

教材編號：U9544a

微軟 MCTS/MCPD 認證系列  
ASP.NET 3.5 網站開發進階  
(Advanced Visual Studio 2008: ASP.NET 3.5)



恒逸資訊教育訓練中心  
台北市復興北路 99 號 16F / 14 F / 12F  
TEL:02-25149191 FAX:02-25149292  
新竹市光復路二段 295 號 3 樓  
TEL:03-5723322 FAX:03-5726566  
台中市中港路一段 201 號 2 樓  
TEL:04-23297722 FAX:04-23102000  
高雄市中山二路 2 號 25 樓  
TEL:07-5361199 FAX:07-5366161  
<http://edu.uuu.com.tw>

【版權所有，未經恒逸書面授權請勿翻印或轉載】

微軟 MCTS/MCPD 認證系列  
**ASP.NET 3.5 網站開發進階**  
Advanced Visual Studio 2008: ASP.NET 3.5

---

**SYSTEX**  
making it happen 精誠資訊

**UCOM** 恒逸資訊  
Information Technology Education Center

作者／許薰尹、趙敏翔  
初版日期／2009/1/1

地址／台北市復興北路 99 號 14F  
電話／(02)25149191  
傳真／(02)25149292  
網址／<http://edu.uuu.com.tw>  
電子郵件／[service@edu.uuu.com.tw](mailto:service@edu.uuu.com.tw)

版權所有，未經恒逸書面授權請勿翻印或轉載

---

# ASP.NET 3.5 網站開發進階 (Advanced Visual Studio 2008: ASP.NET 3.5)

微軟 MCTS/MCPD 認證系列

教材編號：U9544a

# 教材大綱

## 第一章: 運用 ADO.NET 建置網頁

ADO.NET 概觀.....	4
ADO.NET 物件.....	5
連線字串.....	6
Web.config 中的 <connectionStrings> 區段.....	8
讀取設定檔中的連線字串.....	10
使用 Command 物件.....	12
練習 1.1 : 使用 DataReader 讀取資料.....	14
DataAdapter 物件模型.....	18
DataSet.....	20
使用 DataView 物件.....	24
練習 1.2 : 使用 DataView 進行資料篩選.....	27
ASP.NET 資料存取模型.....	31
使用 SqlDataSource 存取關聯式資料庫.....	33
自訂資料來源控制項參數.....	35
練習 1.3 : 自訂資料來源控制項參數.....	38
使用 ListView 控制項.....	46
ListView 控制項常用事件.....	48
以程式動態設定樣式.....	49
練習 1.4 : 以程式動態設定展示樣式.....	51

## 第二章: 資料存取進階設計

設計資料元件.....	6
透過類別提供的方法維護資料.....	7
獨立的類別庫.....	9
練習 2.1 : 設計資料元件.....	10
在資料元件的 SelectMethod 裡加入參數.....	17
ObjectDataSource 設定參數.....	18
練習 2.2 : 設定資料元件的查詢參數.....	19
排序參數設定.....	23
練習 2.3 : 設定 ObjectDataSource 排序.....	24
ObjectDataSource 編輯資料.....	28
DataObjectMethod 屬性標籤.....	28
練習 2.4 : 設計 ObjectDataSource 的更新功能.....	31
何謂 LINQ.....	40
LINQ 專案類型.....	41
LINQ To SQL Classes.....	42
ASP.NET 中的 LinqDataSource.....	45
練習 2.5 : 設定 LINQ to SQL 連接到資料庫.....	47

