技術文件分享

John

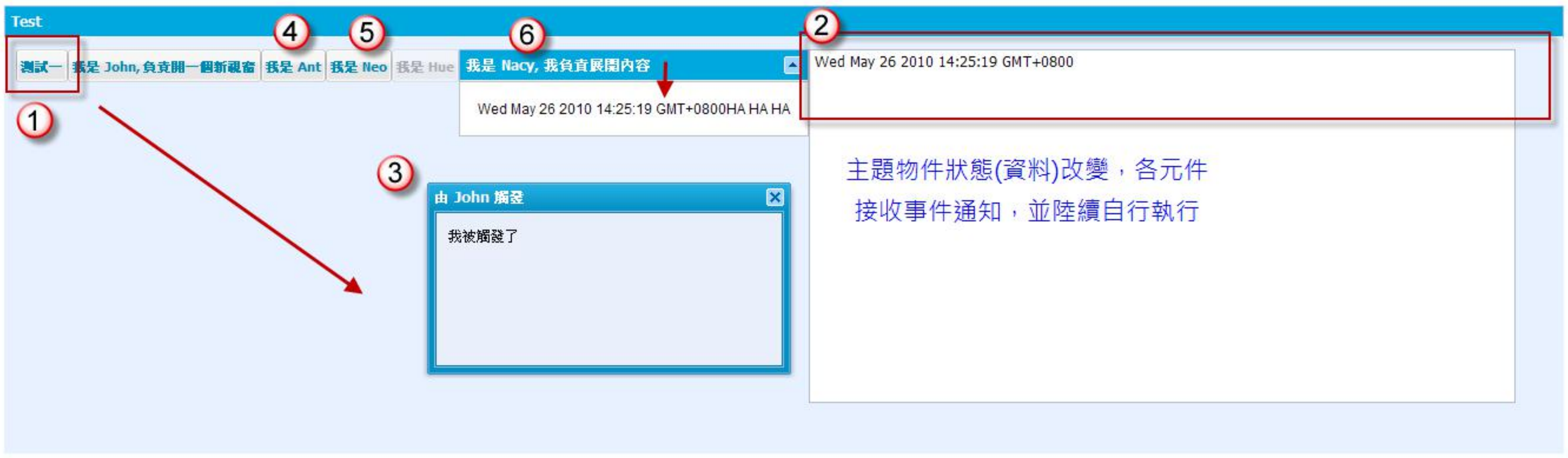
2010/05/27

前端UI - 廣播事件介紹 (Observer Pattern Using ExtJS)

情境：
假設有一個表單，包含了30個組件(文字框、下拉式選單、核選鈕...)，每一個組件狀態改變會影響主題，而主題又得改變其它組件狀態，或是再回傳改變原組件狀態，在傳統上手法上，可能會在每一個組件加上 onChange, onClick, onBlur .... 等等的Event ，然後撰寫一個(數個)方法，一個一個去改變狀態、改變資料. ..

假設今天需求變動，某些組件需移除，或是某些狀態的邏輯需要更動，這時程式碼就會開始亂掉，同時邏輯也極有可能發生衝突..

如果，每個組件能夠自行處理 本身狀態並回報給上層，再由上層決定如何將事件處理，要移除時也不會影響整體架構...



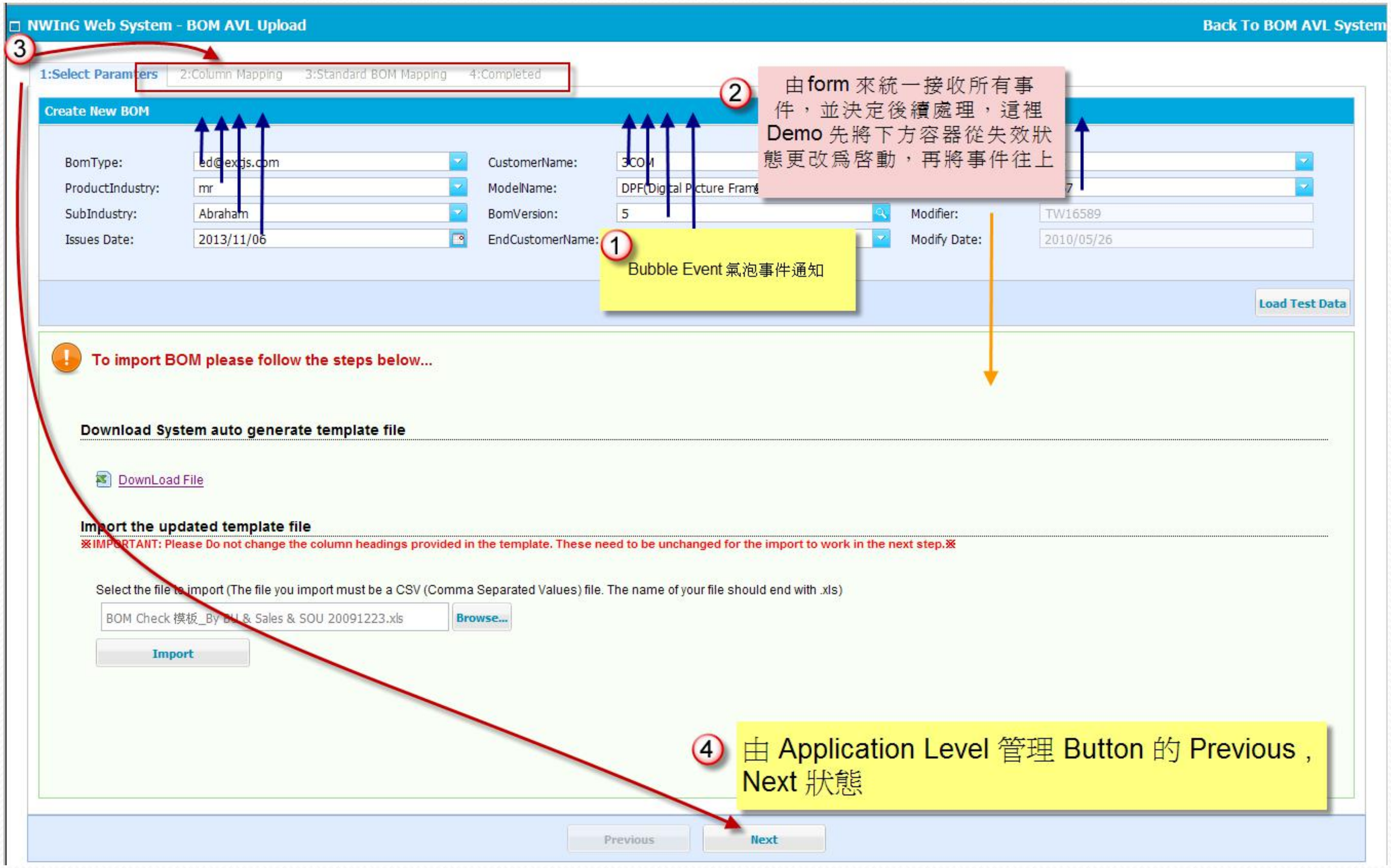
Wed May 26 2010 14:25:19 GMT+0800HA HA HA

Wed May 26 2010 14:25:19 GMT+0800

我被觸發了

主題物件狀態(資料)改變，各元件
接收事件通知，並陸續自行執行

Events Demo



NWInG Web System - BOM AVL Upload

Back To BOM AVL System

1:Select Paramters 2:Column Mapping 3:Standard BOM Mapping 4:Completed

Create New BOM

BomType:	ed@exgjs.com	CustomerName:	3COM		
ProductIndustry:	mr	ModelName:	DPF(Digital Picture Frame		7
SubIndustry:	Abraham	BomVersion:	5	Modifier:	TW16589
Issues Date:	2013/11/06	EndCustomerName:		Modify Date:	2010/05/26

Load Test Data

To import BOM please follow the steps below...

Download System auto generate template file

DownLoad File

Import the updated template file

※IMPORTANT: Please Do not change the column headings provided in the template. These need to be unchanged for the import to work in the next step.※

Select the file to import (The file you import must be a CSV (Comma Separated Values) file. The name of your file should end with .xls)

BOM Check 模板_By BU & Sales & SOU 20091223.xls Browse...

Import

Previous Next

1 Bubble Event 氣泡事件通知
2 由 form 來統一接收所有事件，並決定後續處理，這裡 Demo 先將下方容器從失效狀態更改爲啓動，再將事件往上
3
4 由 Application Level 管理 Button 的 Previous，Next 狀態

SubSonnic ORM

- 利用 T4 產生 DataBase ORM
- 好用的 ActiveRecord
- 已實作 Repository Pattern
- 可使用標準LINQ 表達式語法
- 可使用標準LINQ Lambda 語法
- 可使用 Transaction機制
- 可再將物件轉回 DataTable, DataSet, 靈活性高
- 亦可使用其提供的 Fluent 語法 (利如 And,Or,Like...)

SubSonic - 輸出成 SQL COmmand

程式碼

```
var query = db.Select.From()
.Paged(2, 30)
.BuildSqlStatement();
```

輸出

```
SELECT * FROM (SELECT ROW_NUMBER() OVER ( ORDER BY
[dbo].[BOM_UserList].[id]) AS Row, [dbo].[BOM_UserList].[id],
[dbo].[BOM_UserList].[userID], [dbo].[BOM_UserList].[userName],
[dbo].[BOM_UserList].[userForeignName],
[dbo].[BOM_UserList].[userPassword],
[dbo].[BOM_UserList].[departmentID],
[dbo].[BOM_UserList].[userExtension], [dbo].[BOM_UserList].[notesID],
[dbo].[BOM_UserList].[userEMail], [dbo].[BOM_UserList].[userTitleID],
[dbo].[BOM_UserList].[userResponsibilityID],
[dbo].[BOM_UserList].[factoryID], [dbo].[BOM_UserList].[BU],
[dbo].[BOM_UserList].[BG], [dbo].[BOM_UserList].[userIndustry],
[dbo].[BOM_UserList].[defaultLanguage],
[dbo].[BOM_UserList].[userIDCreatorID],
[dbo].[BOM_UserList].[userIDCreatedDate],
[dbo].[BOM_UserList].[userIDModifierID],
[dbo].[BOM_UserList].[userIDModifyDate],
[dbo].[BOM_UserList].[userIDActive], [dbo].[BOM_UserList].[agentID],
[dbo].[BOM_UserList].[sales], [dbo].[BOM_UserList].[PM],
[dbo].[BOM_UserList].[salesmanager], [dbo].[BOM_UserList].[BUHEad],
[dbo].[BOM_UserList].[CEVP], [dbo].[BOM_UserList].[IT],
[dbo].[BOM_UserList].[role], [dbo].[BOM_UserList].[orderItem],
[dbo].[BOM_UserList].[salesBoss] FROM [dbo].[BOM_UserList] ) AS
PagedResults WHERE Row >= 31 AND Row <= 60
```

SubSonic **ExecuteTypedList**

程式碼

```
var query = Select
.From<BOM_UserList>()
.Paged(2, 30)
.ExecuteTypedList<BOM_UserList>();
```

輸出

```
foreach (var bomUserList in query)
{
        Response.Write(bomUserList.userID + "--" +
         bomUserList.salesmanager + "-" +
         bomUserList.userForeignName + "<BR>");
}
```

```
F0817287---Xiang-Pin Peng
F0817290---Xin-Xin Liu
F0817291---Xue Qiong
F0817293---Fei-Hua Wang
F0817294---Qiu-Mei Guo
F0817295---Ming-Juan Xiao
F0817296---Huang Xiang
F0817297---Zong-Zong Chen
F0817298---Zhang Pan
```

SubSonic ExecuteReader – 轉回DataTable

程式碼

```
var query =
db.Select.From<BOM_BOMINFO>()
             .Paged(1, 30).ExecuteReader();
```

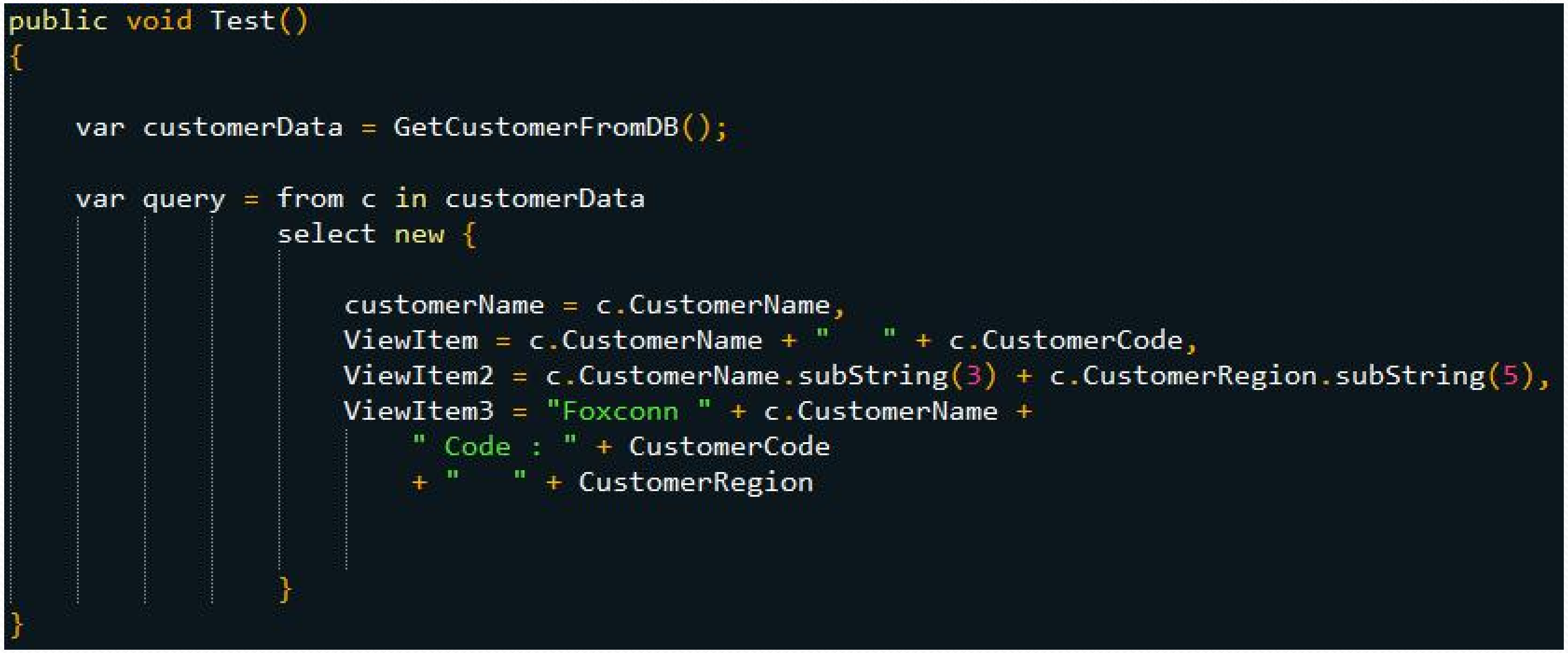
輸出

```
int fc = query.FieldCount;
while (query.Read())
{
        for (int i = 0; i < fc;i++ )
              test += query.GetValue(i) + "<BR>";
}
Response.Write(test);

// 轉成 DataTable

DataTable dt = new DataTable("dtTest");
dt.Load(query);

//Bind To GridView
GridView2.DataSource = dt;
GridView2.DataBind();
```

