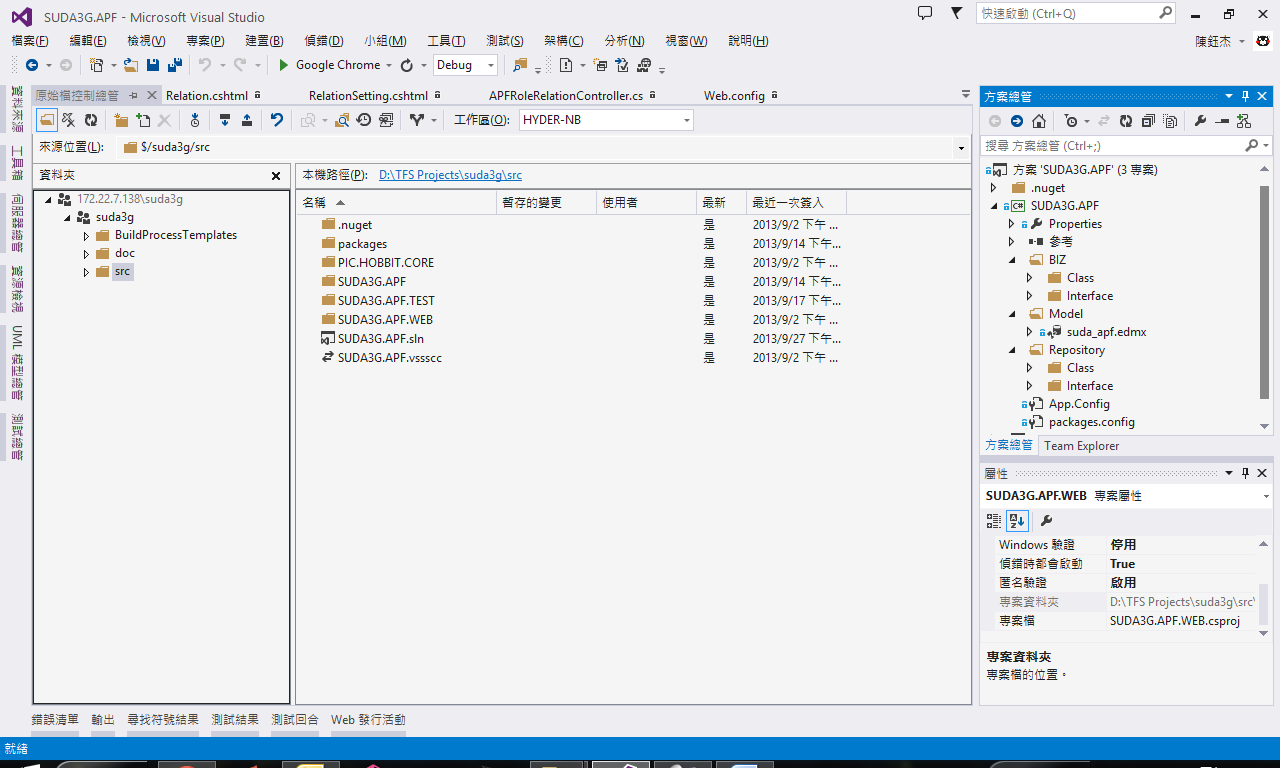
1. PIC.HOBBIT.CORE：為PIC公司所提供之基盤API元件庫
2. SUDA3G.APF：為速達三代APF框架之API元件庫專案
3. SUDA3G.APF.TEST：為速達三代APF框架測試專案
4. SUDA3G.APF.WEB：為速達三代APF網頁專案

## PIC.HOBBIT.CORE



主要存放PIC公司所提供之基盤API及SSO元件庫的dll檔

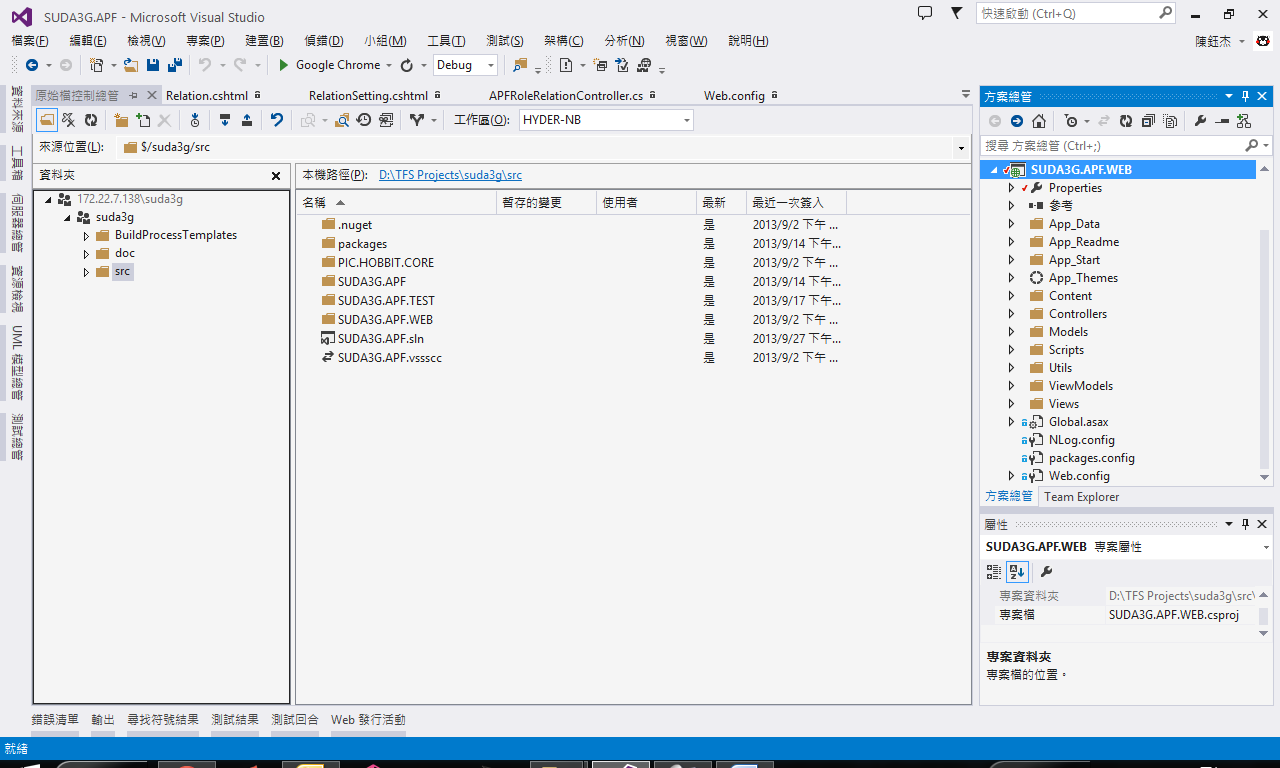
## SUDA3G.APF



如上圖所示，SUDA3G.APF專案資料夾分為：

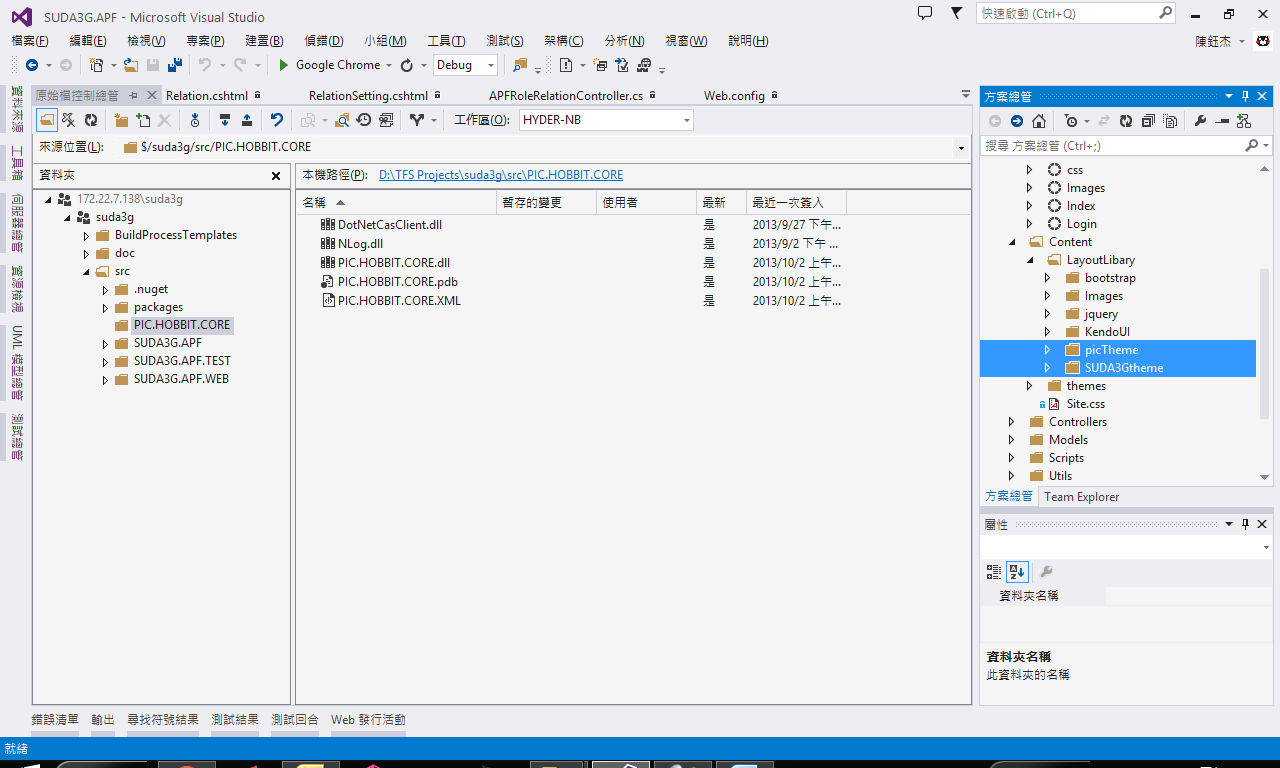
1. BIZ：存放商業邏輯層程式並提供前端Web叫用之API
2. Model：存放由EntityFramework從資料庫轉換的資料模型
3. Repository：存放資料存取層程式，主要提供資料庫方法給BIZ呼叫

## SUDA3G.APF.WEB



如上圖所示，SUDA3G.APF.WEB專案的重要資料夾分為：

1. App\_Start：存放ASP MVC重要設定檔
2. App\_Themes：存放原始ASP MVC的主題相關靜態檔案，如Image、CSS
3. Content：存放APF所採用之主題相關靜態檔案，如Image、CSS，主要存放於picTheme及SUDA3Gtheme兩個資料夾：



1. Controllers：存放ASP MVC架構中的Controller程式
2. Models：存放ASP MVC架構中的資料模型，不過現階段APF已另外拆至SUDA.APF的專案當中
3. Scripts：存放Javascript程式
4. Utils：存放Web所使用的工具程式
5. ViewModels：存放用來溝通Controller及View之間的資料模型
6. Views：存放ASP MVC架構中的View網頁程式