## 第三章: 網頁多國語言程式設計

全球化與當地語系化.....	4
語言喜好設定.....	5
取得用戶端選用語系.....	6
設定語系.....	7

練習 3.1 : 判斷用戶端語言喜好設定.....	9
自動化取得用戶端選用語系.....	13
練習 3.2 : 用戶端設定語言喜好設定.....	14
網頁當地語系化(Localization).....	17
建立 Local 資源檔.....	20
當地語系化運算式.....	21
練習 3.3 : 使用資源檔.....	23
使用程式取回 Local 資源.....	26
建立 Global 資源檔.....	27
使用程式存取 Global 資源檔.....	29
練習 3.4 : 使用 Global 資源檔.....	31
明確指定資源檔繫結屬性.....	33
使用 Localize 控制項 .....	34
練習 3.5 : 明確資源檔繫結與 Localize 控制項 .....	36
總結 .....	39
<b>第四章: 狀態管理機制</b>	
什麼是狀態維護 .....	4
狀態維護的類型 .....	6
狀態維護的運作 .....	7
HttpApplicationState 物件 .....	8
練習 4.1 : 使用 Application 物件 .....	10
Session 物件.....	15
使用 Session 物件 .....	17
練習 4.2 : 使用 Session 物件 .....	19
Webfarm 架構 下的狀態維護.....	24
使用 Out-of-Process Session .....	26
練習 4.3 : 使用 StateServer 儲存 Session .....	27
儲存 Session 到 SQL Server .....	29
練習 4.4 : 使用 SQL Server 儲存 Session .....	30
ViewState 物件 .....	36
使用 Cookie .....	37
HttpCookie 物件的使用 .....	38
練習 4.5 : 使用 Cookie 紀錄使用者工作階段 .....	40
<b>第五章: AJAX 進階設計</b>	
ASP.NET AJAX 架構 .....	4
Microsoft Ajax 程式庫 .....	6
簡易的捷徑語法 .....	8
網頁生命周期與事件 .....	9
註冊 Application 物件用戶端事件處理常式.....	11
使用 PageRequestManager 控管部分更新 .....	13
停止或取消非同步 Postback .....	15
練習 5.1 : 防止使用者重複按按鈕 .....	16
在部分更新過程更新 UpdatePanel 外的資料 .....	19
練習 5.2 : 使用 RegisterDataItem 傳遞自訂資料 .....	21
AJAX 除錯設定 .....	24
使用 Sys.Debug 類別除錯 .....	26
練習 5.3 : 使用 TraceDump 印出除錯資訊 .....	28

使用及管理指令碼 (Script) .....	31
將 JavaScript 檔案內嵌在資源組件 .....	33
發行版與除錯版指令碼.....	34
練習 5.4 : 部署內嵌在資源組件的 AJAX 指令碼 .....	37
指令碼全球化設計.....	42
指令碼當地語系化.....	44
練習 5.5 : 指令碼當地語系化.....	45
動態建立指令碼 .....	49
動態建立指令碼範例 .....	50
組合指令碼 .....	52
<b>第六章: 網站會員機制規劃</b>	
ASP.NET 網站會員機制.....	4
認識會員管理服務.....	5
Membership Provider .....	6
Membership API 之 Membership.....	7
Membership API 之 Membership.....	9
練習 6.1 : 使用 Membership API 建立使用者.....	10
使用 AJAX 整合 Membership.....	14
練習 6.2 : Ajax Membership API 的整合運用 .....	15
認識角色管理系統.....	20
Role Provider.....	21
Role API .....	23
練習 6.3 : 使用 Role API 管理角色.....	24
認識使用者設定檔.....	29
Profile Provider.....	30
建立 Profile 屬性 .....	31
練習 6.4 : 存取 Profile 的個人資料 .....	33
Profile 整合 Theme .....	37
練習 6.5 : 使用 Profile 搭配 Theme 佈景主題 .....	38
<b>第七章: 使用者控制項設計</b>	
何謂 Web User Control.....	4
如何設計 Web User Control.....	5
使用 User Control .....	7
練習 7.1 : 設計與使用 User Control .....	8
設計 User Control 自訂屬性.....	12
使用 User Control 自訂屬性.....	14
練習 7.2 : 存取 User Control 的自訂屬性.....	15
動態載入 User Control .....	18
練習 7.3 : 網頁中動態載入 User Control .....	19
User Control 的狀態維護 .....	21
練習 7.4 : 設計維護 User Control 狀態.....	23
<b>第八章: 自訂控制項進階設計</b>	
什麼是 Web Custom Control.....	4
Web Custom Control 的特色.....	5
設計 Web Custom Control 的步驟.....	7
練習 8.1 : 設計一個 Web Custom Control.....	9
在開發工具加入 Web Custom Control .....	12

網頁使用 Web Custom Control .....	14
練習 8.2 : 在網站上使用 Web Custom Control.....	16
自訂控制項的狀態維護 .....	20
狀態維護設計 .....	22
練習 8.3 : 設計維護自訂控制項狀態 .....	23
設計 Custom Composite Control .....	29
練習 8.4 : 設計 Custom Composite Control .....	31
支援樣版的 Custom Control .....	36
如何設計支援樣版的 Custom Control.....	37
網頁中繫結控制項樣板的資料 .....	39
練習 8.5 : 設計支援樣板的自訂控制項 .....	40

## 第九章: 效能調校與網站監控

ASP.NET 網頁快取架構.....	4
網頁輸出快取.....	5
建立 OutputCache 組態檔.....	7
使用程式操作快取.....	9
練習 9.1 : 使用網頁輸出快取 .....	11
網頁區塊快取.....	14
更新快取網頁的部分內容 .....	15
利用 API 或程式取代網頁快取內容 .....	17
資料快取.....	19
練習 9.2 : 使用資料快取.....	21
Cache 逾期與相依性設定 .....	24
使用 SQL Server 快取相依性 .....	26
練習 9.3 : 使用 SqlCacheDependency .....	28
使用健康監視 (Health Monitoring) .....	31
健康監視運作方式.....	32
Web 事件與提供者 .....	33
啓用健康監視功能.....	35
練習 9.4 : 使用健康監視功能監控網站活動 .....	37