ViewModel Pattern 介紹

在實際情況中，我們常常會面臨到資料庫模型及領域(or 檢視模型)的轉換來符合輸出的需求，例如以下狀況:

LINQ 提供了匿名型別可以方便的轉換

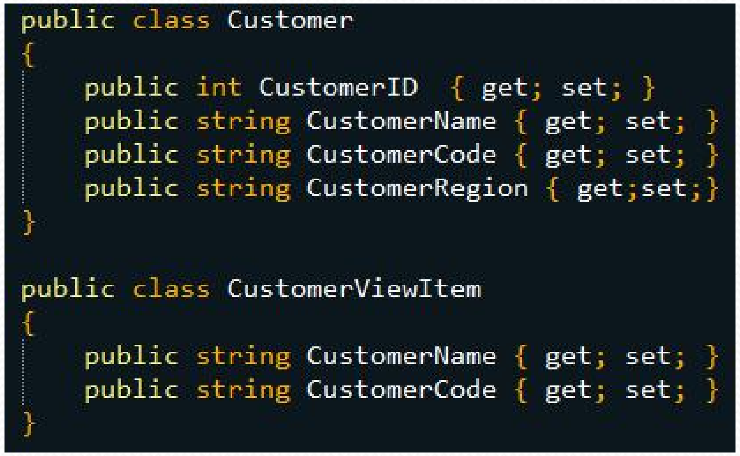


```
public void Test()
{

    var customerData = GetCustomerFromDB();

    var query = from c in customerData
                select new {

                    customerName = c.CustomerName,
                    ViewItem = c.CustomerName + "   " + c.CustomerCode,
                    ViewItem2 = c.CustomerName.subString(3) + c.CustomerRegion.subString(5),
                    ViewItem3 = "Foxconn " + c.CustomerName +
                        " Code : " + CustomerCode
                        + "   " + CustomerRegion


                }
}
```

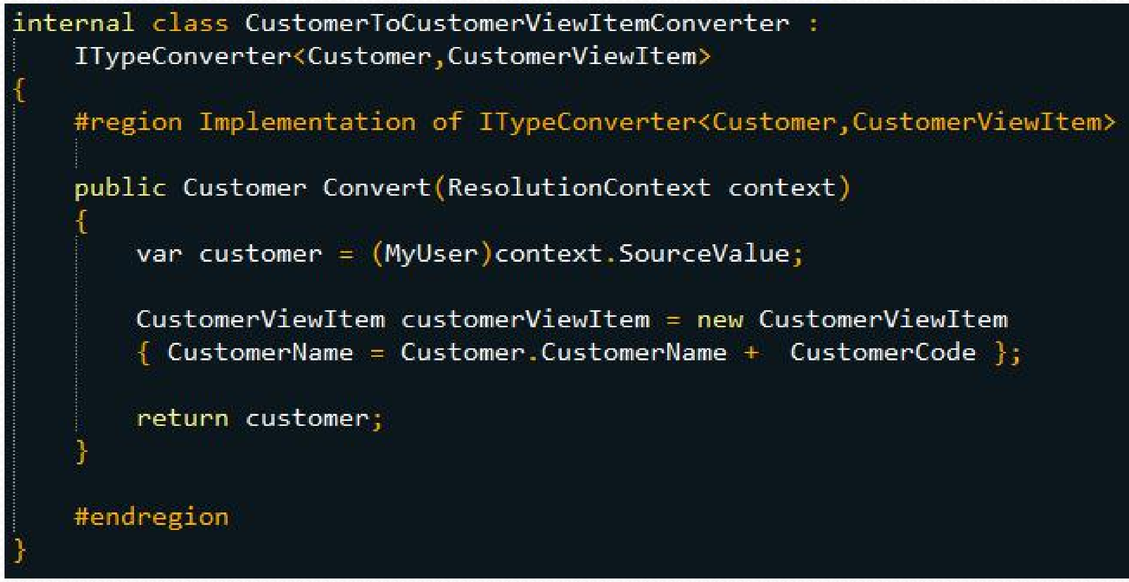
但是在某些狀況下，如果這個需求一多，就會導致Copy & Paste 的動作增加，副作用就是當面臨需求變動時，要更改的地方也多。

AutoMapper

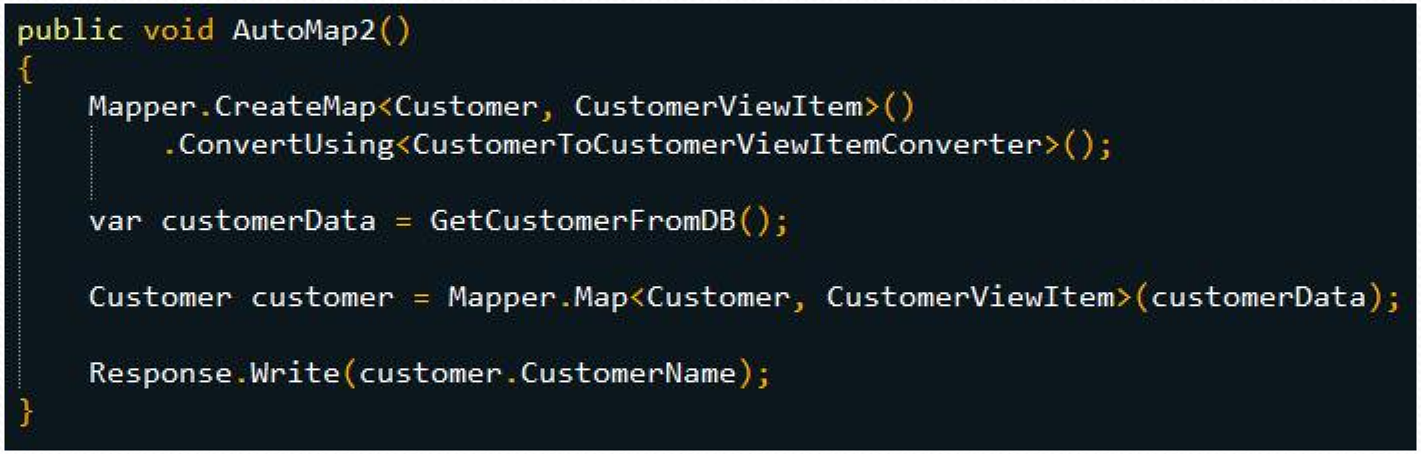
- 利用 AutoMapper 可以幫助我們在資料庫模型及領域模式自動轉換資料



```
public class Customer
{
    public int CustomerID  { get; set; }
    public string CustomerName { get; set; }
    public string CustomerCode { get; set; }
    public string CustomerRegion { get;set;}
}

public class CustomerViewItem
{
    public string CustomerName { get; set; }
    public string CustomerCode { get; set; }
}
```

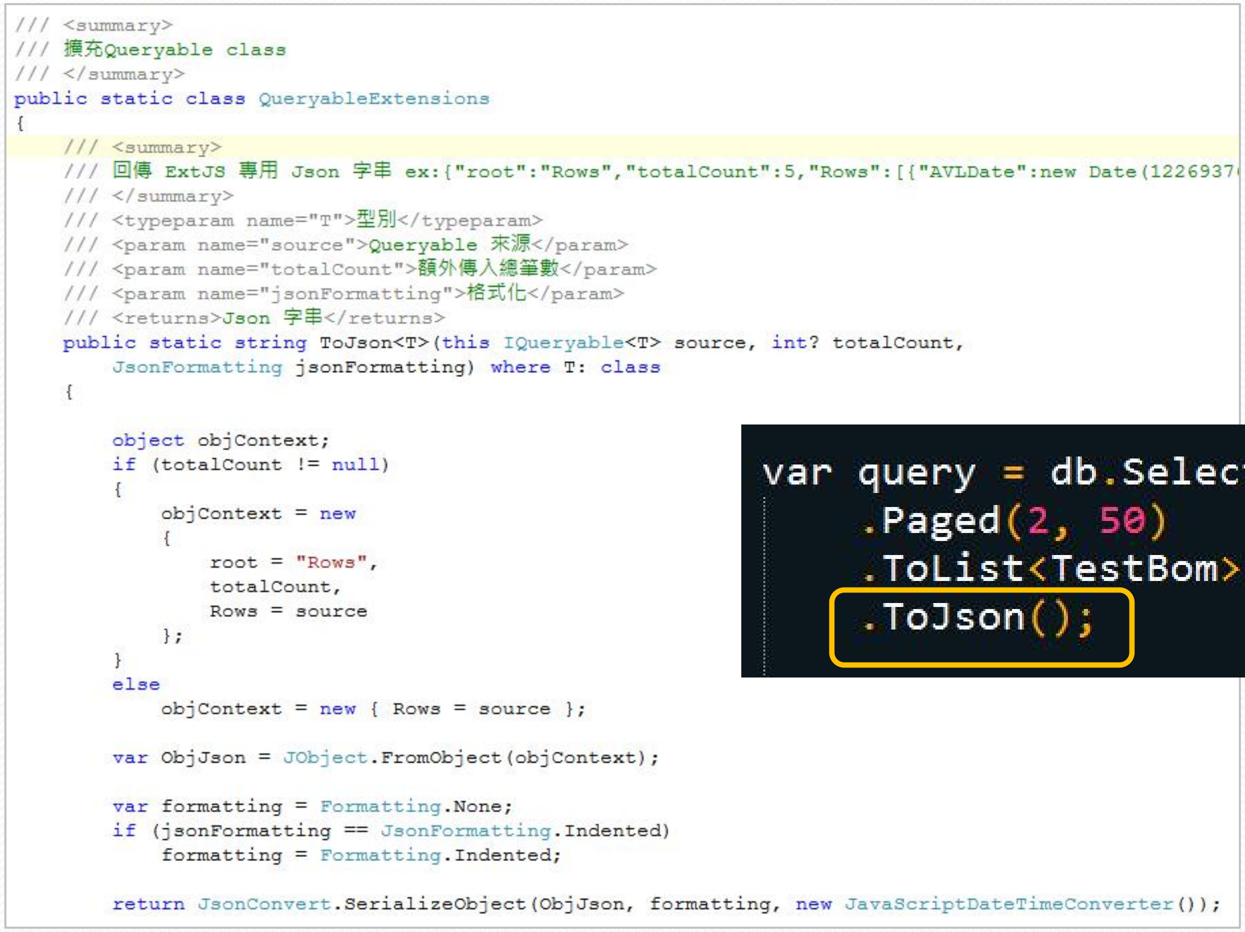


```
internal class CustomerToCustomerViewItemConverter :
    ITypeConverter<Customer,CustomerViewItem>
{
    #region Implementation of ITypeConverter<Customer,CustomerViewItem>

    public Customer Convert(ResolutionContext context)
    {
        var customer = (MyUser)context.SourceValue;

        CustomerViewItem customerViewItem = new CustomerViewItem
        { CustomerName = Customer.CustomerName +  CustomerCode };

        return customer;
    }

    #endregion
}
```

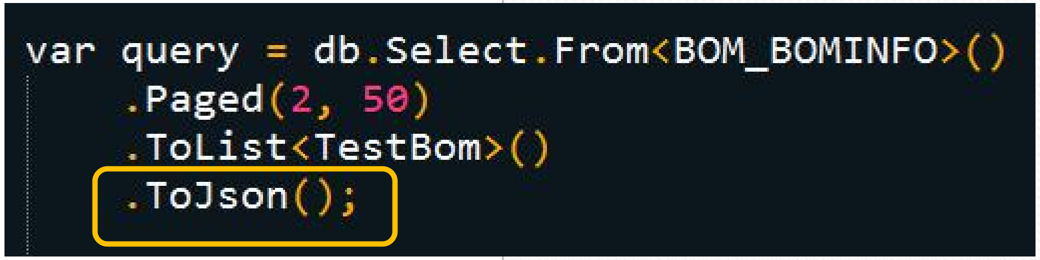


```
public void AutoMap2()
{
    Mapper.CreateMap<Customer, CustomerViewItem>()
        .ConvertUsing<CustomerToCustomerViewItemConverter>();

    var customerData = GetCustomerFromDB();

    Customer customer = Mapper.Map<Customer, CustomerViewItem>(customerData);

    Response.Write(customer.CustomerName);
}
```