# Database Table Schema說明

|  |  |
| --- | --- |
| 表格名稱 | 說明 |
| PIC公司所提供之基盤Table | |
| Core\_User | 使用者帳號 |
| Core\_UserProfile | 使用者資訊 |
| Core\_User\_RoleRelation | 使用者與角色關聯 |
| Core\_Role | 角色 |
| Core\_Menu\_RoleRelation | 選單與角色關聯 |
| Core\_Menu\_Root | 選單根節點 |
| Core\_Menu\_Node | 選單子節點 |
| Core\_Menu\_Action | 選單子節點對應功能 |
| Core\_Controller | 功能 |
| Core\_Permission | 功能與角色關聯 |
| Core\_SerialNumber | 序號 |
| Core\_Log | 日誌 |
| 速達三代APF所提供之Table | |
| click\_log | 點擊率 |
| login\_log | 登入日誌 |
| apf\_menu\_role\_relation | APF選單與角色關聯 |
| apf\_menu\_root | APF選單根節點 |
| apf\_menu\_node | APF選單子節點 |
| apf\_controller | APF功能 |
| apf\_permission | APF功能與角色關聯 |
| apf\_role\_profile | APF角色資訊 |
| apf\_user\_role\_block | APF使用者與角色關聯之排除名單 |

# 共用API 呼叫說明

## PIC.HOBBIT.CORE

請參閱PIC\_Framework2.0\_API\_使用文件.docx

## SUDA.APF

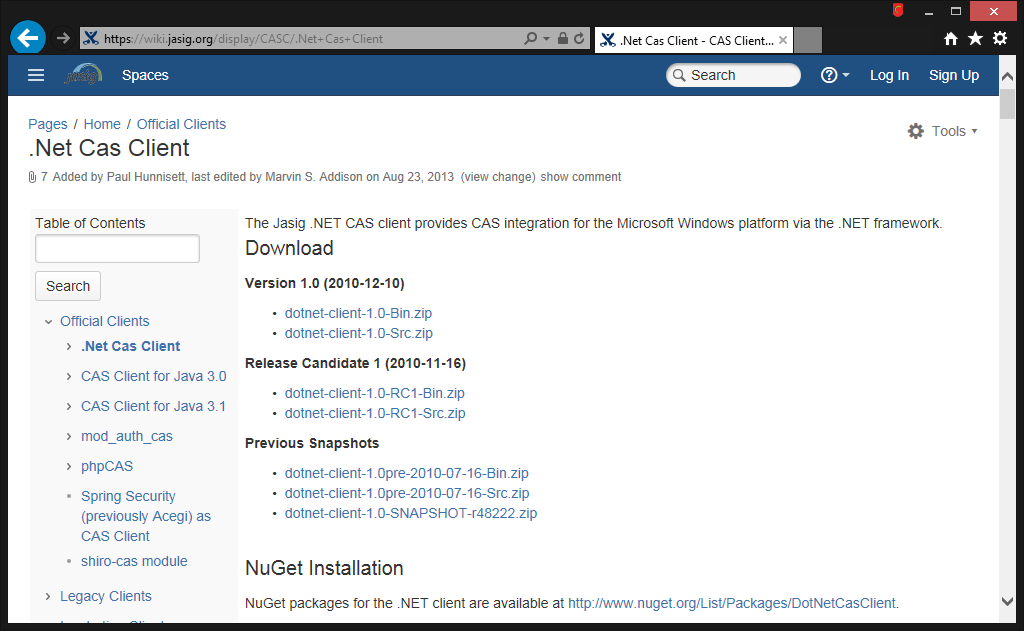
請參閱SUDA\_APF\_API\_使用文件.docx

# SSO介接說明

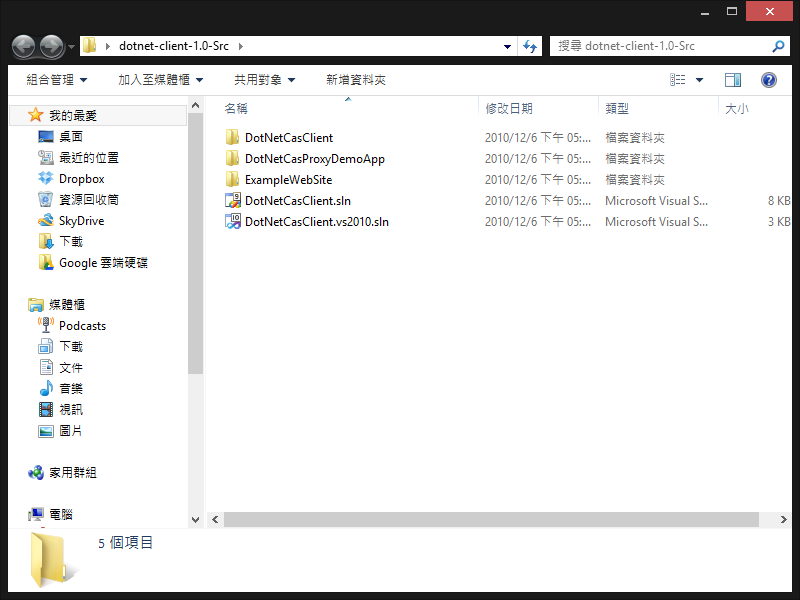
## CAS Client for ASP.NET元件建置

1. 下載.NET CAS Client。

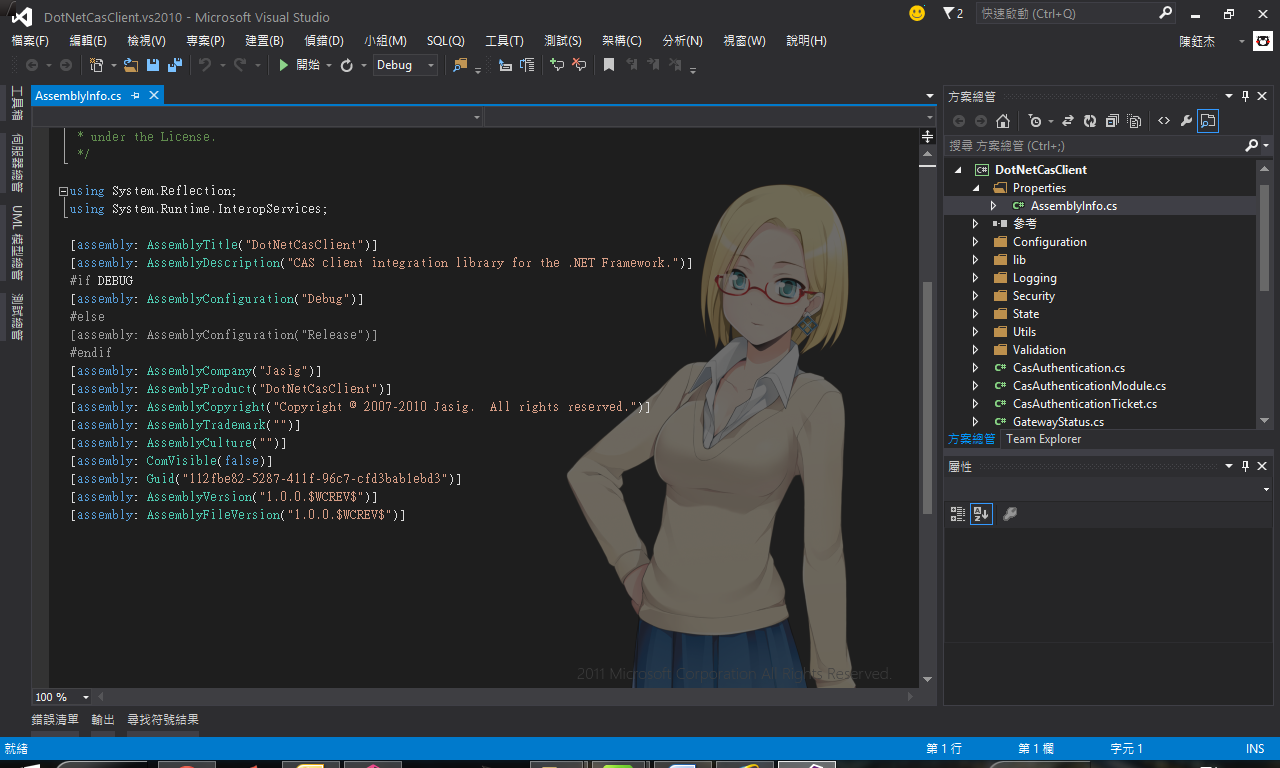
.NET CAS Client下載位置：<https://wiki.jasig.org/display/CASC/.Net+Cas+Client>



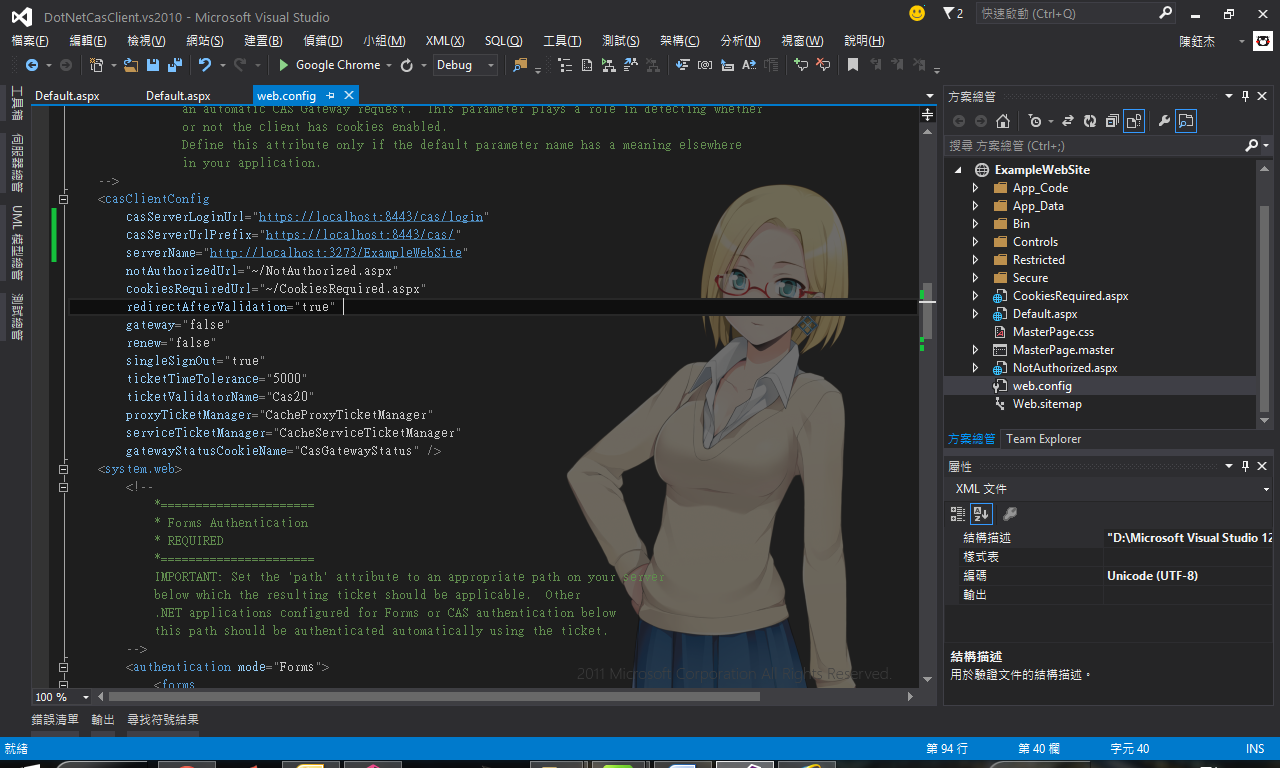
1. 下載「dotnet-client-1.0-Src.zip」並解壓縮：



1. 設定並建置CAS DotNetClient
   1. 開啟「DotNetCasClient.vs2010.sln」方案。
   2. 專案「DotNetCasProxyDemoApp」暫時用不到，所以從方案中移除。
   3. 將「DotNetCasClient」中「Properties」資料夾下的「AssemblyInfo.cs」刪除，將「AssemblyInfo.cs.tmpl」重新命名為「AssemblyInfo.cs」。
   4. 開啟「DotNetCasClient」專案，並在該專案中「Properties」資料夾下的「AssemblyInfo.cs」，將所有”$WCREV$”更改成”0”(或其它版號)。