## 第十章: HTTP 執行時期

ASP.NET Request 處理過程 .....	4
HttpApplication 事件.....	6
Global.asax .....	9
HTTP Module 運作模式.....	11
HTTP Module 與 Global.asax 比較 .....	12
設計 HTTP Module .....	13
練習 10.1 : 設計 HTTP Module.....	15
註冊 HTTP Module .....	19
練習 10.2 : 註冊及測試 HTTP Module.....	20
何謂 HTTP Handler.....	22
設計 HTTP Handler.....	23
練習 10.3 : 設計 BMPHandler .....	25
註冊 HTTP Handler.....	29
練習 10.4 : 註冊及測試 HTTP Module .....	32

## 第十一章: Web Service 設計

什麼是 Web Service .....	4
為什麼使用 Web Service .....	5
找尋並使用 Web Service 運作方式 .....	6
如何建立 Web Service .....	7
Web Service 程式架構 .....	9
練習 11.1 : 設計一個 Web Service .....	11
存取 Web Service .....	14
Proxy 的概念 .....	16
Proxy 存取 Web Service .....	17
練習 11.2 : 透過 Proxy 呼叫 Web Service .....	19
WCF 用戶端 .....	23
練習 11.3 : 使用 WCF 叫用 Web Service .....	24

## 關於本課程...

本課程為期四天，主要介紹使用 Visual Studio 2008 開發 ASP.NET 3.5 程式的進階設計，從網頁存取資料庫設計、AJAX 設計、多國語言的設計到網頁效能調較以及分散式的 Web Service 等等。以介紹 ASP.NET 3.5 許多新功能的進階設計，讓開發程式設計者能夠更了解 .NET 網頁程式設計的觀念以及應用，並可將此課程所學習的內容實作應用在專案開發和認證考試上。此為 MCTS、MCPD 必修課程，以協助取得認證。

## 對象

欲了解及如何應用最新 ASP.NET 3.5 技術架構、程式撰寫技巧與祕笈的 Web 程式開發人員，以及欲取得 MCTS、MCPD 認證者。

## 背景知識

已完成以下課程所具備技術能力：

- U9995 : Visual Basic 2008 程式語言概論
- U9996 : Visual C# 2008 程式語言概論
- U9543VB : ASP.NET 3.5 網站開發基礎(Visual Basic 2008)
- U9543C# : ASP.NET 3.5 網站開發基礎(Visual C# 2008)

## 參考課程

完成本課程之後，可參考以下相關課程：

- U2956VB : .NET 核心程式設計(Visual Basic 2008 & Visual Basic 2005)
- U2956C# : .NET 核心程式設計(Visual C# 2008 & C# 2005)
- U2546VB : .NET Windows 程式設計(Visual Basic 2005)
- U2546C# : .NET Windows 程式設計(Visual C# 2005)
- U70529VB : .NET 分散式應用程式開發(Visual Basic 2005)
- U70529C# : .NET 分散式應用程式開發(Visual C# 2005)

## 課程目標

上完本課程您將具備：

- 學會 ASP.NET3.5 相關進階技術，開發功能更強大的 Web 應用程式
- 深入了解 ASP.NET 技術架構
- 提升執行效能及穩定度的 ASP.NET 技術平台
- 協助通過 ASP.NET 3.5 相關認證

# 第一章: 運用 ADO.NET 建置網頁

## 本章大綱

ADO.NET 概觀.....	4
ADO.NET 物件.....	5
連線字串.....	6
Web.config 中的 <connectionStrings> 區段.....	8
讀取設定檔中的連線字串.....	9
使用 Command 物件.....	11
練習 1.1 : 使用 DataReader 讀取資料.....	13
DataAdapter 物件模型.....	17
DataSet.....	19
使用 DataView 物件.....	23
練習 1.2 : 使用 DataView 進行資料篩選.....	26
ASP.NET 資料存取模型.....	30
使用 SqlDataSource 存取關聯式資料庫.....	32
自訂資料來源控制項參數.....	34
練習 1.3 : 自訂資料來源控制項參數.....	37
使用 ListView 控制項.....	45
ListView 控制項常用事件.....	47
以程式動態設定樣式.....	48
練習 1.4 : 以程式動態設定展示樣式.....	50