擴充方法範例 – 製作ToJson

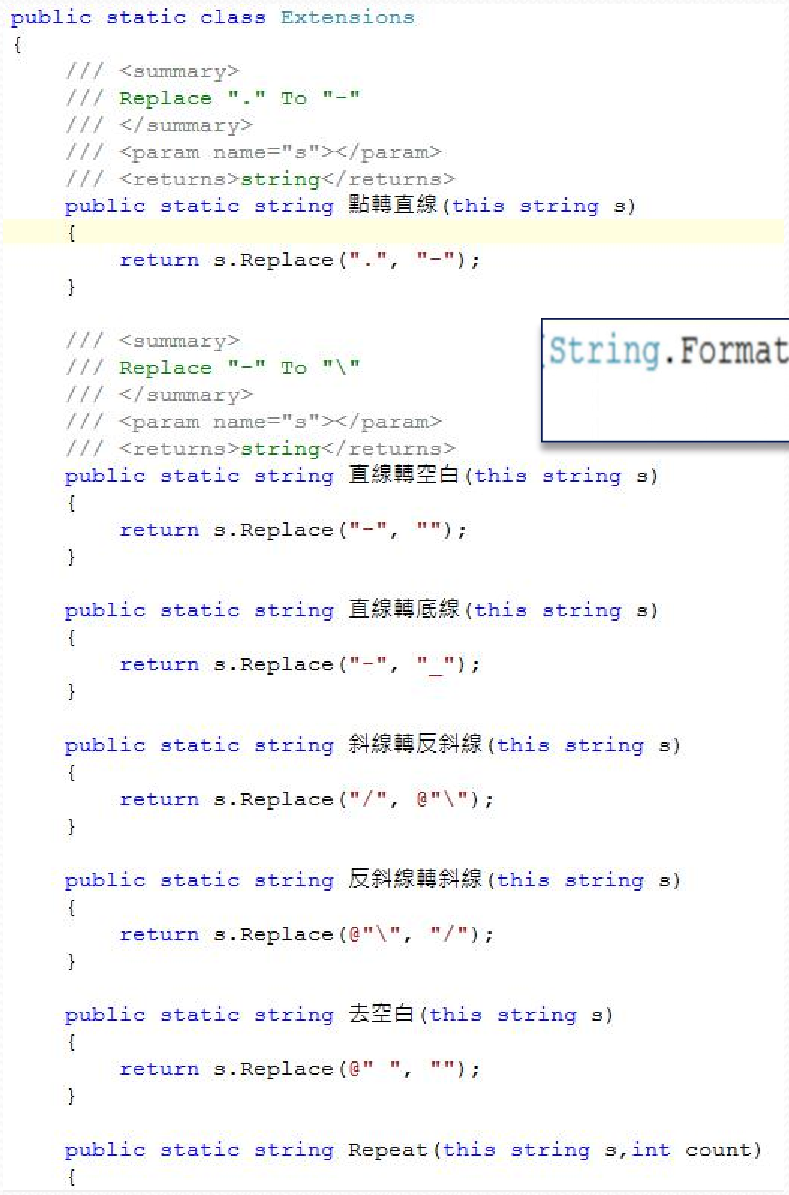


```
/// <summary>
/// 擴充Queryable class
/// </summary>
public static class QueryableExtensions
{
    /// <summary>
    /// 回傳 ExtJS 專用 Json 字串 ex:{"root":"Rows","totalCount":5,"Rows":[{"AVLDate":new Date(1226937
    /// </summary>
    /// <typeparam name="T">型別</typeparam>
    /// <param name="source">Queryable 來源</param>
    /// <param name="totalCount">額外傳入總筆數</param>
    /// <param name="jsonFormatting">格式化</param>
    /// <returns>Json 字串</returns>
    public static string ToJson<T>(this IQueryable<T> source, int? totalCount,
        JsonFormatting jsonFormatting) where T: class
    {

        object objContext;
        if (totalCount != null)
        {
            objContext = new
            {
                root = "Rows",
                totalCount,
                Rows = source
            };
        }
        else
            objContext = new { Rows = source };

        var ObjJson = JObject.FromObject(objContext);

        var formatting = Formatting.None;
        if (jsonFormatting == JsonFormatting.Indented)
            formatting = Formatting.Indented;

        return JsonConvert.SerializeObject(ObjJson, formatting, new JavaScriptDateTimeConverter());
```

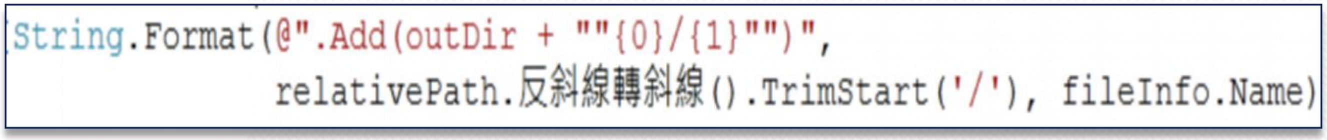


```
var query = db.Select.From<BOM_BOMINFO>()
    .Paged(2, 50)
    .ToList<TestBom>()
    .ToJson();
```

擴充方法簡單介紹 – 將常用的String 處理

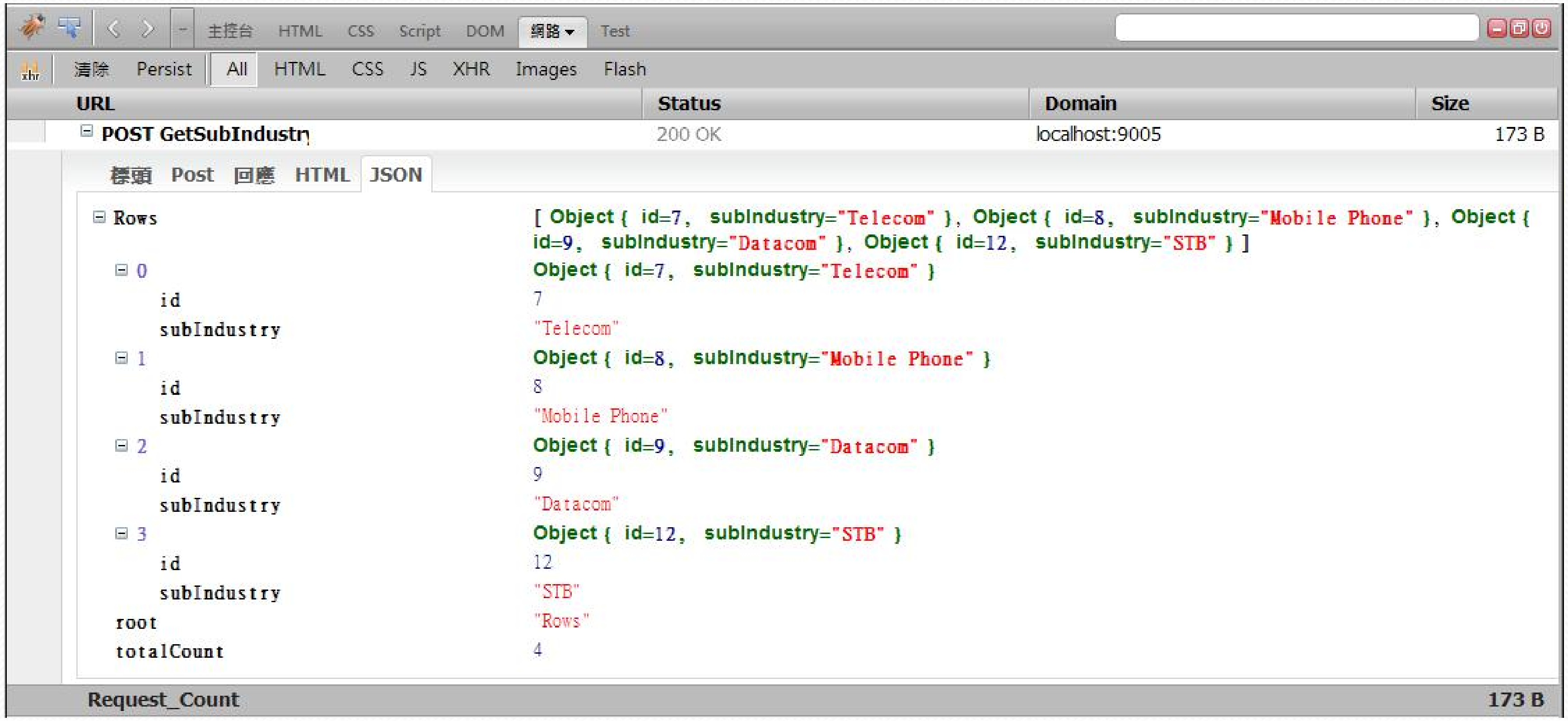


```
public static class Extensions
{
    /// <summary>
    /// Replace "." To "-"
    /// </summary>
    /// <param name="s"></param>
    /// <returns>string</returns>
    public static string 點轉直線(this string s)
    {
        return s.Replace(".", "-");
    }

    /// <summary>
    /// Replace "-" To "\"
    /// </summary>
    /// <param name="s"></param>
    /// <returns>string</returns>
    public static string 直線轉空白(this string s)
    {
        return s.Replace("-", "");
    }

    public static string 直線轉底線(this string s)
    {
        return s.Replace("-", "_");
    }

    public static string 斜線轉反斜線(this string s)
    {
        return s.Replace("/", @"\");
    }

    public static string 反斜線轉斜線(this string s)
    {
        return s.Replace(@"\", "/");
    }

    public static string 去空白(this string s)
    {
        return s.Replace(@" ", "");
    }

    public static string Repeat(this string s,int count)
    {
```



```
String.Format(@".Add(outDir + ""{0}/{1}"")",
                relativePath.反斜線轉斜線().TrimStart('/'), fileInfo.Name)
```

JSON Data View (Use FireBug)


主控台 HTML CSS Script DOM 網路 Test
清除 Persist All HTML CSS JS XHR Images Flash
標頭 Post 回應 HTML JSON

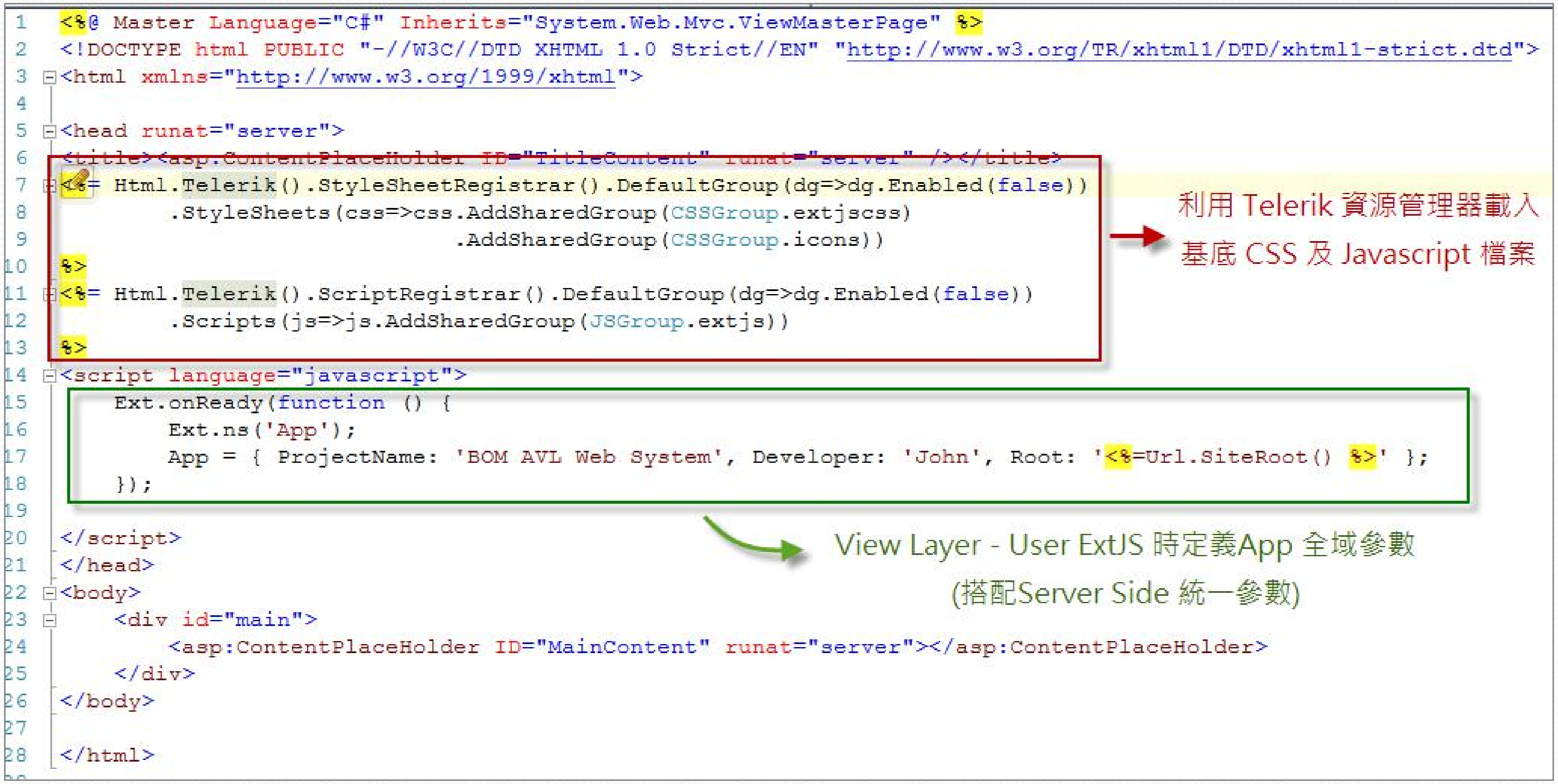
URL	Status	Domain	Size
POST GetSubIndustry	200 OK	localhost:9005	173 B

```
⊟ Rows              [ Object { id=7, subIndustry="Telecom" }, Object { id=8, subIndustry="Mobile Phone" }, Object {
                    id=9, subIndustry="Datacom" }, Object { id=12, subIndustry="STB" } ]
  ⊟ 0               Object { id=7, subIndustry="Telecom" }
      id            7
      subIndustry   "Telecom"
  ⊟ 1               Object { id=8, subIndustry="Mobile Phone" }
      id            8
      subIndustry   "Mobile Phone"
  ⊟ 2               Object { id=9, subIndustry="Datacom" }
      id            9
      subIndustry   "Datacom"
  ⊟ 3               Object { id=12, subIndustry="STB" }
      id            12
      subIndustry   "STB"
  root              "Rows"
  totalCount        4
```

Request_Count 173 B

專案建立設定流程(My Pattern)