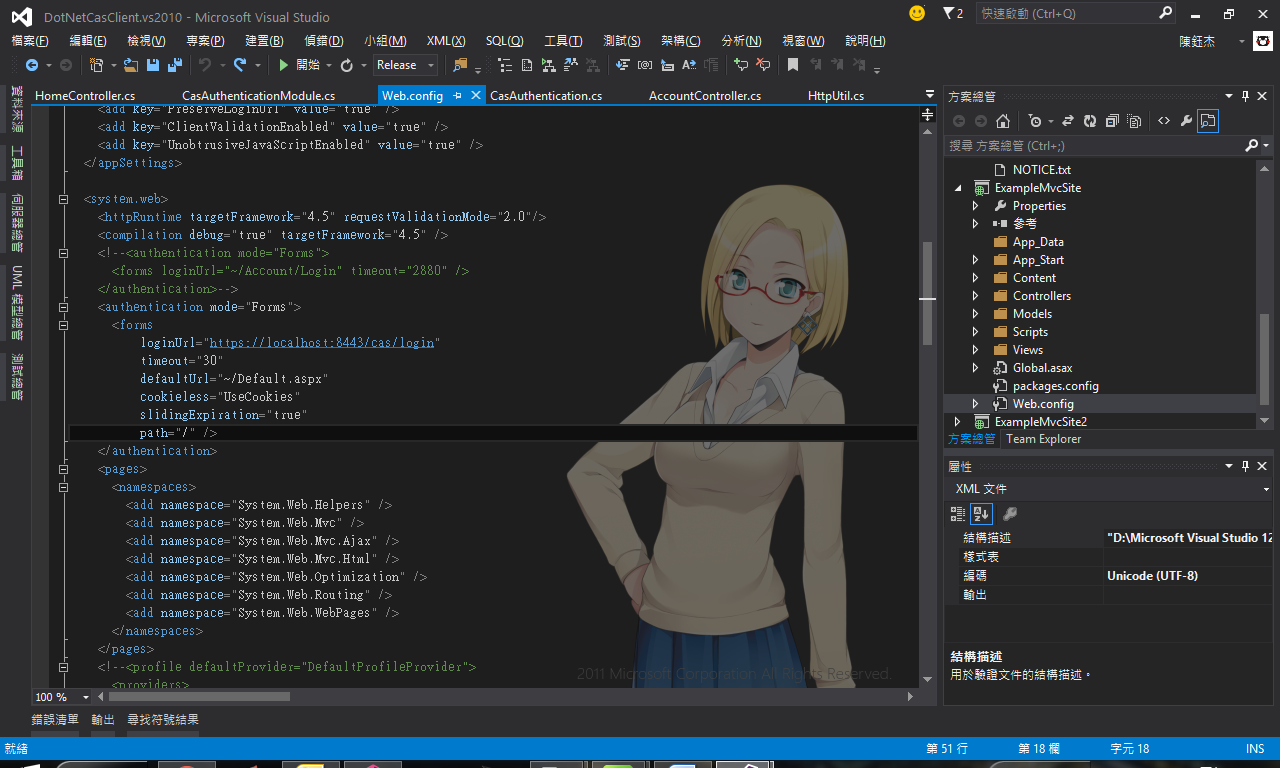


* 1. 將「ExampleWebSite」專案設置為啟始專案。
  2. 將「ExampleWebSite」專案根目錄下的「web.config.sample」重新命名為「web.config」
  3. 開啟「web.config」檔，找到「casClientConfig」，並將「casServerLoginUrl」屬性設定為「https://localhost:8443/cas/login」，將「casServerUrlPrefix」屬性設定為「https://localhost:8443/cas/」，將「serverName」屬性設置為「<http://localhost:3273/ExampleWebSite>」。

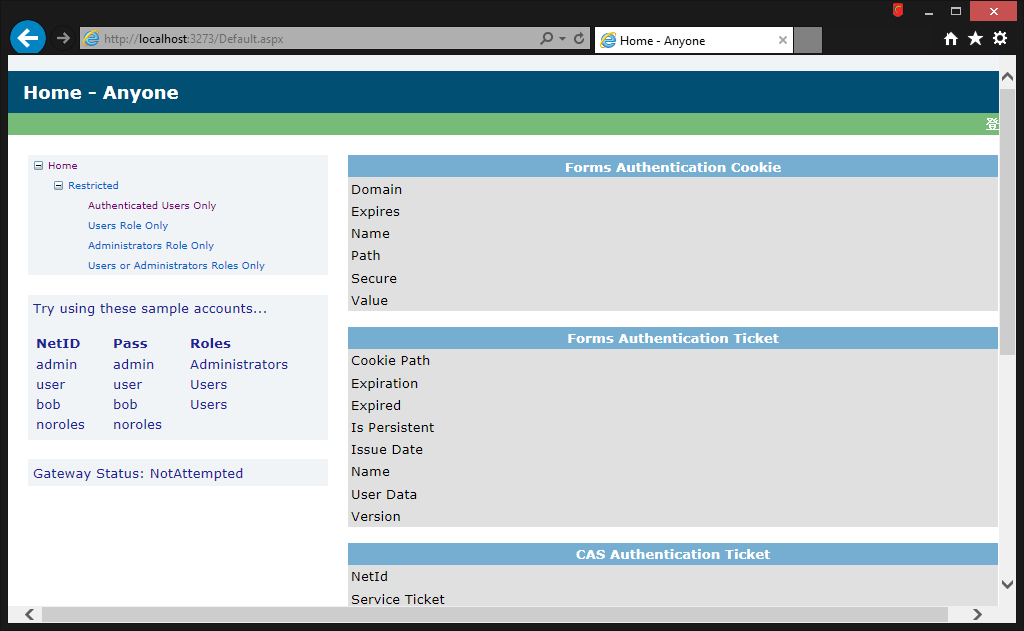


注意：「serverName」屬性中網路位址最後不要加「/」。

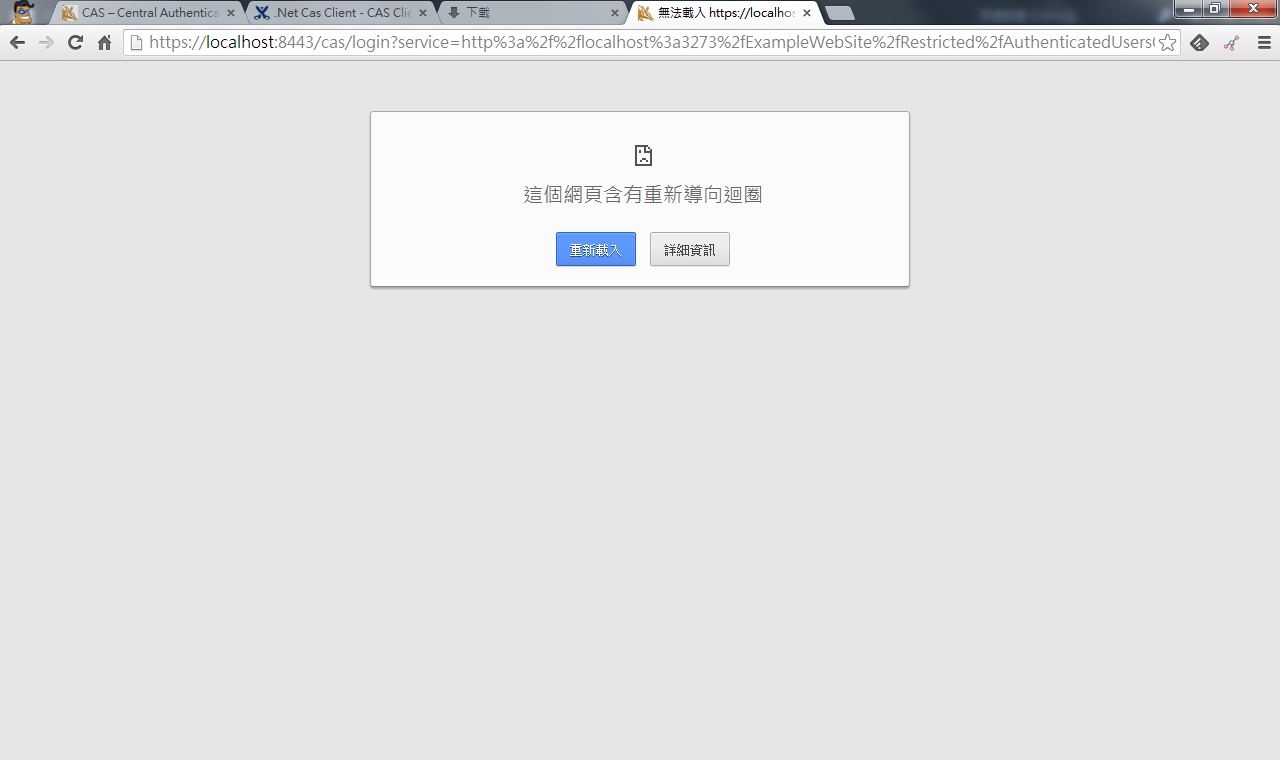
* 1. 找到「authentication」，將其中的「loginUrl」屬性設定為「https://localhost:8443/cas/login」，將「path」屬性設定為「/」。



1. 測試CAS DotNetClient
   1. 滑鼠右鍵點擊「DotNetCasClient」專案根目錄下的「Default.aspx」，選擇「在瀏覽器中檢視...」。



* 1. 點擊「Authenticated Users Only」連結，系統自動導向到CAS SSO登入頁面(如果IE有憑證警告訊息，直接點擊「繼續瀏覽此網站」)。
  2. 在CAS SSO登入頁面輸入完帳號密碼後，此時會出現登入異常，IE會顯示「無法顯示此網頁」，可是並無法知道錯誤細節。如果使用Chrome瀏覽器，就會發現系統遇到「重新導向迴圈」的問題了。



* 1. 解決CAS DotNetClient重新導向迴圈問題
* 在Visual Studio中打開「CASDotNetClient」專案中的「\Utils\HttpUtil.cs」文件，添加如下命名空間：

using System.Net.Security;

using System.Security.Authentication;

using System.Security.Cryptography.X509Certificates;

* 在HttpUtil類別中增加如下方法：

internal static bool CheckValidationResult(object sender, X509Certificate certificate, X509Chain chain, SslPolicyErrors errors)

{

return true;

}

* 在PerformHttpGet方法中新增驗證伺服器憑證後callback自動驗證程式，新增程式後的PerformHttpGet方法如下：

internal static string PerformHttpGet(string url, bool requireHttp200)

{

string responseBody = null;

//-- 以下為新增加的程式：驗證伺服器憑證後callback自動驗證程式

ServicePointManager.ServerCertificateValidationCallback = new System.Net.Security.RemoteCertificateValidationCallback(CheckValidationResult);

//-- 以上為新增加的程式

HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);

using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())

{

if (!requireHttp200 || response.StatusCode == HttpStatusCode.OK)

{

using (Stream responseStream = response.GetResponseStream())

{

if (responseStream != null)

{

using (StreamReader responseReader = new StreamReader(responseStream))