- SiteMaster



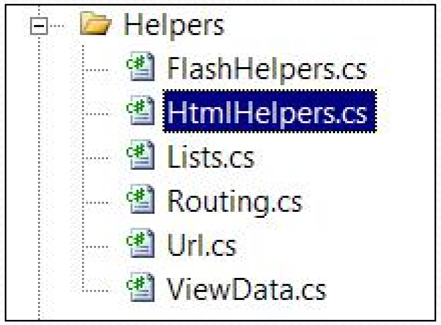
```
<%@ Master Language="C#" Inherits="System.Web.Mvc.ViewMasterPage" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">
<title><asp:ContentPlaceHolder ID="TitleContent" runat="server" /></title>
<%= Html.Telerik().StyleSheetRegistrar().DefaultGroup(dg=>dg.Enabled(false))
        .StyleSheets(css=>css.AddSharedGroup(CSSGroup.extjscss)
                             .AddSharedGroup(CSSGroup.icons))
%>
<%= Html.Telerik().ScriptRegistrar().DefaultGroup(dg=>dg.Enabled(false))
        .Scripts(js=>js.AddSharedGroup(JSGroup.extjs))
%>
<script language="javascript">
    Ext.onReady(function () {
        Ext.ns('App');
        App = { ProjectName: 'BOM AVL Web System', Developer: 'John', Root: '<%=Url.SiteRoot() %>' };
    });

</script>
</head>
<body>
    <div id="main">
        <asp:ContentPlaceHolder ID="MainContent" runat="server"></asp:ContentPlaceHolder>
    </div>
</body>

</html>
```

利用 Telerik 資源管理器載入
基底 CSS 及 Javascript 檔案

View Layer - User ExtJS 時定義App 全域參數
(搭配Server Side 統一參數)

搭配資源管理第二種引入CSS 及 JS 方法

定義 HtmlHelper



由於開發時會陸續開發前端UI 類別，我的習慣是**一個類別一個檔案**，如果一個個加入 ScriptsManager , 可能會有點煩瑣，這時可以利用 HTMLHelper 先引入，最終利用T4 一次將所有檔案壓縮並引入

HTMLHelper : 預先定義好的路徑，只需傳入參數

例如:

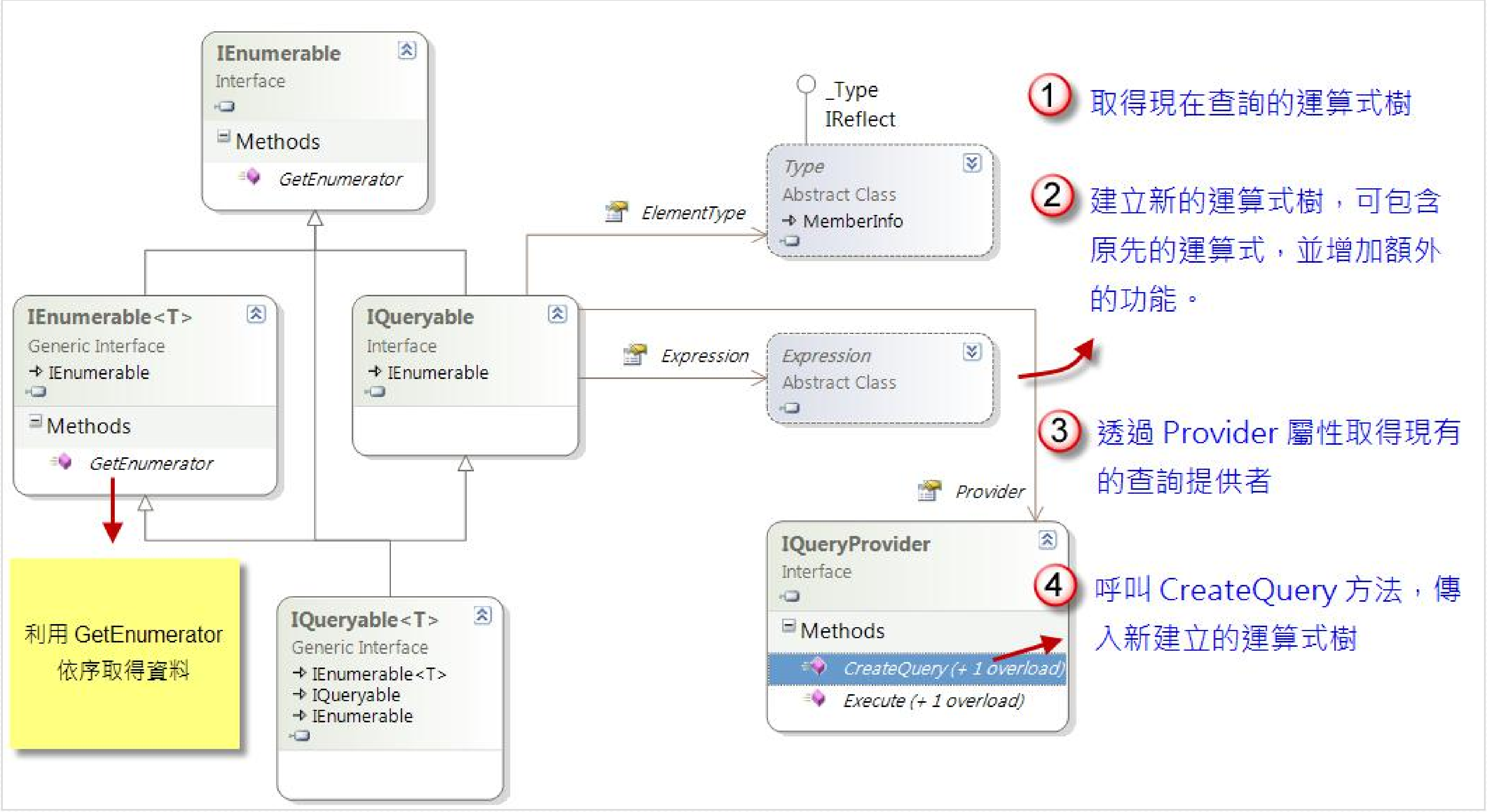
```
<%=Html.ExtUXCSS("FileUploadField.css")%>
<%=Html.Script("extux/FileUploadField.js")%>
<%=Html.UIScript("UploadWizard/WizardTabPanel.js")
%>
<%=Html.UIScript("UploadWizard/index.js") %>
```

LINQ 心得

- 統一的資料查詢語言，使用熟悉的查詢運算式
- 編譯器會為我們做大量的驗證
- VS2008/2010 提供的好用 IntelliSense, 幫助我們有效撰寫查詢運算式。
- ORM 系統必須留意查詢算式的寫法，因為有可能造成的 SQL Command 會導致資料庫執行效能變差(不過這部份與下 TSQL 一樣都需要調校)
- LINQ 也允許直接下 T-SQL 指令，在某些狀況下，如果查詢運算式無法完成式，手動寫指令反而是好方法。
- 研究如何寫自訂的查詢語法，可以有效提高效率。

IQueryable 介面

IQueryable 提供查詢的介面，執行查詢運算式取得我們要的結果



利用 GetEnumerator 依序取得資料

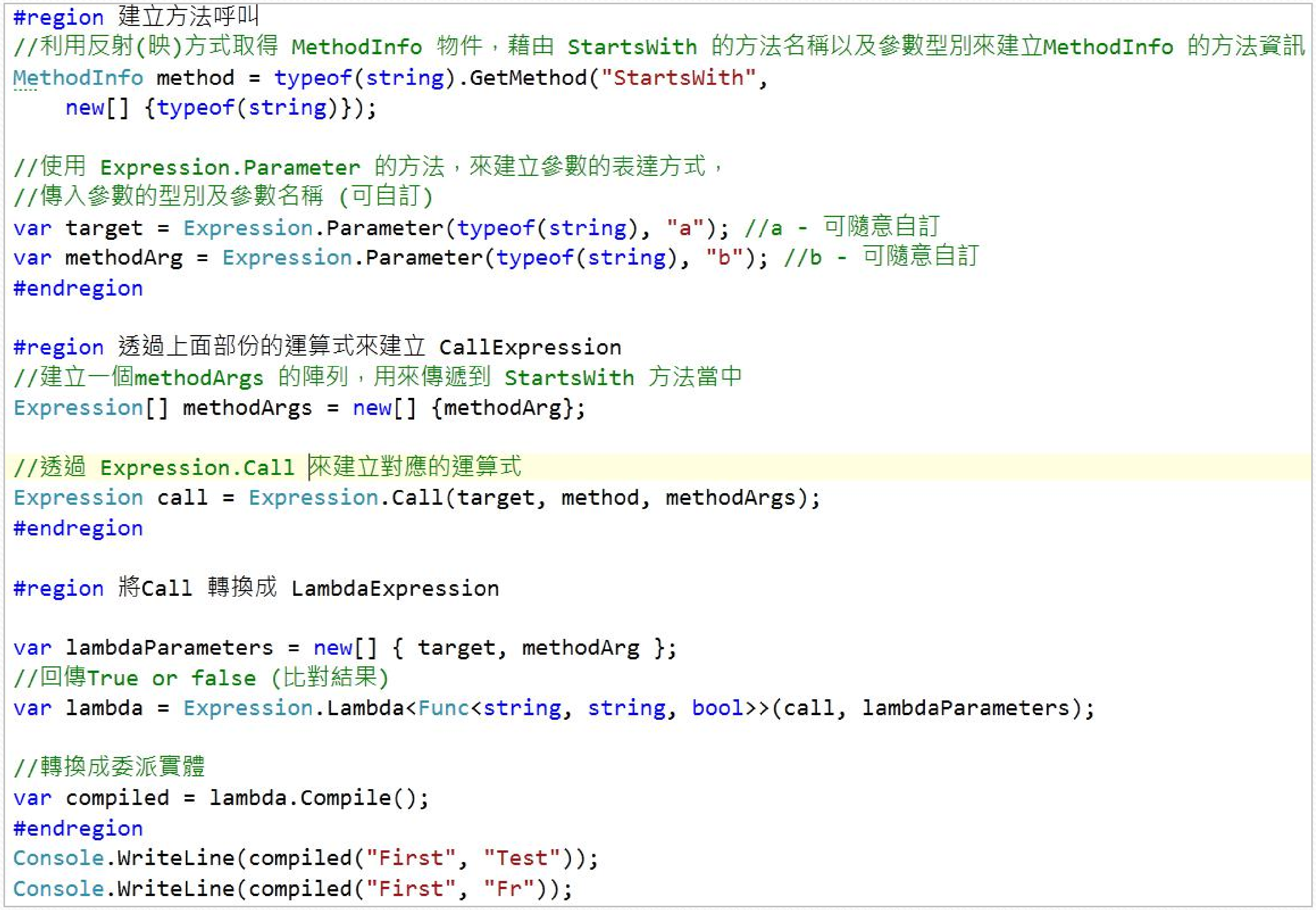
1 取得現在查詢的運算式樹

2 建立新的運算式樹，可包含原先的運算式，並增加額外的功能。

3 透過 Provider 屬性取得現有的查詢提供者

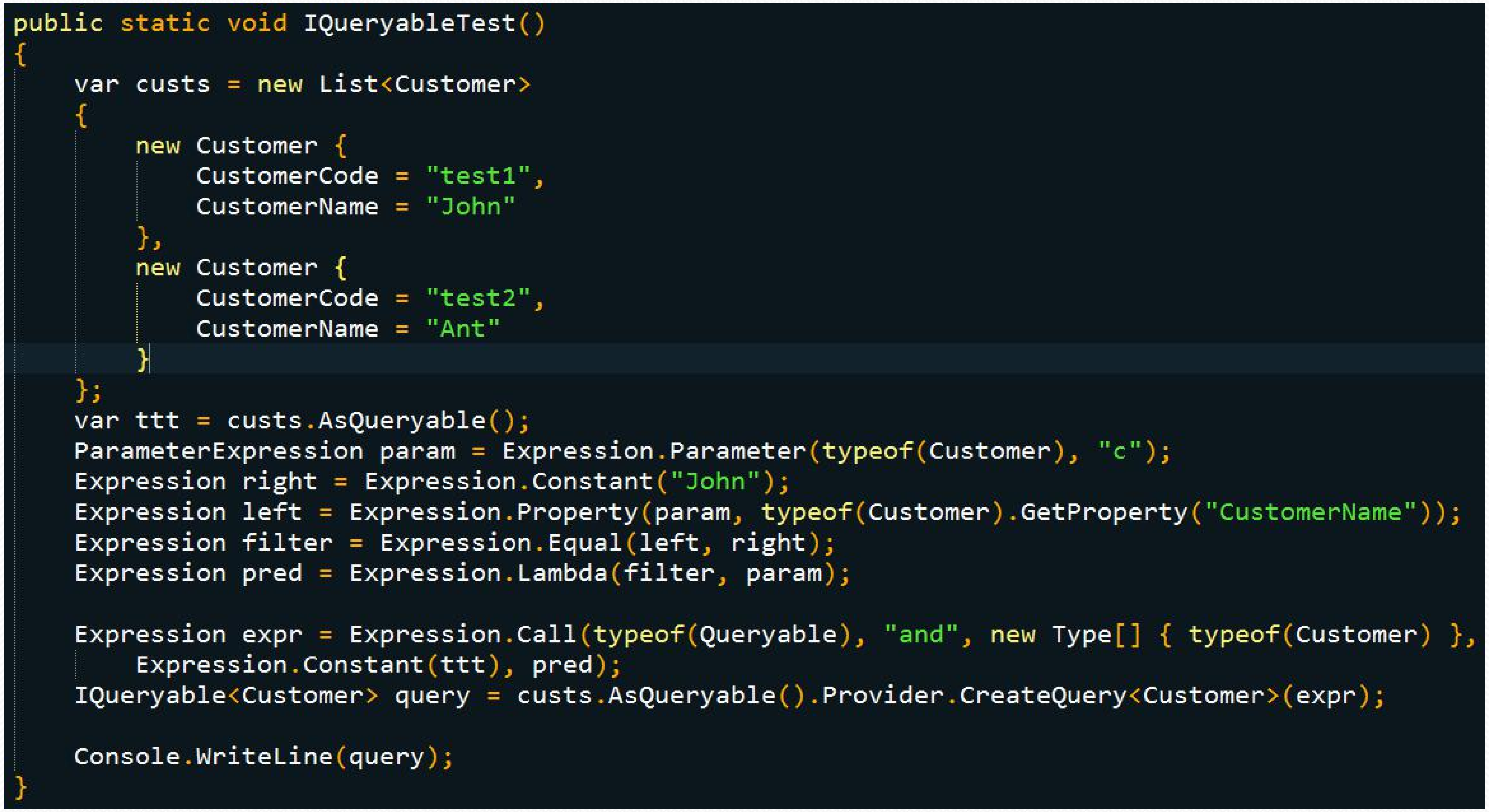
4 呼叫 CreateQuery 方法，傳入新建立的運算式樹

Expression 運算式樹



```
#region 建立方法呼叫
//利用反射(映)方式取得 MethodInfo 物件，藉由 StartsWith 的方法名稱以及參數型別來建立MethodInfo 的方法資訊
MethodInfo method = typeof(string).GetMethod("StartsWith",
    new[] {typeof(string)});

//使用 Expression.Parameter 的方法，來建立參數的表達方式，
//傳入參數的型別及參數名稱 (可自訂)
var target = Expression.Parameter(typeof(string), "a"); //a - 可隨意自訂
var methodArg = Expression.Parameter(typeof(string), "b"); //b - 可隨意自訂
#endregion

#region 透過上面部份的運算式來建立 CallExpression
//建立一個methodArgs 的陣列，用來傳遞到 StartsWith 方法當中
Expression[] methodArgs = new[] {methodArg};

//透過 Expression.Call 來建立對應的運算式
Expression call = Expression.Call(target, method, methodArgs);
#endregion

#region 將Call 轉換成 LambdaExpression

var lambdaParameters = new[] { target, methodArg };
//回傳True or false (比對結果)
var lambda = Expression.Lambda<Func<string, string, bool>>(call, lambdaParameters);

//轉換成委派實體
var compiled = lambda.Compile();
#endregion
Console.WriteLine(compiled("First", "Test"));
Console.WriteLine(compiled("First", "Fr"));
```

運算樹範例 -2
製作LINQ 串接語法 And



```
public static void IQueryableTest()
{
    var custs = new List<Customer>
    {
        new Customer {
            CustomerCode = "test1",
            CustomerName = "John"
        },
        new Customer {
            CustomerCode = "test2",
            CustomerName = "Ant"
        }
    };
    var ttt = custs.AsQueryable();
    ParameterExpression param = Expression.Parameter(typeof(Customer), "c");
    Expression right = Expression.Constant("John");
    Expression left = Expression.Property(param, typeof(Customer).GetProperty("CustomerName"));
    Expression filter = Expression.Equal(left, right);
    Expression pred = Expression.Lambda(filter, param);

    Expression expr = Expression.Call(typeof(Queryable), "and", new Type[] { typeof(Customer) },
        Expression.Constant(ttt), pred);
    IQueryable<Customer> query = custs.AsQueryable().Provider.CreateQuery<Customer>(expr);

    Console.WriteLine(query);
}
```

運算式樹稍複雜..

```
Expression<Func<Product, bool>> predicate = c => true;
```

等同於

⬇

```
var predicate = PredicateBuilder.True <Product> ();
```

```
IQueryable<Product> SearchProducts (params string[] keywords) {
        var predicate = PredicateBuilder.False<Product>();
        foreach (string keyword in keywords) {
                string temp = keyword;
                predicate = predicate.Or (p => p.Description.Contains (temp));
        }
return dataContext.Products.Where (predicate); }
```

製作自己的And

- 假設: 有一個多條件查詢，系統要根據使用者是否有選擇項目來判斷需不需要組合查詢，一般會這樣做

SQL Command

```
string sqlCmd = "Select * From Customer Where 1=1 ";
string  customerName = Request["customer"];
string user = Request["user"]
string customerCode = Request["customerCode"]
string customerRegion = Request["customerRegion"]
if(! String.IsNullOrEmpty(customerName ))
                SqlCmd = SqlCmd + " and customerName ='" + customerName + "'";
if(! String.IsNullOrEmpty(user ))
                SqlCmd = SqlCmd + " and user='" + user+ "'";
if(! String.IsNullOrEmpty(customerCode ))
                SqlCmd = SqlCmd + " and customerCode ='" + customerCode + "'";
```

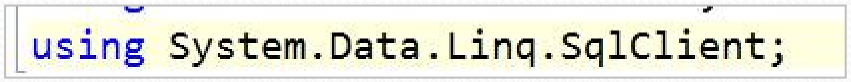
LINQ

```
var query = from c in db.Customer select c;
if(! String.IsNullOrEmpty(customerName ))
                query = query.where(c=>c. customerName == customerName)
                ............................................... Repeat
```

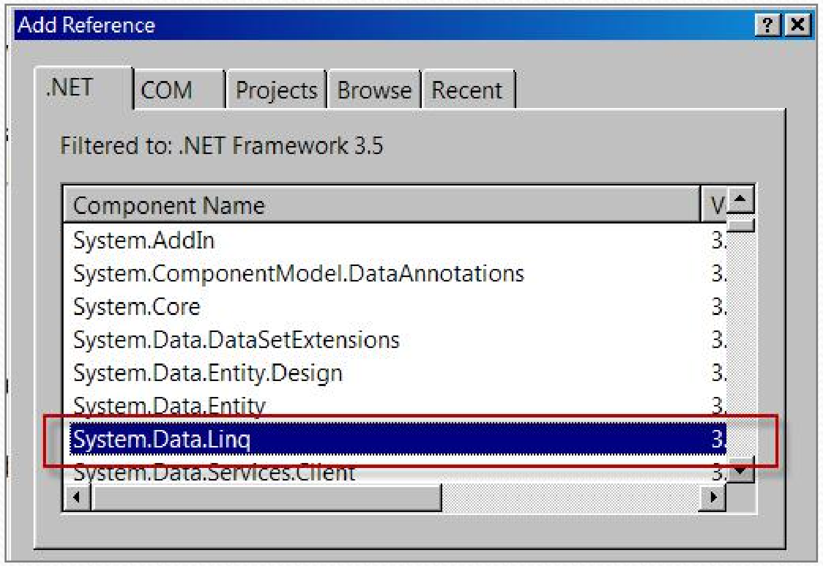
如果能..

```
var query = db.select.From<Customer>()
.TryAnd(c=>c.customerName == customerName)
```

要擴充自己的SQL 對應語法需先引入 System.Data.LINQ



```
using System.Data.Linq.SqlClient;
```



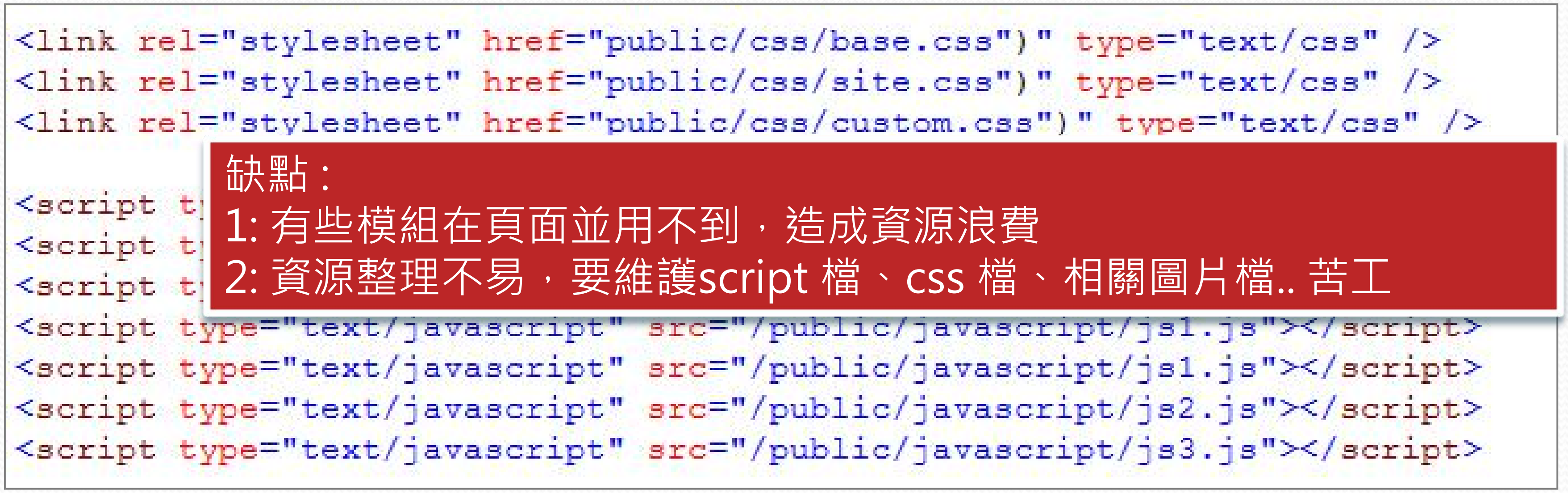
Filtered to: .NET Framework 3.5

Component Name	V
System.AddIn	3.
System.ComponentModel.DataAnnotations	3.
System.Core	3.
System.Data.DataSetExtensions	3.
System.Data.Entity.Design	3.
System.Data.Entity	3.
System.Data.Linq	3.
System.Data.Services.Client	3.

JAVASCRIPT&CSS FILE MANAGEMENT 心得分享

JAVASCRIPT &CSS FILE MANAGEMENT 心得分享

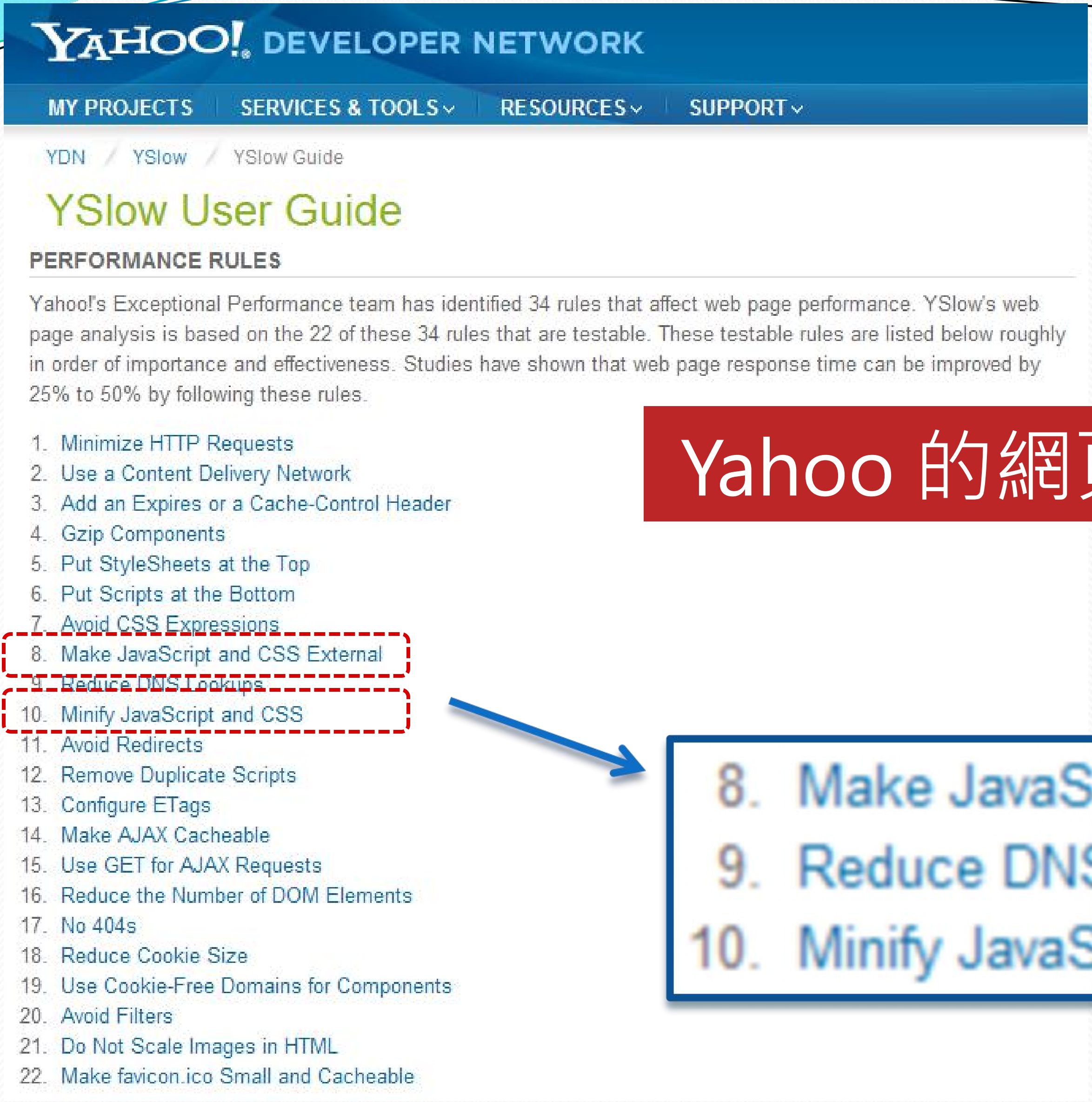
現今的俗稱 Web 2.0 , 強調 AJAX 及 使用者介面操作的便利性，而這表示Javascript 及CSS的使用量佔相當大的比重，在引入資源檔案時最常使用的手法可能這樣寫 ....



```
<link rel="stylesheet" href="public/css/base.css")" type="text/css" />
<link rel="stylesheet" href="public/css/site.css")" type="text/css" />
<link rel="stylesheet" href="public/css/custom.css")" type="text/css" />

<script t
<script t
<script t
<script type="text/javascript" src="/public/javascript/js1.js"></script>
<script type="text/javascript" src="/public/javascript/js1.js"></script>
<script type="text/javascript" src="/public/javascript/js2.js"></script>
<script type="text/javascript" src="/public/javascript/js3.js"></script>
```