{

responseBody = responseReader.ReadToEnd();

}

}

}

}

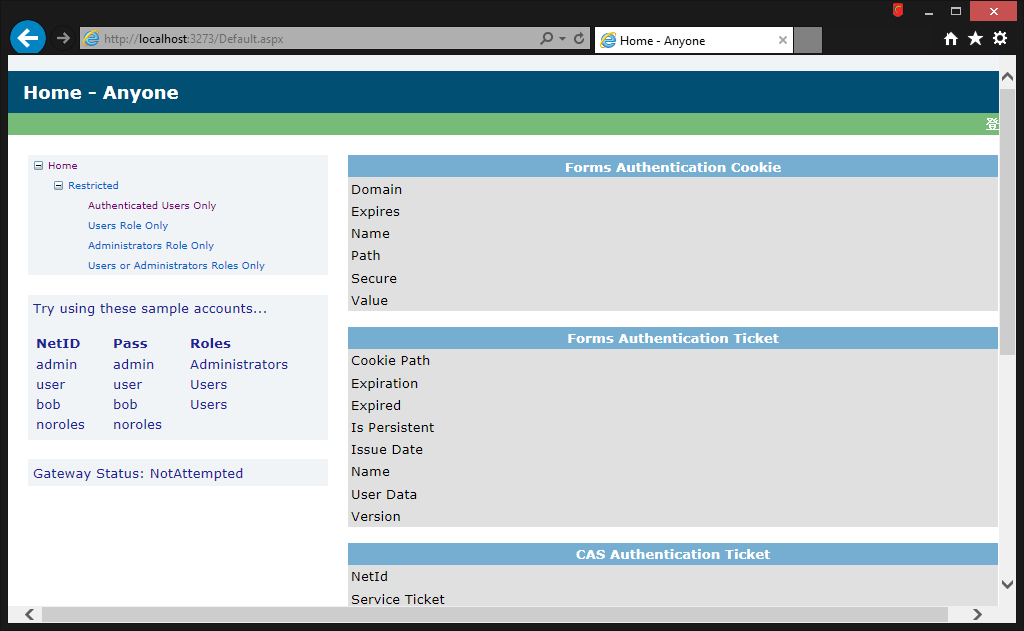
}

return responseBody;

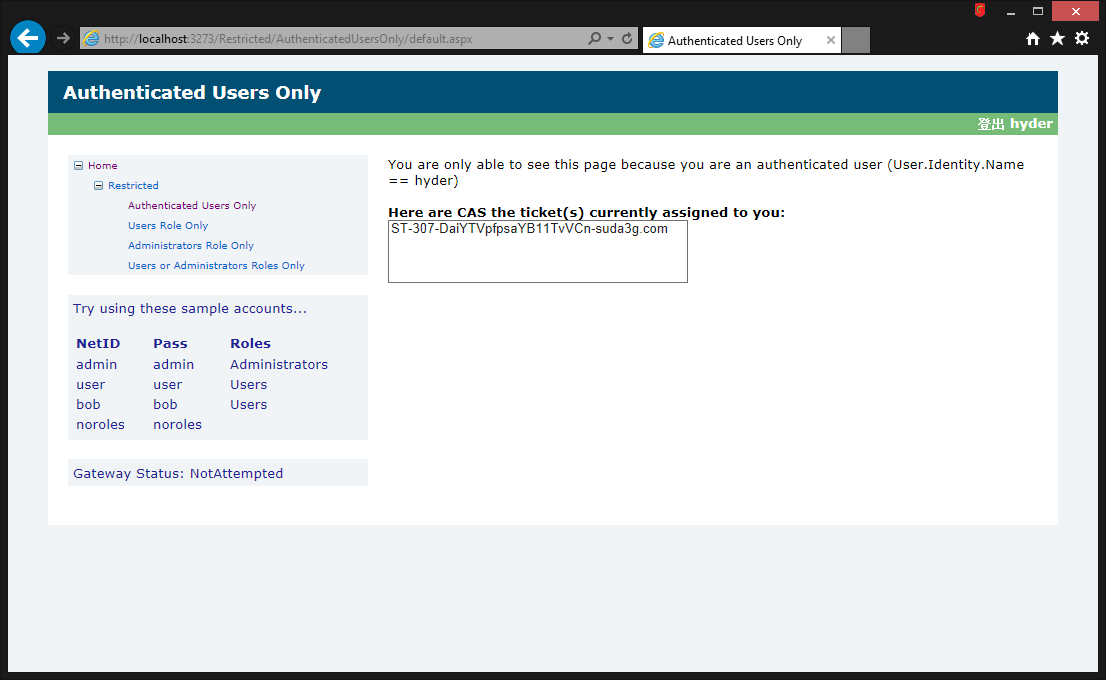
}

* 儲存修改並重新建置方案。

1. 測試 CAS DotNetClient
   1. 滑鼠右鍵點擊「DotNetCasClient」專案根目錄下的「Default.aspx」，選擇「在瀏覽器中檢視...」



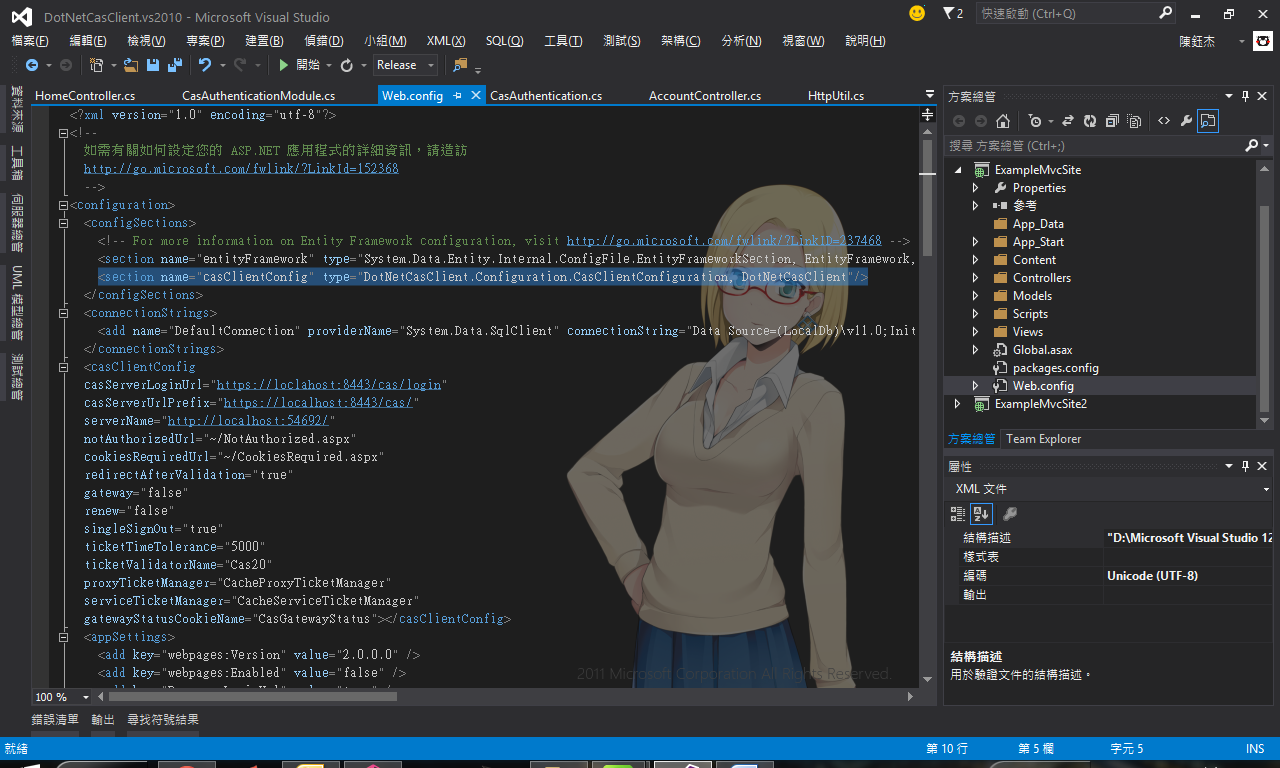
* 1. 點擊「Authenticated Users Only」連結，系統自動導向到CAS SSO登入頁面(如果IE有憑證警告訊息，直接點擊「繼續瀏覽此網站」)。
  2. 在CAS SSO登入頁面輸入完帳號密碼後，CAS自動完成登入並重新導向回：



## ASP.NET MVC與CAS Server設定

1. 開啟ASP.NET MVC專案，並設定web.config：
   1. 於configSections區段裡新增如下：

<section name="casClientConfig" type="DotNetCasClient.Configuration.CasClientConfiguration, DotNetCasClient"/>



* 1. 在connectionStrings區段後新增如下設定：

<casClientConfig

casServerLoginUrl="https://loclahost:8443/cas/login"

casServerUrlPrefix="https://localhost:8443/cas/"

serverName="http://localhost:54692/"

notAuthorizedUrl="~/NotAuthorized.aspx"

cookiesRequiredUrl="~/CookiesRequired.aspx"

redirectAfterValidation="true"

gateway="false"

renew="false"

singleSignOut="true"

ticketTimeTolerance="5000"

ticketValidatorName="Cas20"

proxyTicketManager="CacheProxyTicketManager"

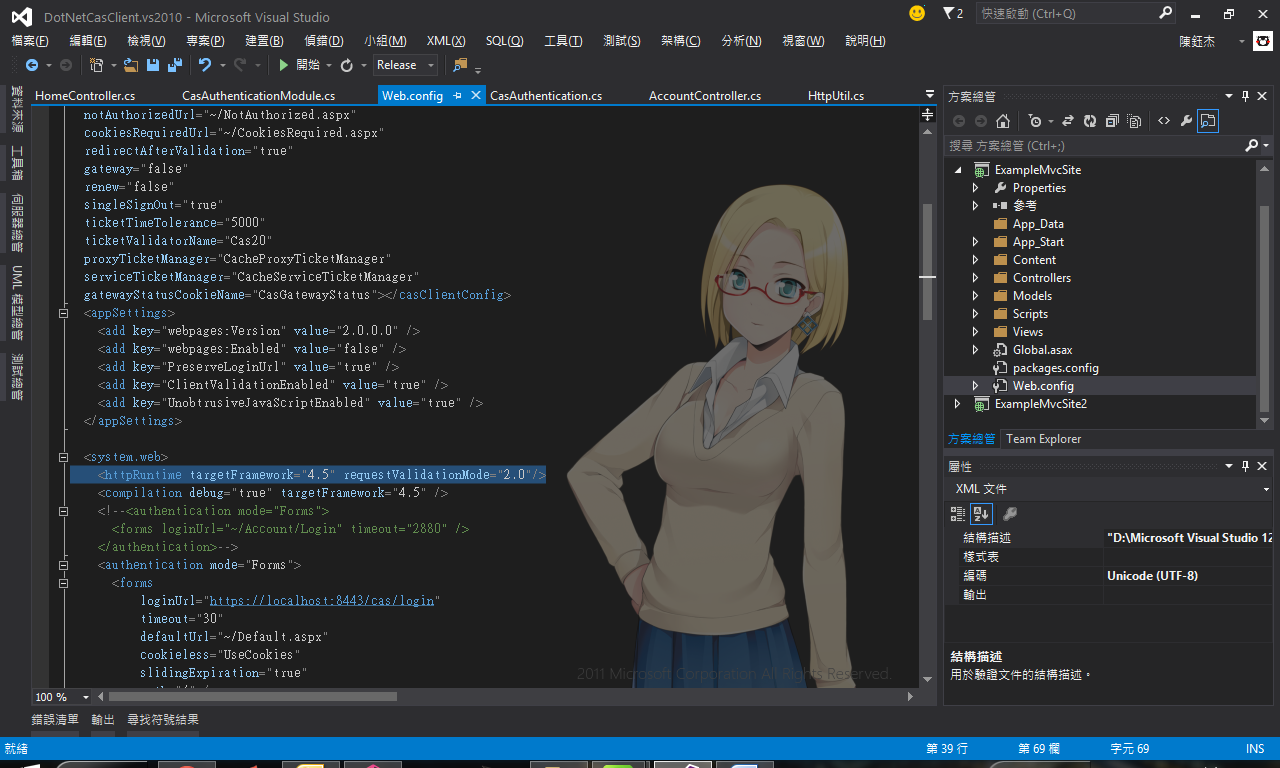
serviceTicketManager="CacheServiceTicketManager"

gatewayStatusCookieName="CasGatewayStatus"></casClientConfig>



* 1. 找到httpRuntime區段，並修改如下：

<httpRuntime targetFramework="4.5" requestValidationMode="2.0"/>



* 1. 找到authentication區段，並修改forms設定：

<authentication mode="Forms">

<forms

loginUrl="https://localhost:8443/cas/login"

timeout="30"