缺點：
1: 有些模組在頁面並用不到，造成資源浪費
2: 資源整理不易，要維護script 檔、css 檔、相關圖片檔.. 苦工

Yahoo 的網頁效能建議指南



YAHOO! DEVELOPER NETWORK

MY PROJECTS | SERVICES & TOOLS | RESOURCES | SUPPORT

YDN / YSlow / YSlow Guide

YSlow User Guide

PERFORMANCE RULES

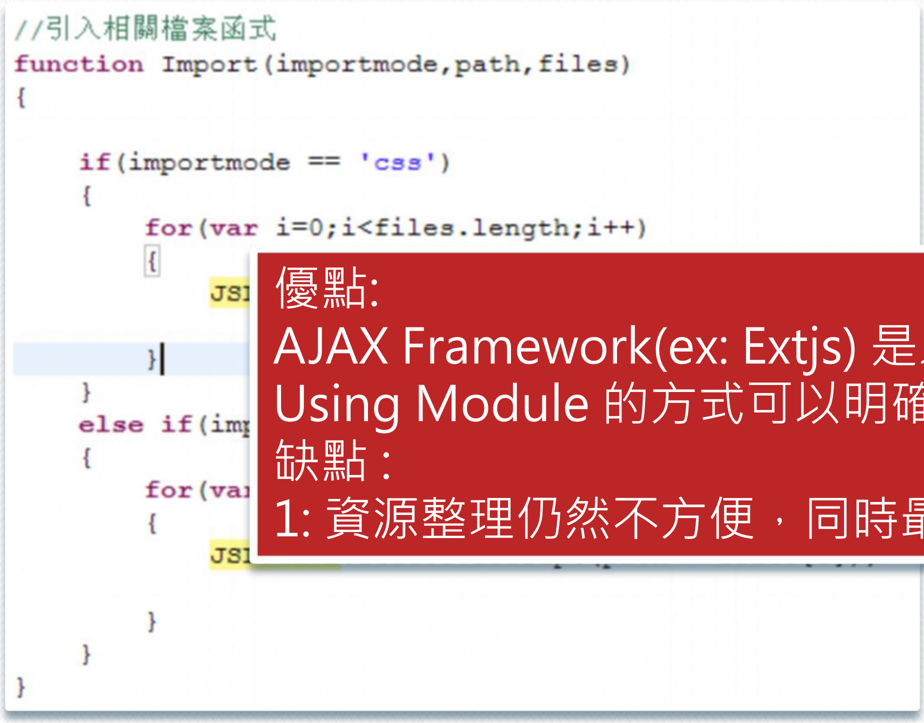
Yahoo!'s Exceptional Performance team has identified 34 rules that affect web page performance. YSlow's web page analysis is based on the 22 of these 34 rules that are testable. These testable rules are listed below roughly in order of importance and effectiveness. Studies have shown that web page response time can be improved by 25% to 50% by following these rules.

1. Minimize HTTP Requests
2. Use a Content Delivery Network
3. Add an Expires or a Cache-Control Header
4. Gzip Components
5. Put StyleSheets at the Top
6. Put Scripts at the Bottom
7. Avoid CSS Expressions
8. Make JavaScript and CSS External
9. Reduce DNS Lookups
10. Minify JavaScript and CSS
11. Avoid Redirects
12. Remove Duplicate Scripts
13. Configure ETags
14. Make AJAX Cacheable
15. Use GET for AJAX Requests
16. Reduce the Number of DOM Elements
17. No 404s
18. Reduce Cookie Size
19. Use Cookie-Free Domains for Components
20. Avoid Filters
21. Do Not Scale Images in HTML
22. Make favicon.ico Small and Cacheable

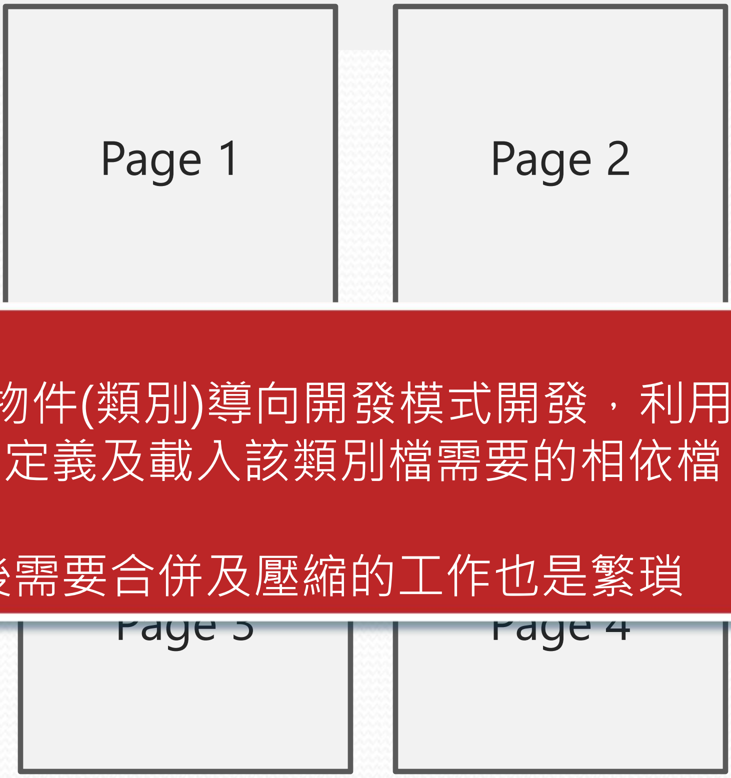
8. Make JavaScript and CSS External
9. Reduce DNS Lookups
10. Minify JavaScript and CSS

JAVASCRIPT &CSS FILE MANAGEMENT 心得分享

後來，有了Javascript 函式庫 – 動態資源載入器(類似C# 的using)，在開發模式時切成一個一個小(類別)檔案，在發佈時合併並壓縮(利用YUI Compressor) 成小模組,可能換成這樣寫..

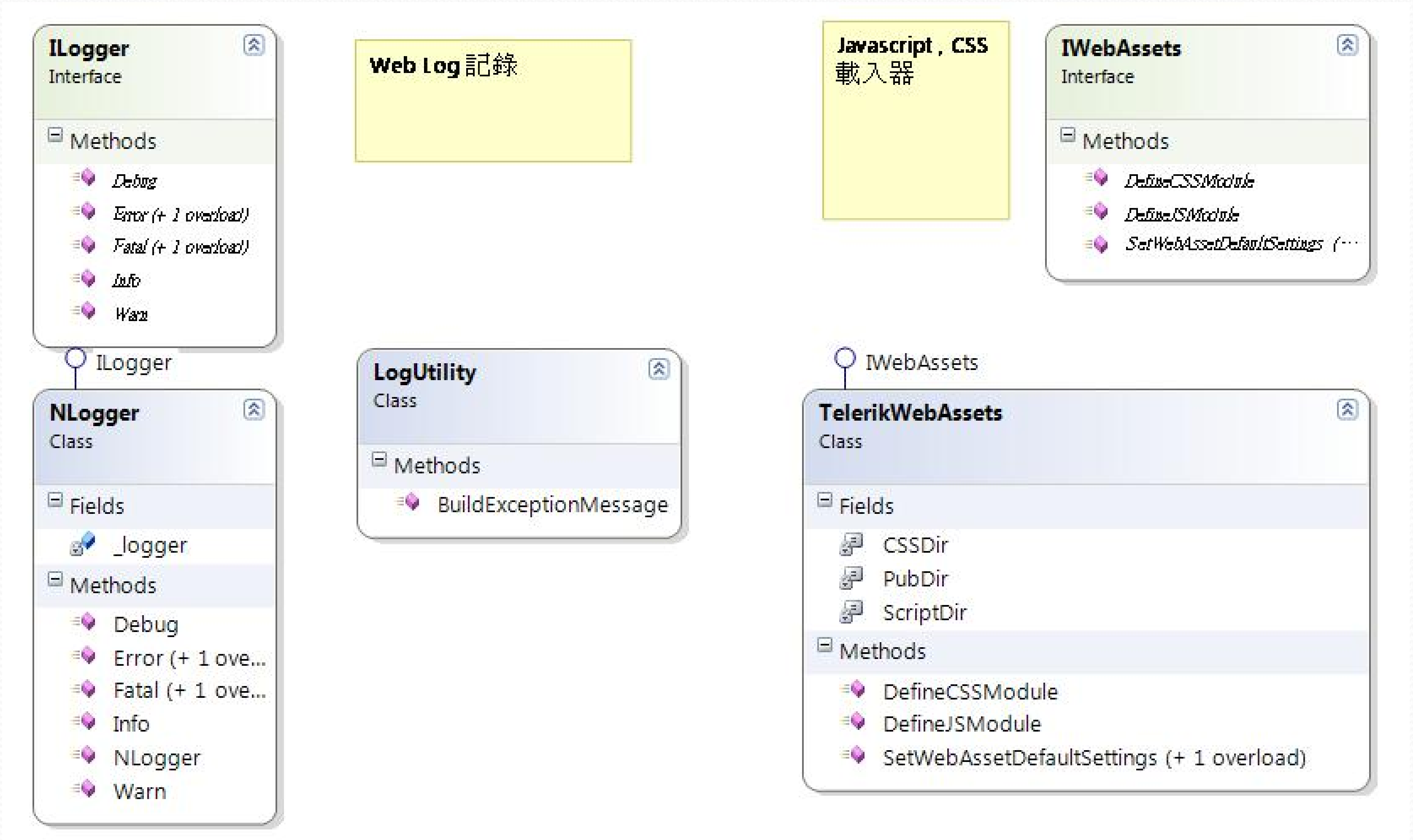


```
//引入相關檔案函式
function Import(importmode,path,files)
{

    if(importmode == 'css')
    {
        for(var i=0;i<files.length;i++)
        {
            JSI
        }
    }
    else if(imp
    {
        for(va
        {
            JSI
        }
    }
}
```

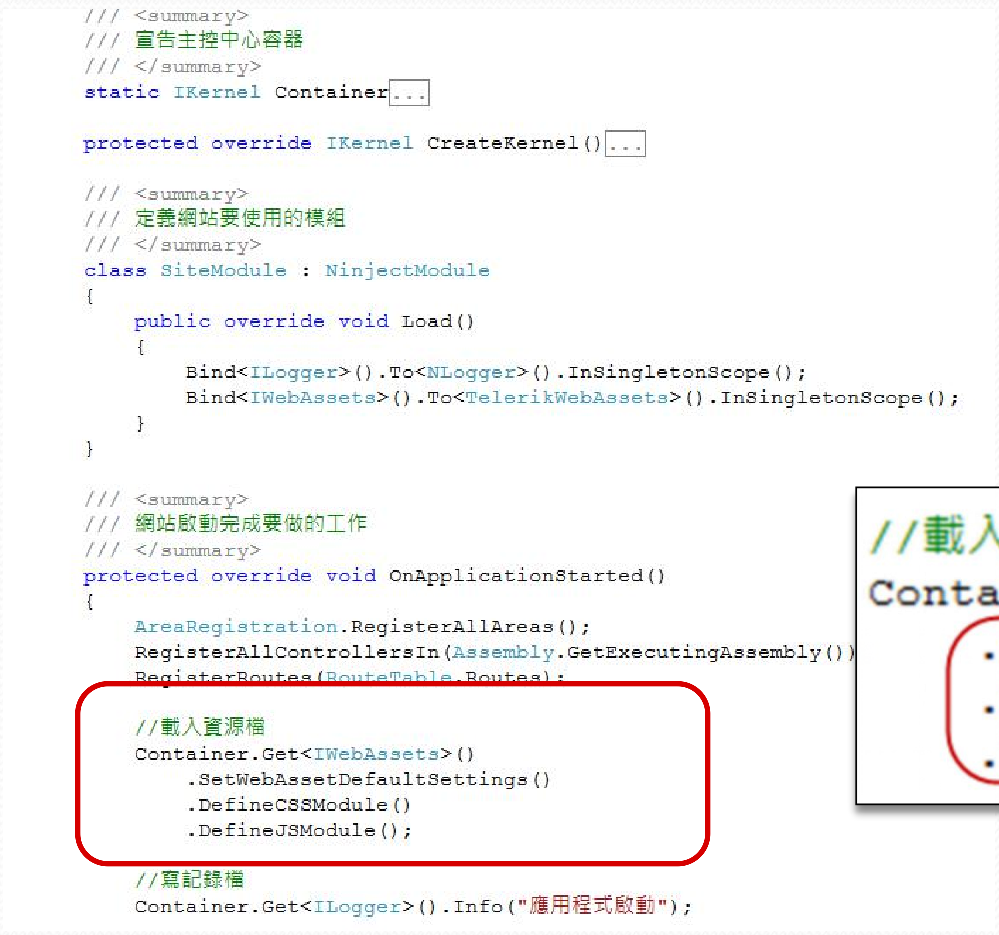


優點:
AJAX Framework(ex: Extjs) 是以物件(類別)導向開發模式開發，利用Using Module 的方式可以明確的定義及載入該類別檔需要的相依檔
缺點：
1: 資源整理仍然不方便，同時最後需要合併及壓縮的工作也是繁瑣

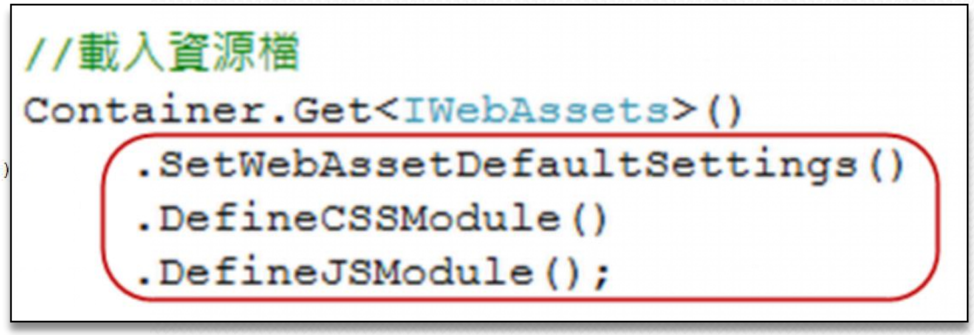
類別圖


ILogger
Interface
Methods
Debug
Error (+ 1 overload)
Fatal (+ 1 overload)
Info
Warn
Web Log 記錄
Javascript , CSS 載入器
IWebAssets
Interface
Methods
DefineCSSModule
DefineJSModule
SetWebAssetDefaultSettings (⋯
ILogger
NLogger
Class
Fields
_logger
Methods
Debug
Error (+ 1 ove...
Fatal (+ 1 ove...
Info
NLogger
Warn
LogUtility
Class
Methods
BuildExceptionMessage
IWebAssets
TelerikWebAssets
Class
Fields
CSSDir
PubDir
ScriptDir
Methods
DefineCSSModule
DefineJSModule
SetWebAssetDefaultSettings (+ 1 overload)