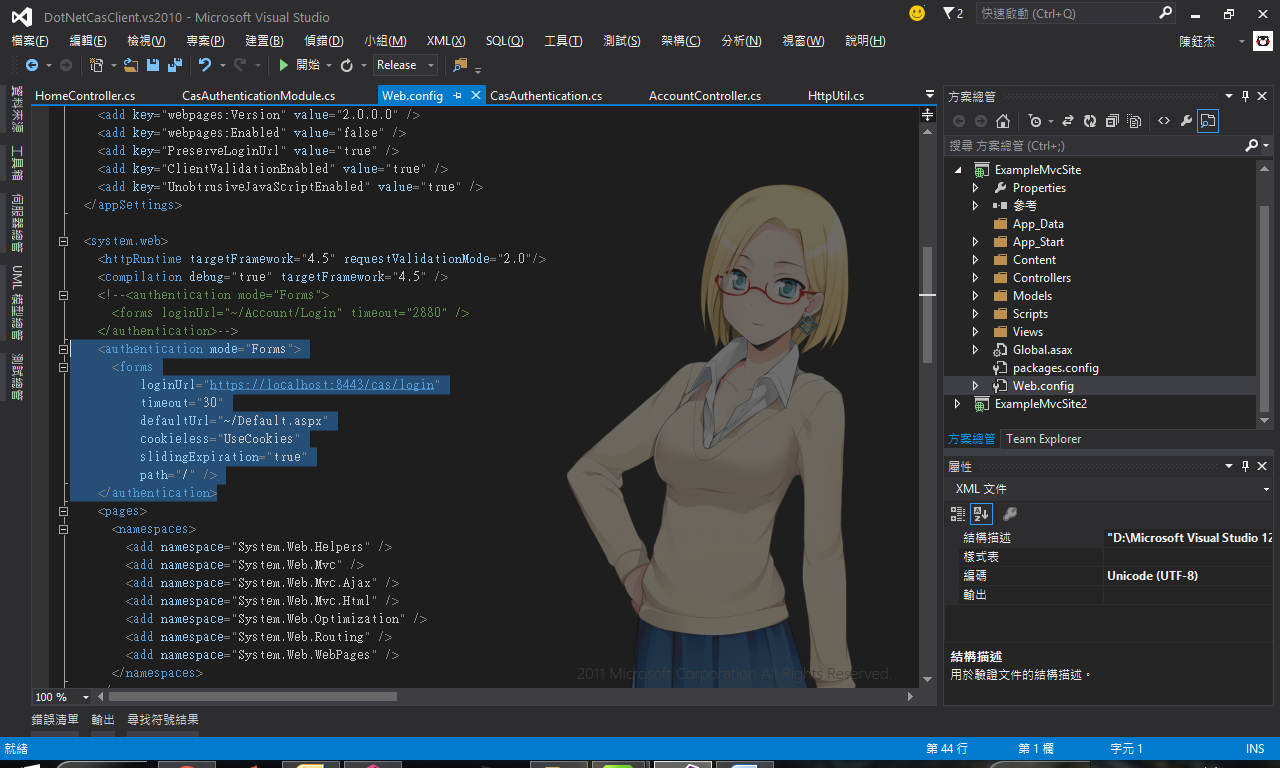
defaultUrl="~/Default.aspx"

cookieless="UseCookies"

slidingExpiration="true"

path="/" />

</authentication>



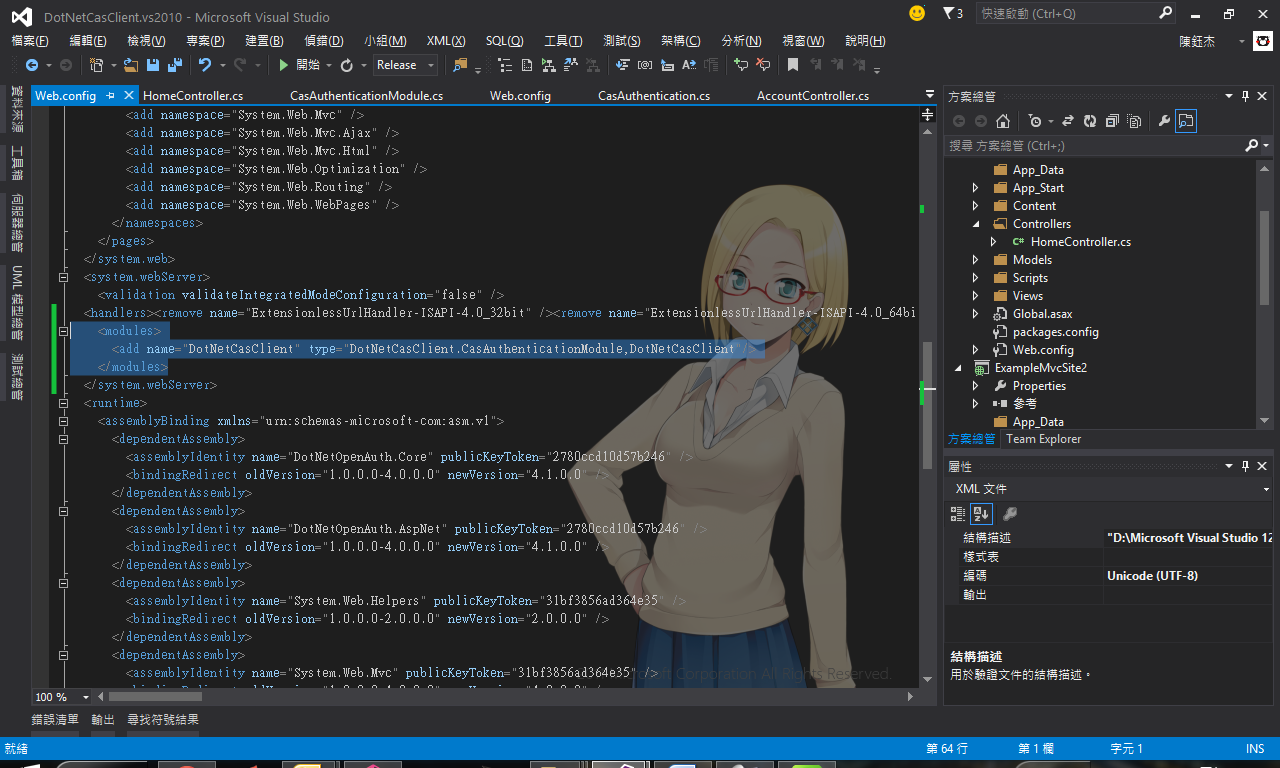
注意：發布到外部網站的話，Path設為"/"。

* 1. 接下來找到system.webServer區段，新增modules如下：

<modules>

<add name="DotNetCasClient" type="DotNetCasClient.CasAuthenticationModule,DotNetCasClient"/>