依賴注入 (NinjectModule)

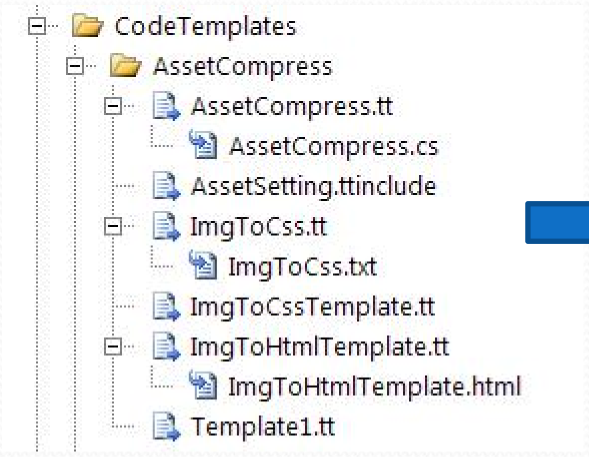


```
/// <summary>
/// 宣告主控中心容器
/// </summary>
static IKernel Container...

protected override IKernel CreateKernel()...

/// <summary>
/// 定義網站要使用的模組
/// </summary>
class SiteModule : NinjectModule
{
    public override void Load()
    {
        Bind<ILogger>().To<NLogger>().InSingletonScope();
        Bind<IWebAssets>().To<TelerikWebAssets>().InSingletonScope();
    }
}

/// <summary>
/// 網站啟動完成要做的工作
/// </summary>
protected override void OnApplicationStarted()
{
    AreaRegistration.RegisterAllAreas();
    RegisterAllControllersIn(Assembly.GetExecutingAssembly())
    RegisterRoutes(RouteTable.Routes);

    //載入資源檔
    Container.Get<IWebAssets>()
        .SetWebAssetDefaultSettings()
        .DefineCSSModule()
        .DefineJSModule();

    //寫記錄檔
    Container.Get<ILogger>().Info("應用程式啟動");
```

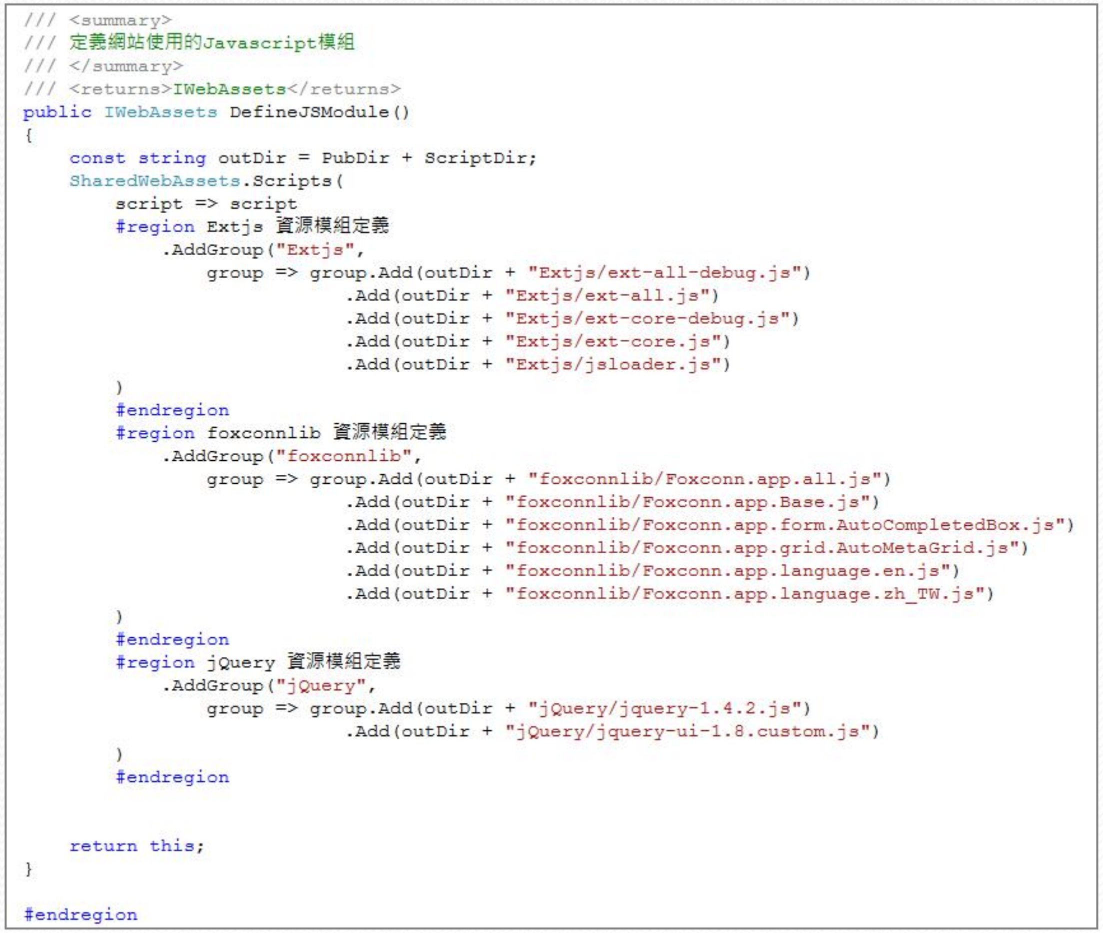


```
//載入資源檔
Container.Get<IWebAssets>()
    .SetWebAssetDefaultSettings()
    .DefineCSSModule()
    .DefineJSModule();
```

利用 T4 程式碼產生 + 壓縮 一次完成

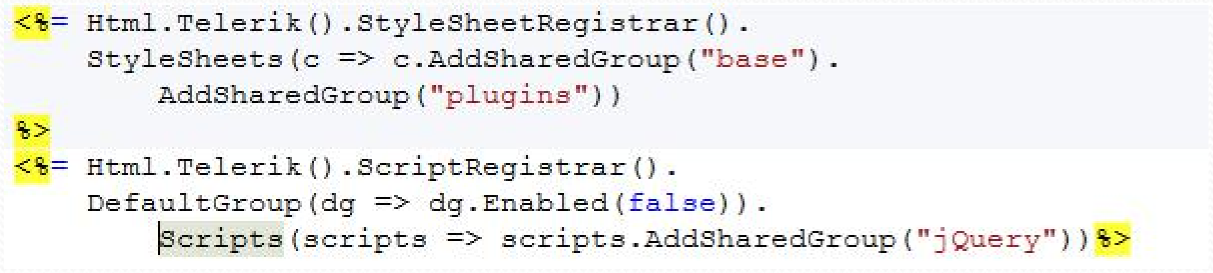


```
CodeTemplates
    AssetCompress
        AssetCompress.tt
            AssetCompress.cs
        AssetSetting.ttinclude
        ImgToCss.tt
            ImgToCss.txt
        ImgToCssTemplate.tt
        ImgToHtmlTemplate.tt
            ImgToHtmlTemplate.html
        Template1.tt
```



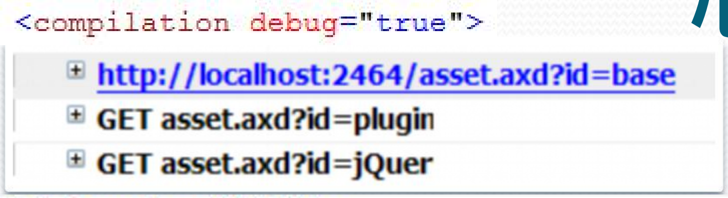
```csharp
/// <summary>
/// 定義網站使用的Javascript模組
/// </summary>
/// <returns>IWebAssets</returns>
public IWebAssets DefineJSModule()
{
    const string outDir = PubDir + ScriptDir;
    SharedWebAssets.Scripts(
        script => script
        #region Extjs 資源模組定義
            .AddGroup("Extjs",
                group => group.Add(outDir + "Extjs/ext-all-debug.js")
                            .Add(outDir + "Extjs/ext-all.js")
                            .Add(outDir + "Extjs/ext-core-debug.js")
                            .Add(outDir + "Extjs/ext-core.js")
                            .Add(outDir + "Extjs/jsloader.js")
        )
        #endregion
        #region foxconnlib 資源模組定義
            .AddGroup("foxconnlib",
                group => group.Add(outDir + "foxconnlib/Foxconn.app.all.js")
                            .Add(outDir + "foxconnlib/Foxconn.app.Base.js")
                            .Add(outDir + "foxconnlib/Foxconn.app.form.AutoCompletedBox.js")
                            .Add(outDir + "foxconnlib/Foxconn.app.grid.AutoMetaGrid.js")
                            .Add(outDir + "foxconnlib/Foxconn.app.language.en.js")
                            .Add(outDir + "foxconnlib/Foxconn.app.language.zh_TW.js")
        )
        #endregion
        #region jQuery 資源模組定義
            .AddGroup("jQuery",
                group => group.Add(outDir + "jQuery/jquery-1.4.2.js")
                            .Add(outDir + "jQuery/jquery-ui-1.8.custom.js")
        )
        #endregion


    return this;
}

#endregion
```

使用方式”:



```
<%= Html.Telerik().StyleSheetRegistrar().
    StyleSheets(c => c.AddSharedGroup("base").
        AddSharedGroup("plugins"))
%>
<%= Html.Telerik().ScriptRegistrar().
    DefaultGroup(dg => dg.Enabled(false)).
        Scripts(scripts => scripts.AddSharedGroup("jQuery"))%>
```

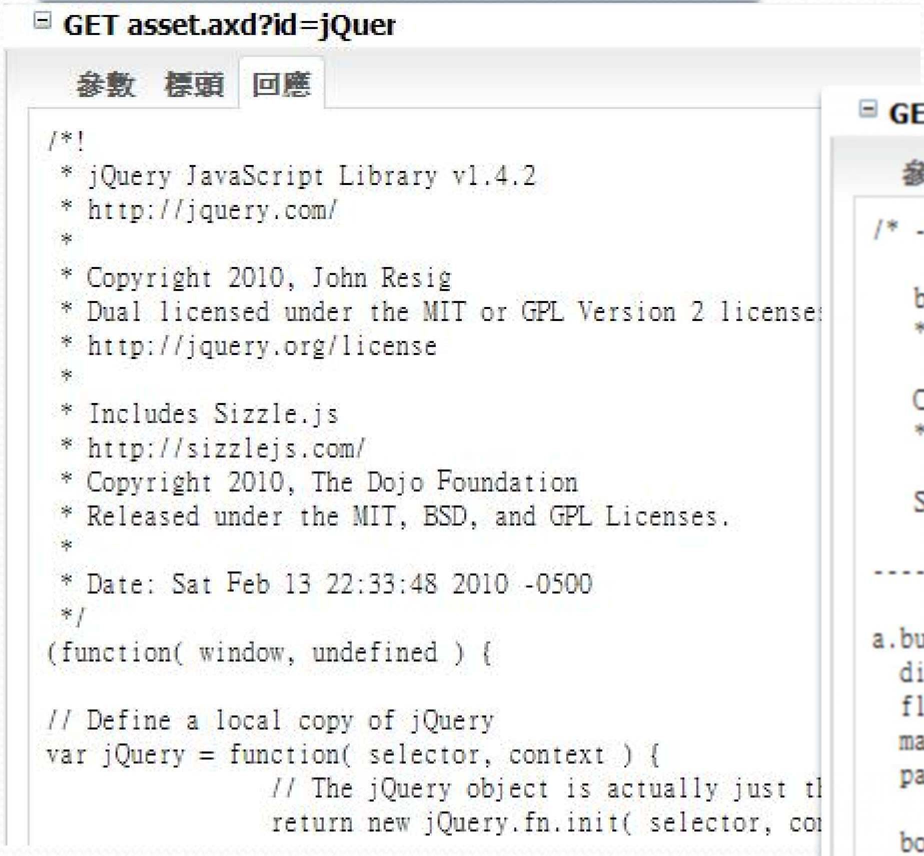
JAVASCRIPT &CSS FILE MANAGEMENT 心得分享



```
<compilation debug="true">
```

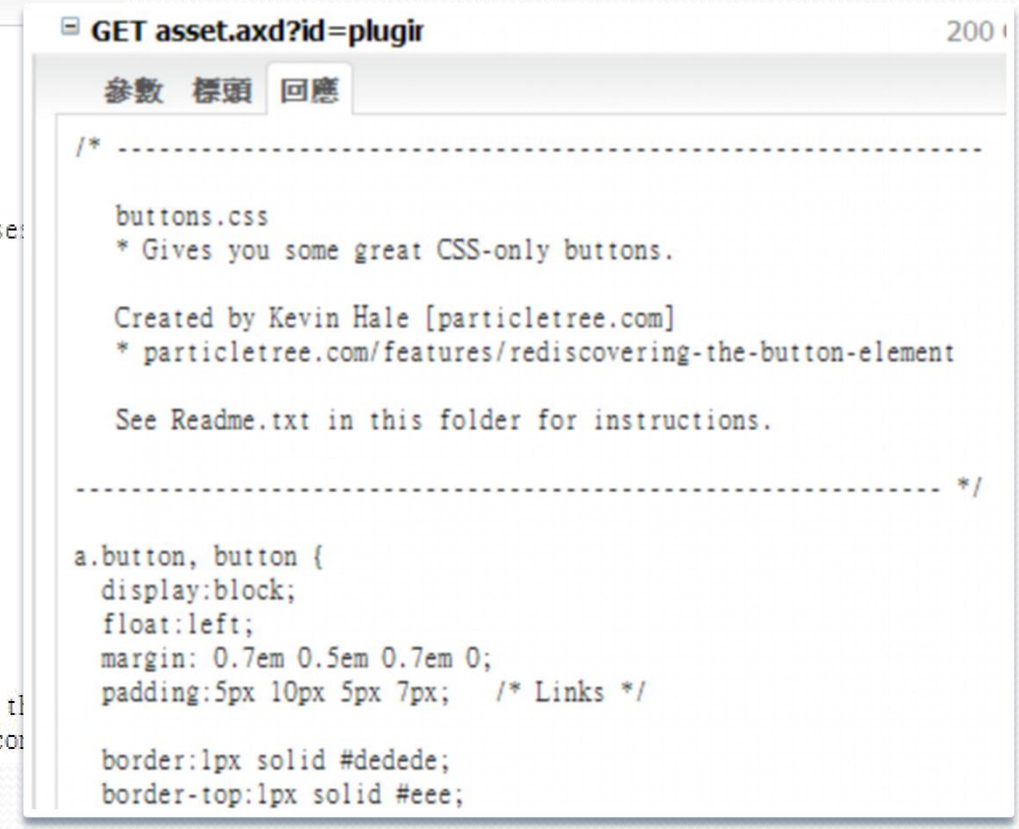
⊞ http://localhost:2464/asset.axd?id=base
⊞ GET asset.axd?id=plugin
⊞ GET asset.axd?id=jQuer