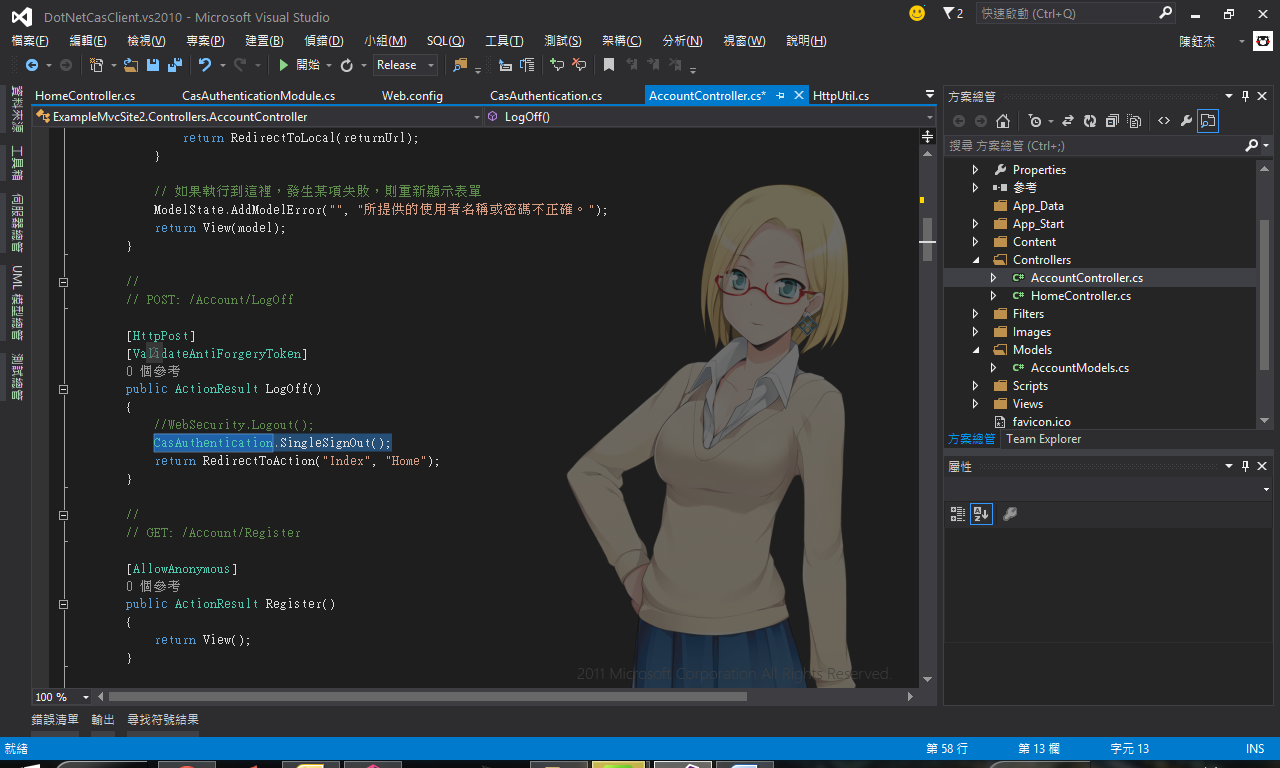
</modules>



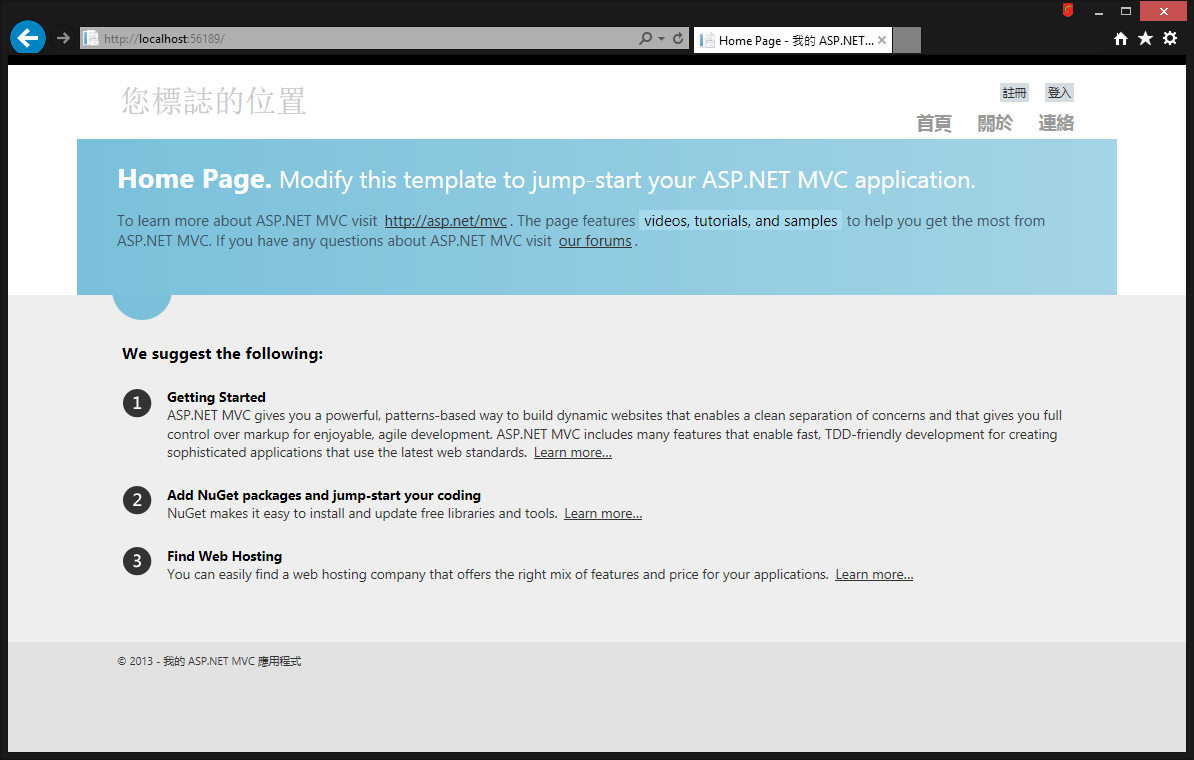
1. 修改程式碼，開啟俱有登入功能之Controller
   1. 在登入功能的Action上加入[authorize]屬性，如圖：



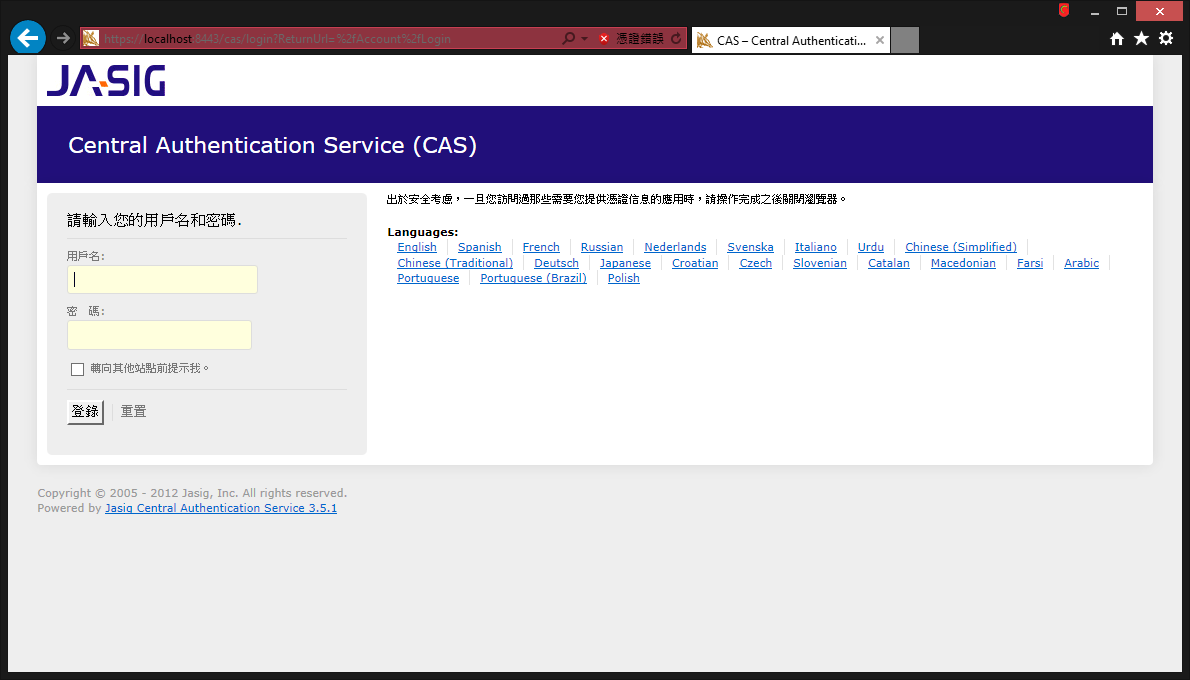
* 1. 在登出功能的Action中改用CasAuthentication.SingleSignOut()進行登出：



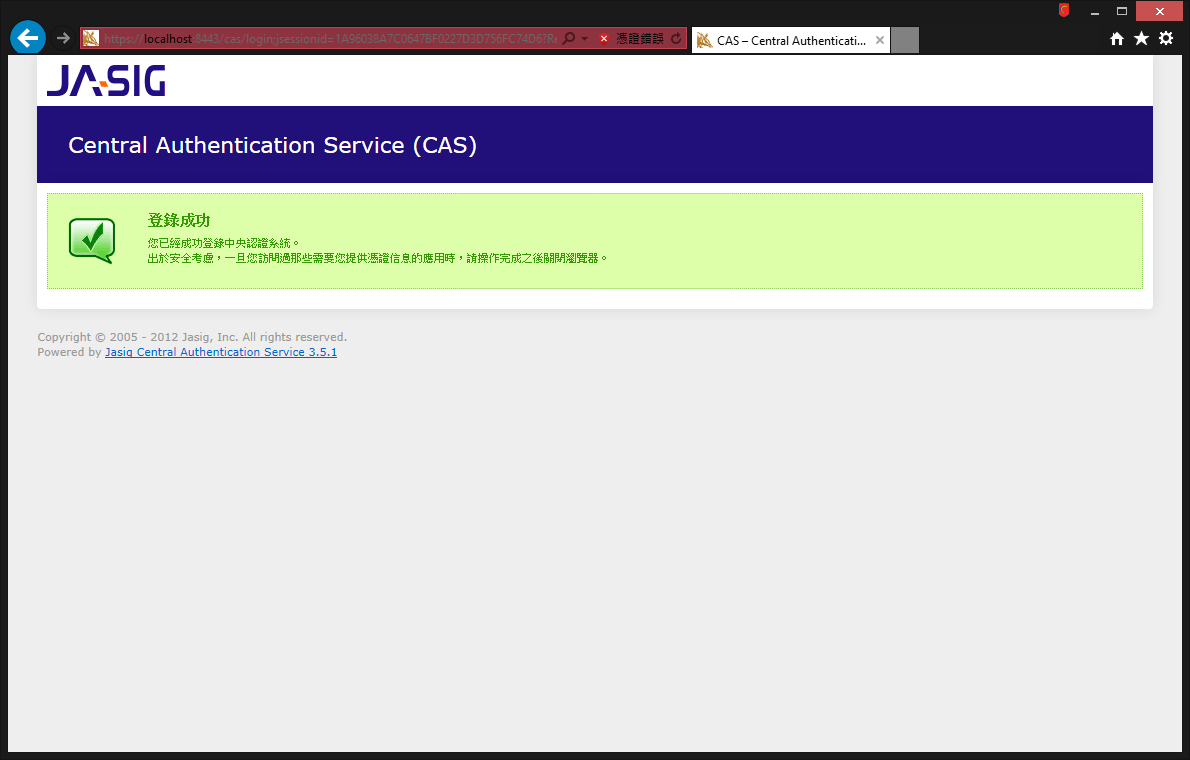
1. 開始偵錯ASP.NET MVC應用程式：
   1. 開啟ASP.NET MVC首頁



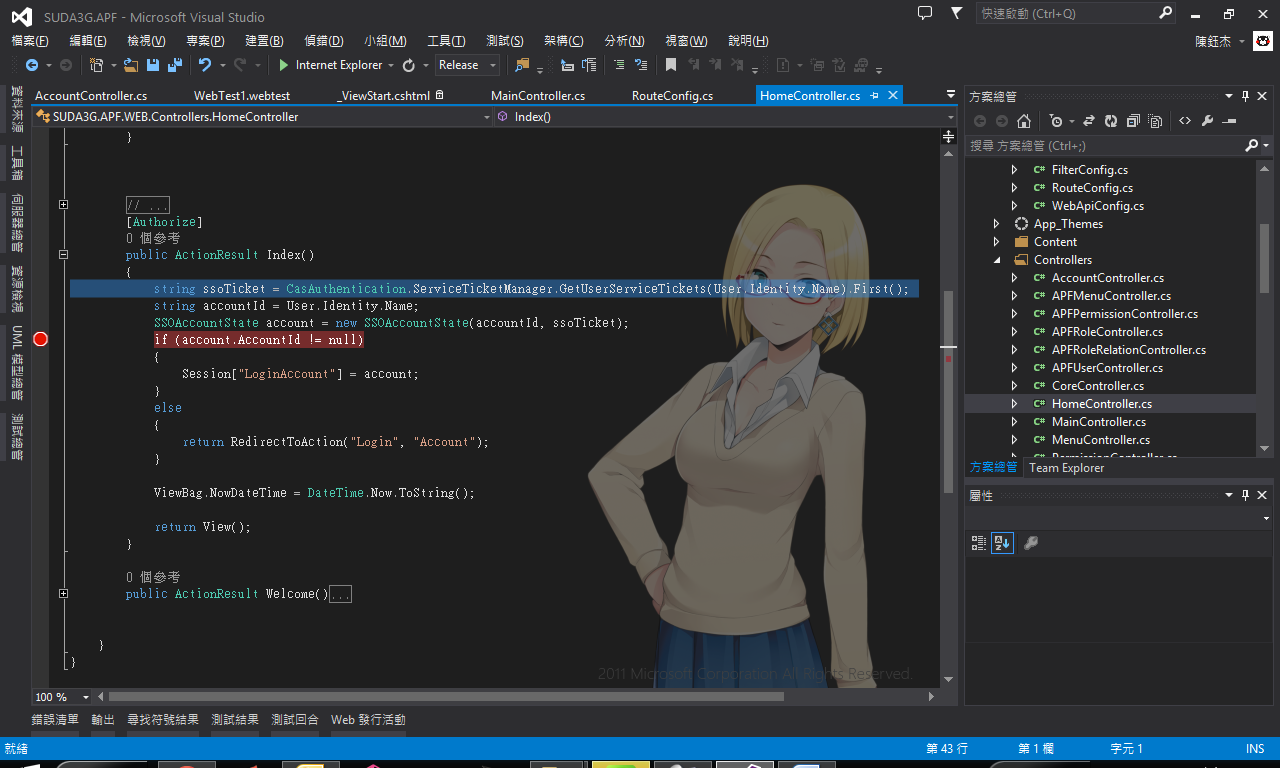
* 1. 點擊「登入」，頁面講導向CAS SSO頁面：



* 1. 打完帳號密碼登入成功：



## 子模組Server取得SSO認證資訊



string ssoTicket = CasAuthentication.ServiceTicketManager.GetUserServiceTickets(User.Identity.Name).First();

string accountId = User.Identity.Name;

SSOAccountState account = new SSOAccountState(accountId, ssoTicket);

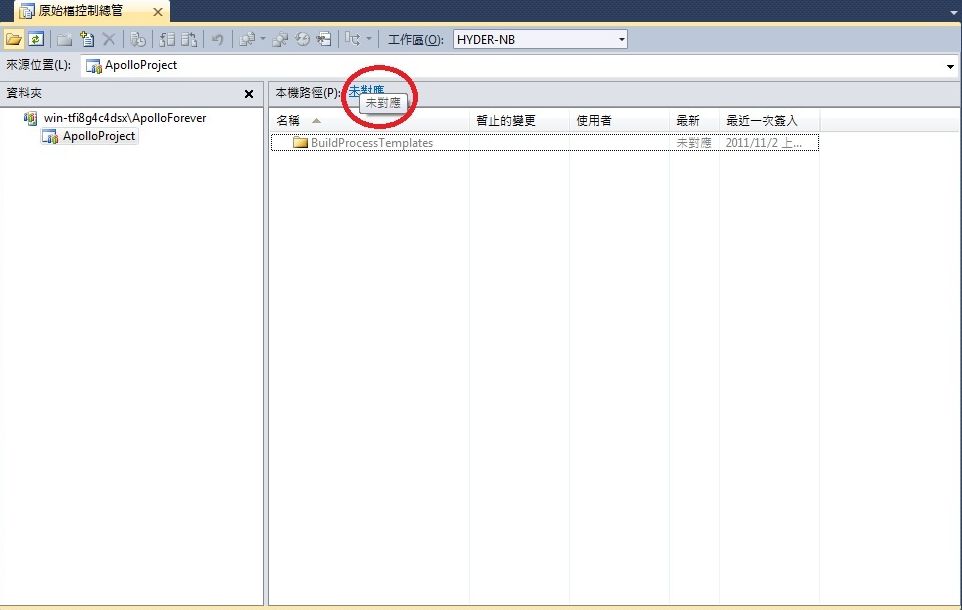
# 版本控管使用說明

版本控管主要用於管控程式碼異動，對於開發、測試階段以及後續維護階段，都是不可或缺的角色。Team Foundation Server 2010提供了十分成熟完整的程式碼管控功能，並與開發工具Visual Studio 2010深度整合，不論開發團隊大小，皆能提供完善的程式碼管控服務。程式碼管控主要步驟如下：

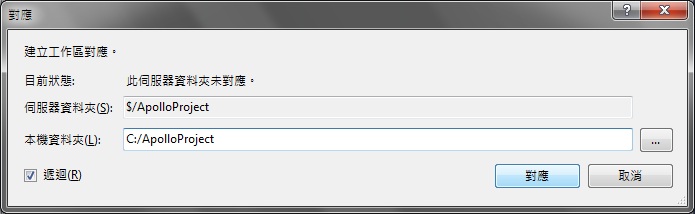


在先前的Team Foundation Server 2010的安裝及Team專案的建立完成後，首先，透過團隊的流程，會經過需求訪談及系統分析、系統設計，在系統設計階段尾聲及系統實作階段初始之間，系統設計人員架構出整個系統雛型時，便可將系統雛型簽入。同時請注意！完整性很重要，方案檔、專案檔、資源檔一個都不可漏。所謂凡走過必留下痕跡，目標是為了讓專案開發人員只要執行「取得最新程式碼」的動作，不須額外步驟即可編譯、執行程式碼，進行測試，同時也可以追蹤程式碼歷來的資訊及過程。以下透過系統雛型的簽入，進行步驟說明：

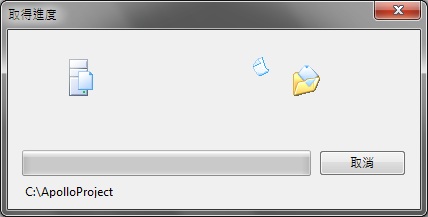
1. 對應開發環境的實體資料夾與伺服器的資料夾，開啟原始檔控制總管後，點擊未對應連結：