Web.config
Compilation debug = "true"
無壓縮，方便開發時期測試除錯



⊟ GET asset.axd?id=jQuer

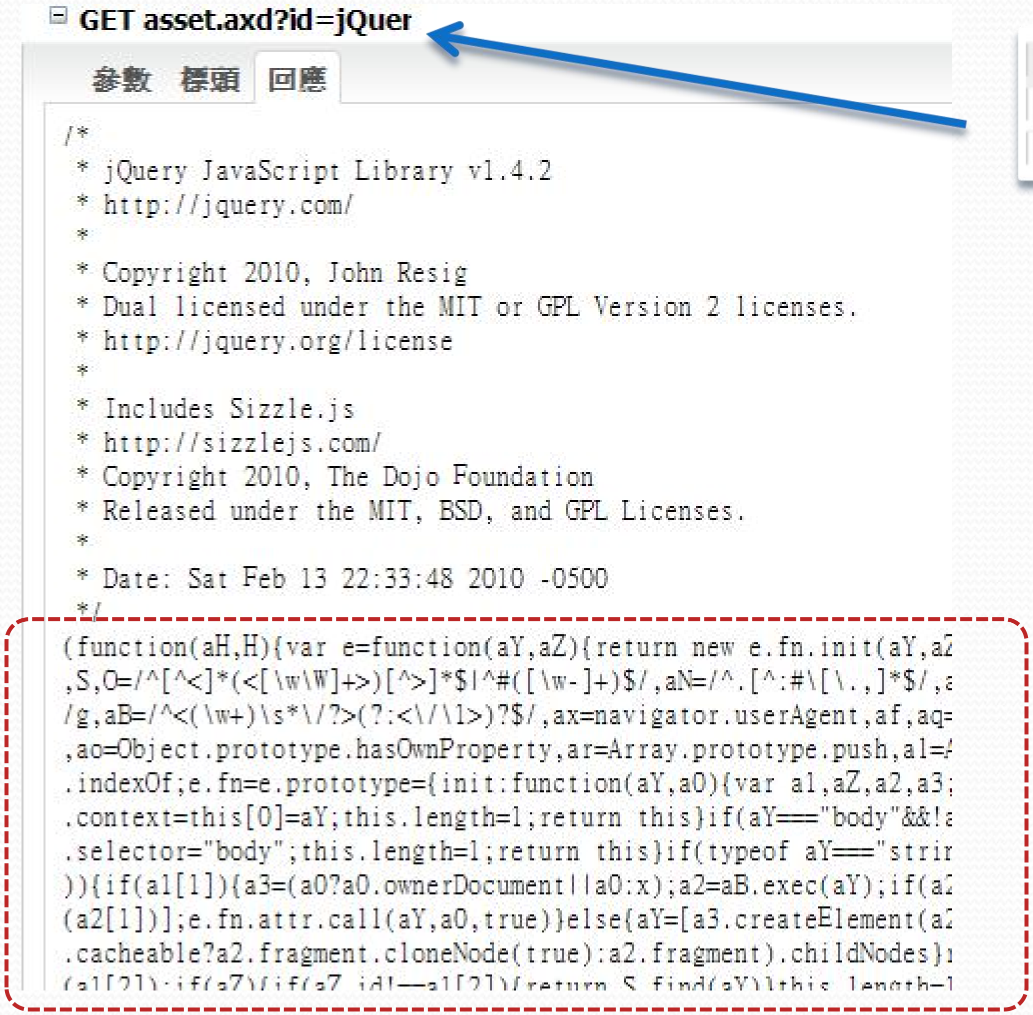
```
/*!
 * jQuery JavaScript Library v1.4.2
 * http://jquery.com/
 *
 * Copyright 2010, John Resig
 * Dual licensed under the MIT or GPL Version 2 license
 * http://jquery.org/license
 *
 * Includes Sizzle.js
 * http://sizzlejs.com/
 * Copyright 2010, The Dojo Foundation
 * Released under the MIT, BSD, and GPL Licenses.
 *
 * Date: Sat Feb 13 22:33:48 2010 -0500
 */
(function( window, undefined ) {

// Define a local copy of jQuery
var jQuery = function( selector, context ) {
		// The jQuery object is actually just t
		return new jQuery.fn.init( selector, co
```



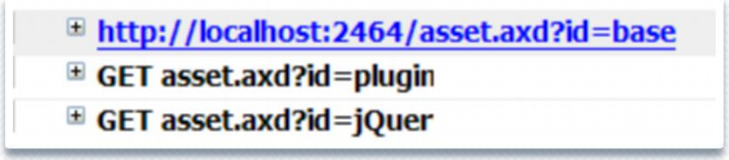
⊟ GET asset.axd?id=plugir 200

```
/* -----------------------------------------------------------------


   buttons.css
   * Gives you some great CSS-only buttons.

   Created by Kevin Hale [particletree.com]
   * particletree.com/features/rediscovering-the-button-element

   See Readme.txt in this folder for instructions.

-------------------------------------------------------------- */

a.button, button {
  display:block;
  float:left;
  margin: 0.7em 0.5em 0.7em 0;
  padding:5px 10px 5px 7px;   /* Links */

  border:1px solid #dedede;
  border-top:1px solid #eee;
```

```
<compilation debug="false">
```



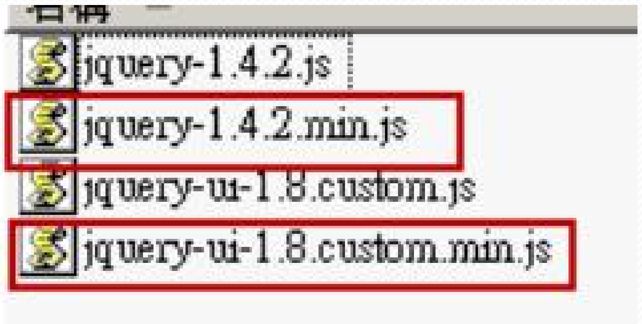
```
⊟ GET asset.axd?id=jQuer
參數  標頭  回應
/*
 * jQuery JavaScript Library v1.4.2
 * http://jquery.com/
 *
 * Copyright 2010, John Resig
 * Dual licensed under the MIT or GPL Version 2 licenses.
 * http://jquery.org/license
 *
 * Includes Sizzle.js
 * http://sizzlejs.com/
 * Copyright 2010, The Dojo Foundation
 * Released under the MIT, BSD, and GPL Licenses.
 *
 * Date: Sat Feb 13 22:33:48 2010 -0500
 */
(function(aH,H){var e=function(aY,aZ){return new e.fn.init(aY,aZ
,S,O=/^[^<]*(<[\w\W]+>)[^>]*$|^#([\w-]+)$/,aN=/^.[^:#\[\.,]*$/,a
/g,aB=/^<(\w+)\s*\/?>(?:<\/\1>)?$/,ax=navigator.userAgent,af,aq=
,ao=Object.prototype.hasOwnProperty,ar=Array.prototype.push,al=A
.indexOf;e.fn=e.prototype={init:function(aY,a0){var a1,aZ,a2,a3;
.context=this[0]=aY;this.length=1;return this}if(aY==="body"&&!a
.selector="body";this.length=1;return this}if(typeof aY==="strin
)){if(a1[1]){a3=(a0?a0.ownerDocument||a0:x);a2=aB.exec(aY);if(a2
(a2[1])];e.fn.attr.call(aY,a0,true)}else{aY=[a3.createElement(a2
.cacheable?a2.fragment.cloneNode(true):a2.fragment).childNodes}r
```



```
⊞ http://localhost:2464/asset.axd?id=base
⊞ GET asset.axd?id=plugin
⊞ GET asset.axd?id=jQuer
```

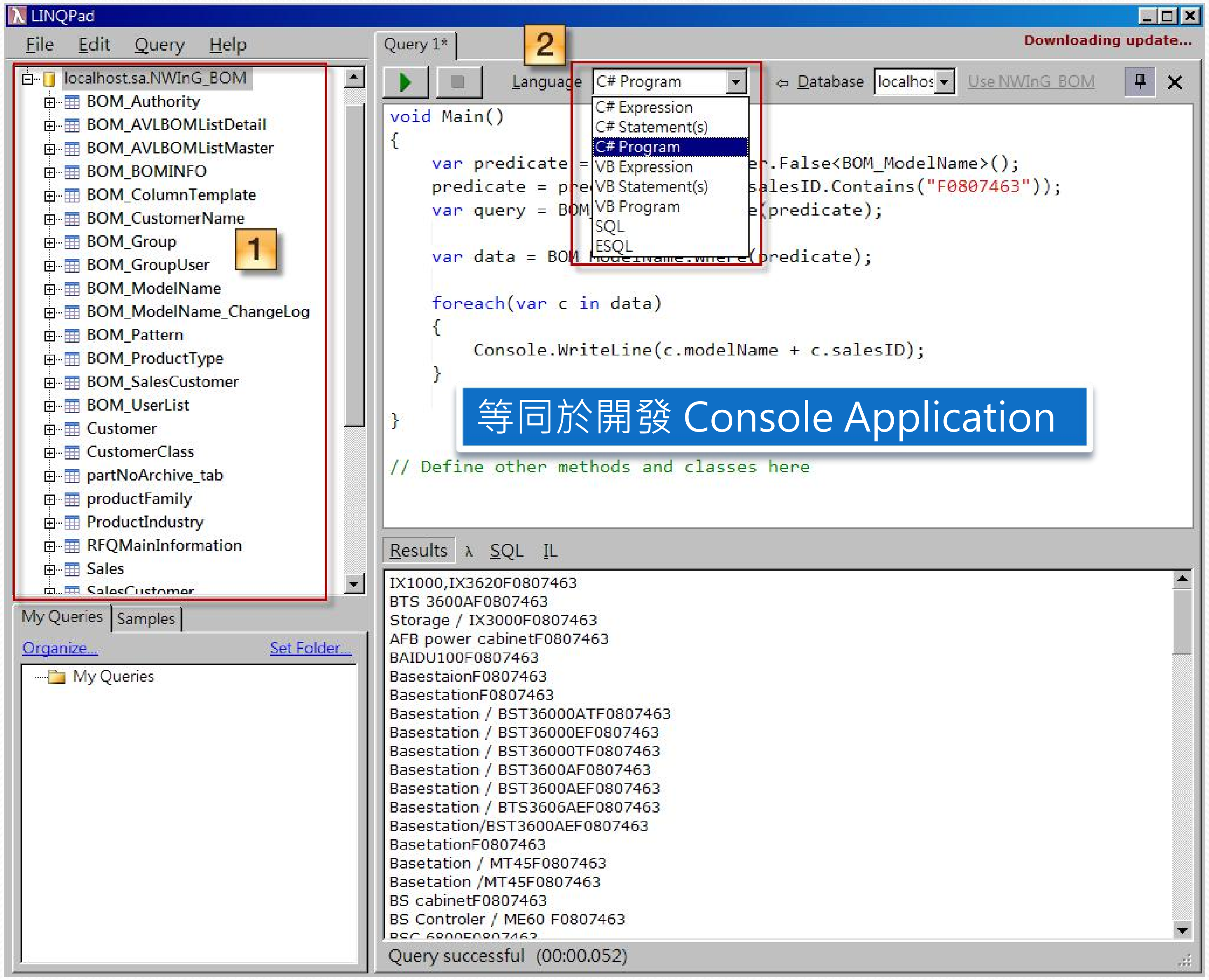
自動整合成單一檔案

自動抓取壓縮過的檔案



```
jquery-1.4.2.js
jquery-1.4.2.min.js
jquery-ui-1.8.custom.js
jquery-ui-1.8.custom.min.js
```

使用 LINQ Pad 撰寫及測試 LINQ



```
void Main()
{
    var predicate = …r.False<BOM_ModelName>();
    predicate = pre… salesID.Contains("F0807463"));
    var query = BOM… e(predicate);

    var data = BOM_ModelName.Where(predicate);

    foreach(var c in data)
    {
        Console.WriteLine(c.modelName + c.salesID);
    }
}

// Define other methods and classes here
```

等同於開發 Console Application

```
IX1000,IX3620F0807463
BTS 3600AF0807463
Storage / IX3000F0807463
AFB power cabinetF0807463
BAIDU100F0807463
BasestaionF0807463
BasestationF0807463
Basestation / BST36000ATF0807463
Basestation / BST36000EF0807463
Basestation / BST36000TF0807463
Basestation / BST3600AF0807463
Basestation / BST3600AEF0807463
Basestation / BTS3606AEF0807463
Basestation/BST3600AEF0807463
BasetationF0807463
Basetation / MT45F0807463
Basetation /MT45F0807463
BS cabinetF0807463
BS Controler / ME60 F0807463
BSC 6800F0807463
```

Query successful (00:00.052)