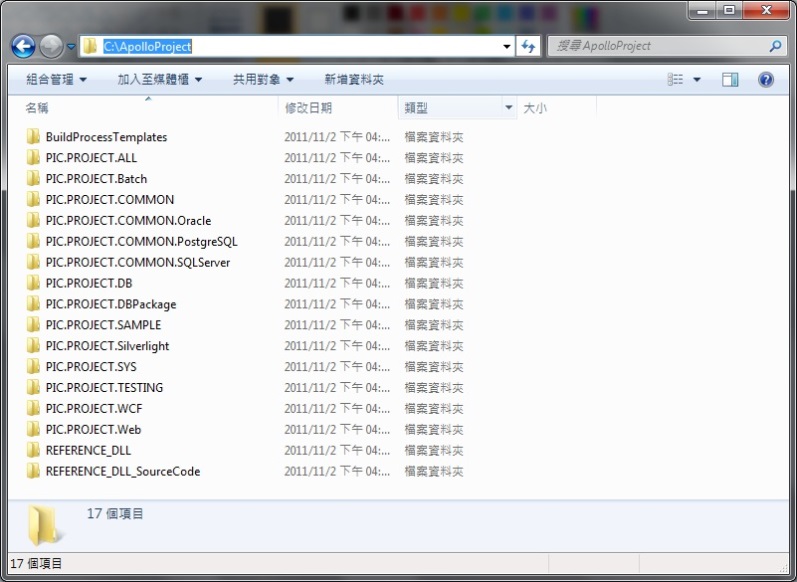
1. 建議實體資料夾路徑應與伺服器相同，以便於識別：



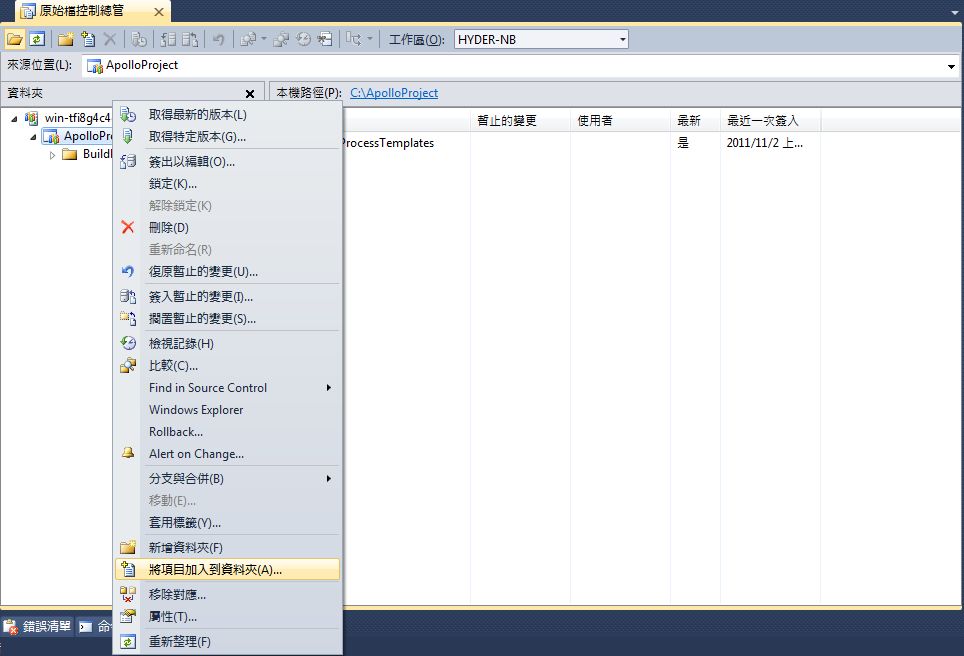
1. 開始進行對應：



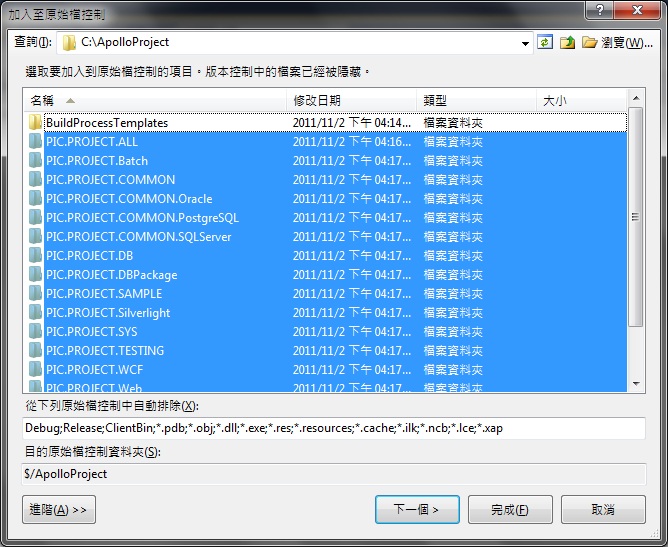
1. 將系統雛型之資料夾及程式碼結構複製至實體目錄：



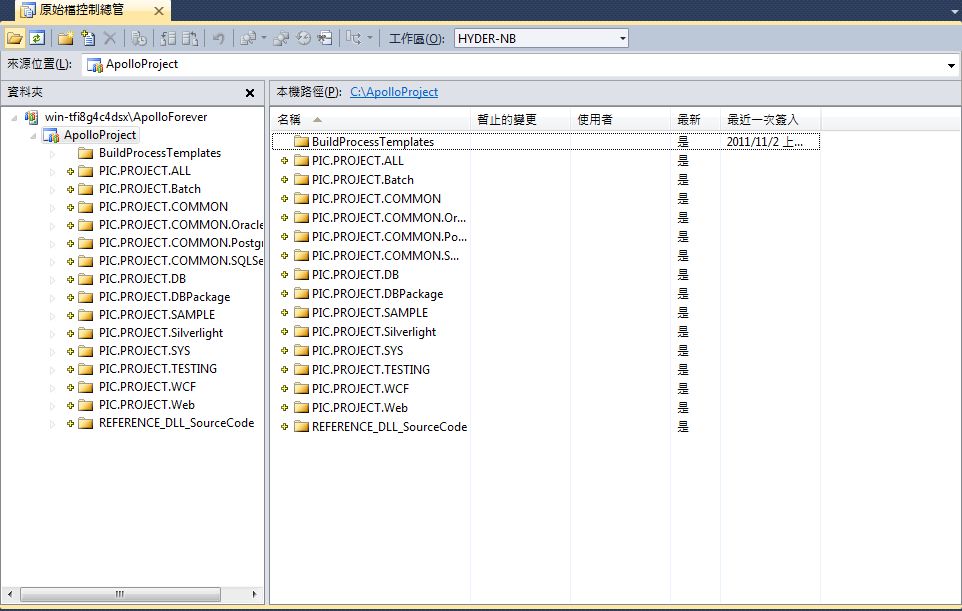
1. 於原始檔控制總管的該專案按右鍵並選取「將項目加入到資料夾」：



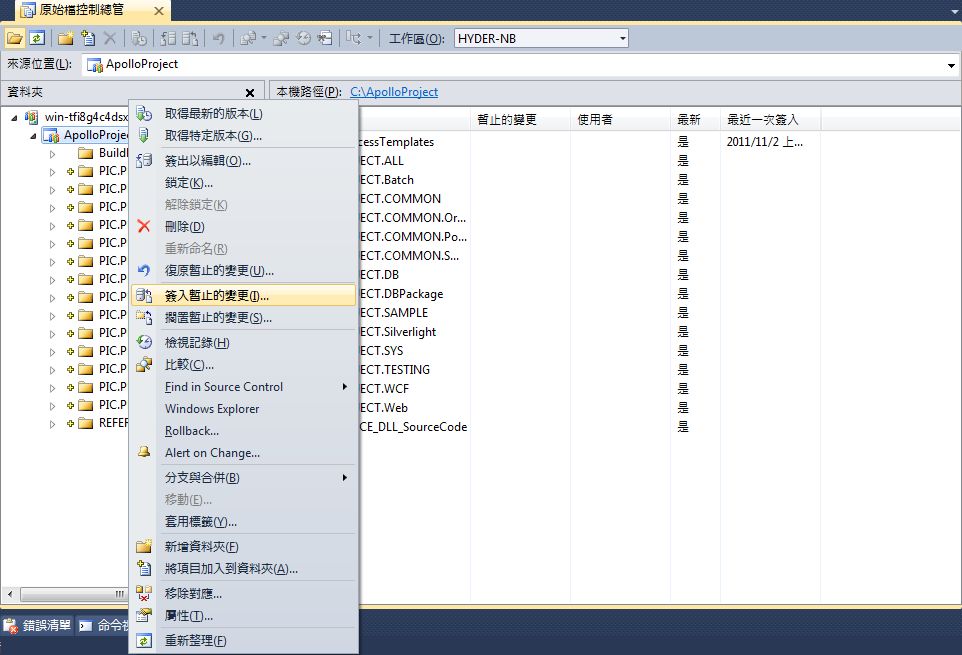
1. 選取方才搬移過來的系統雛型：



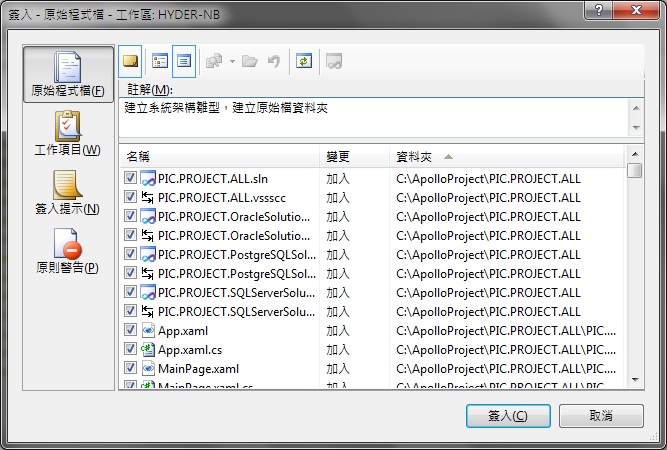
1. 完成系統雛型資料的新增：



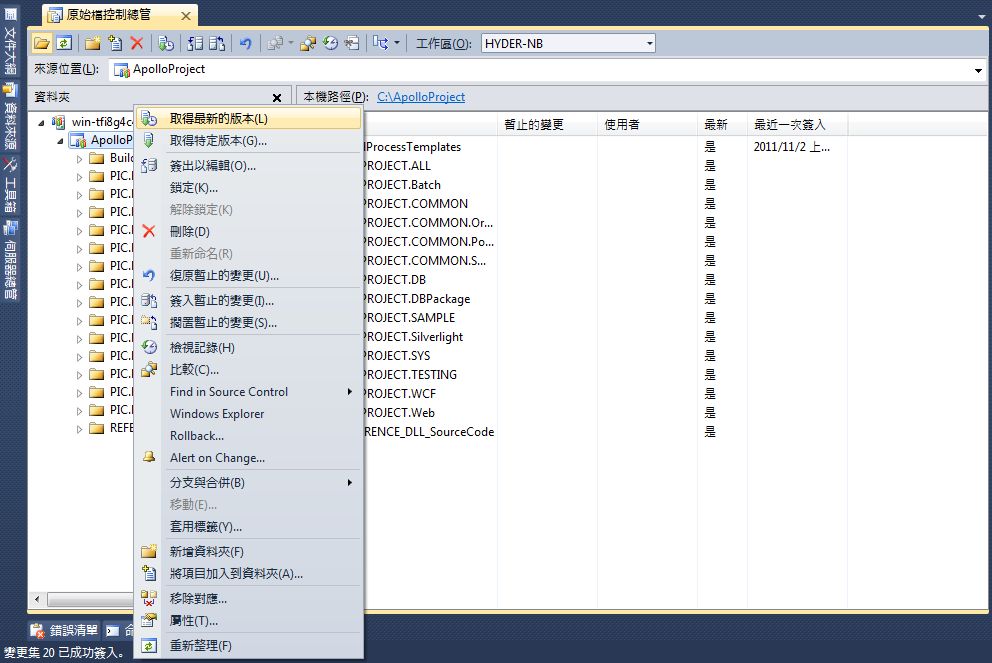
1. 此時於專案按右鍵，選取「簽入暫止的變更」：



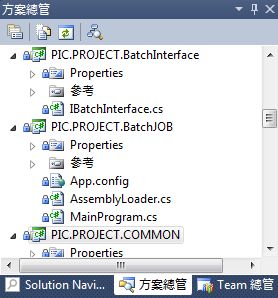
1. 確認將簽入的檔案及資料夾，同時輸入註解：



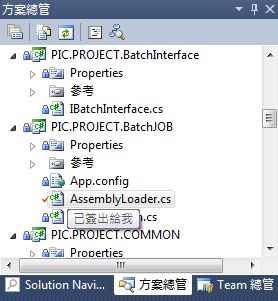
1. 完成簽入後，整個專案便進入版本控管當中。往後再每次開發及更動程式前，可於原始檔控制總管「取得最新的版本」：



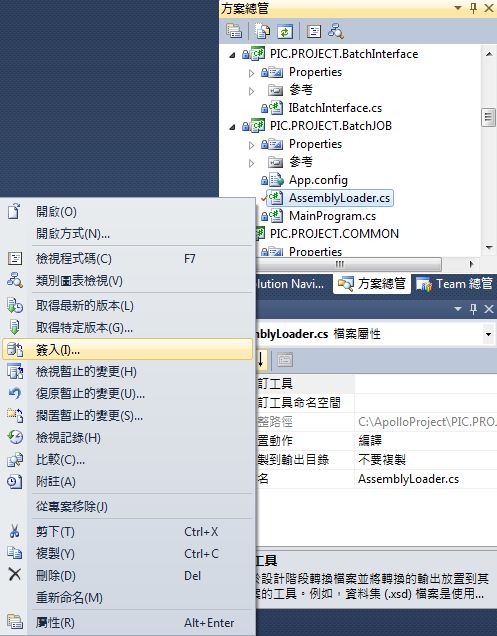
1. 開啟方案後，可以在方案總管看到所有已簽入之檔案皆為上鎖狀態：



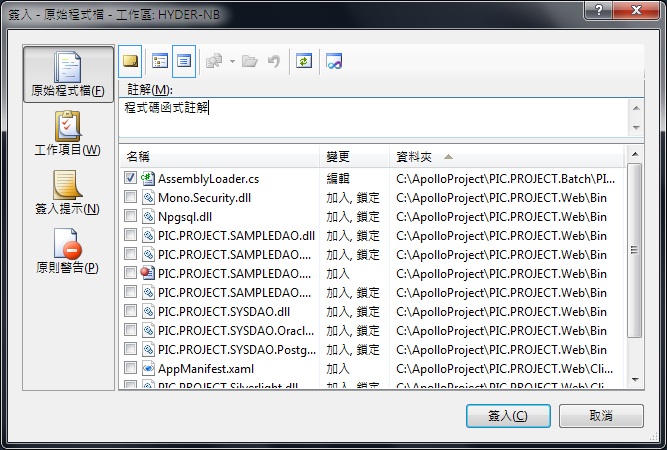
1. 開啟欲更改之程式，並進行程式撰寫，該狀態即會變為簽出狀態：



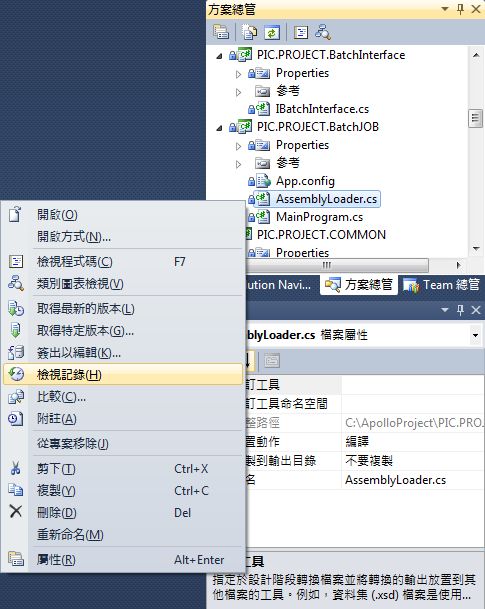
1. 修改完畢，假設測試跟驗證完成後，便可透過「簽入」選項，進行程式碼的簽入：



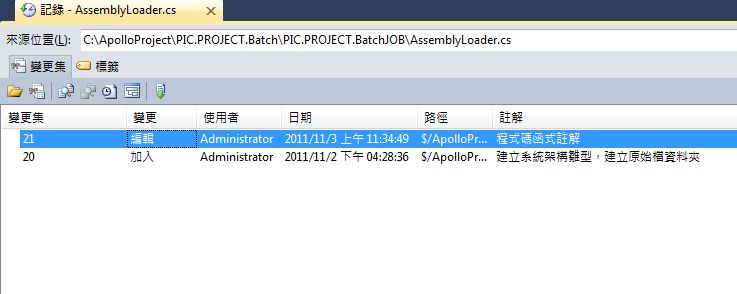
1. 透過簽入對話視窗檢視並加入該次簽入之相關註解：



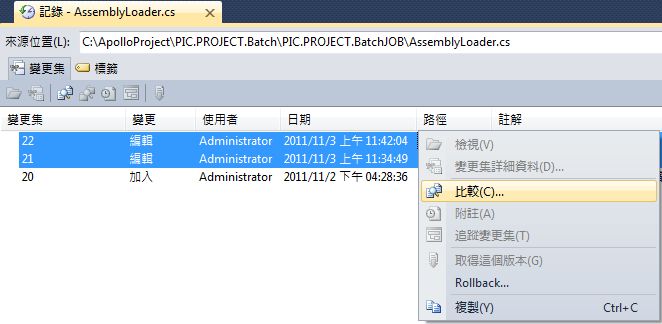
1. 按下簽入後，便將這次的修改、新增或刪除簽入版控，同時也增加一筆次號。可在相關程式檔或檔案按右鍵，選擇「檢視紀錄」，即可觀察程式碼歷來之紀錄：



1. 相關紀錄、註解及變更集次號：



1. 同時往後可透過檢視紀錄進行該檔案不同版次的比較：



1. 即可產生差異比較視窗，方便人員進行驗證：