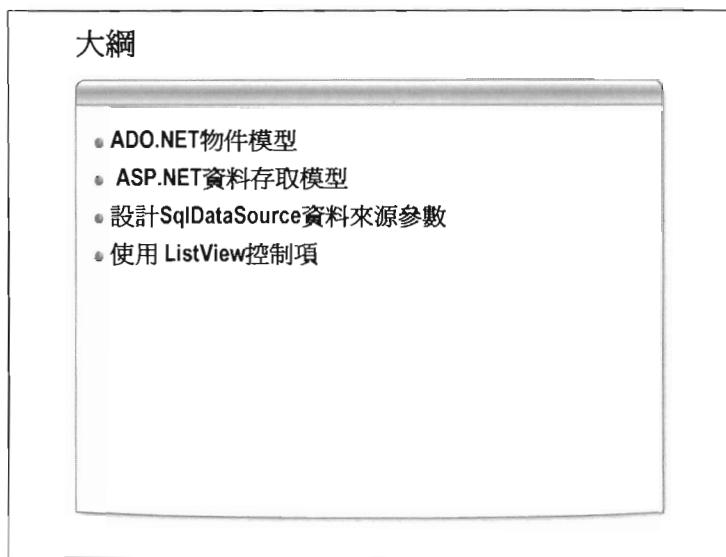
作者：

許薰尹





---

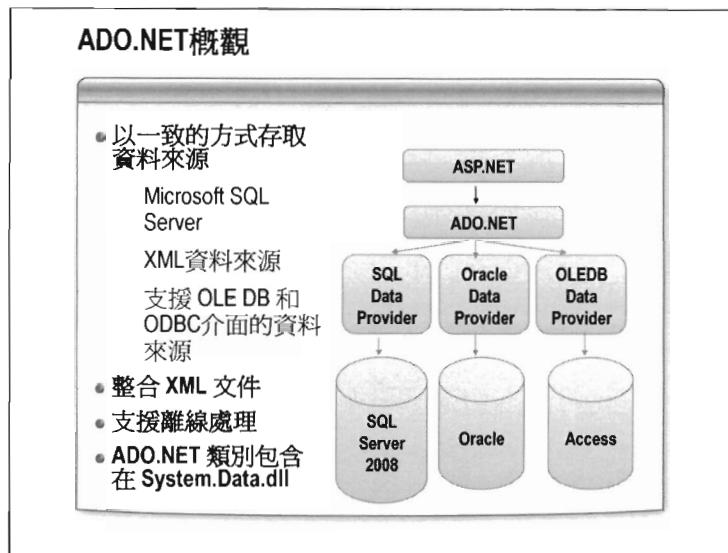


---

本章將介紹什麼是 ADO.NET，以及如何使用 ADO.NET 的基本物件存取資料來源，以建立連線。利用 Command 物件下達 SQL 指令或執行存程序，透過 DataSet 與 DataTable 暫存資料；使用 DataView 物件篩選資料。此外，也將進一步探討 ASP.NET 與 ADO.NET 整合的進階議題。

#### 在這一章中將學習到

- ADO.NET 物件模型
- ASP.NET 資料存取模型
- 設計 SqlDataSource 資料來源參數
- 使用 ListView 控制項

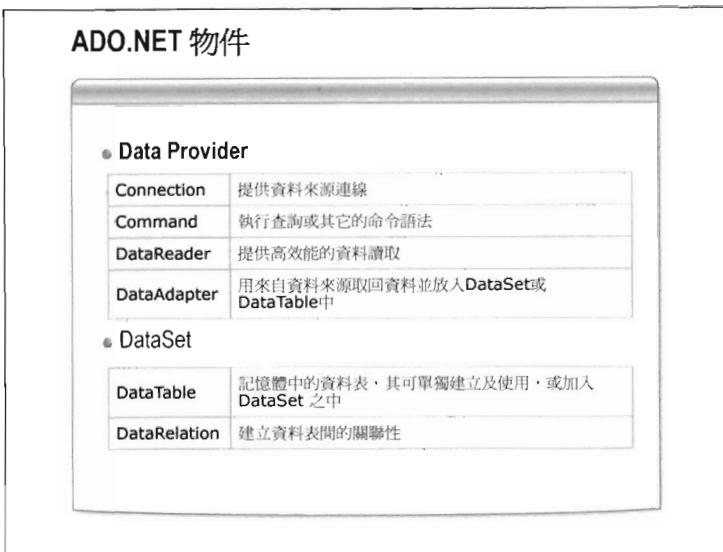


## ADO.NET 概觀

ADO.NET 可以讓程式設計師以一致的方式存取資料來源，如存取 Microsoft SQL Server、XML 資料來源，以及支援 OLE DB 和 ODBC 介面的資料來源。應用程式可利用 ADO.NET 連接至這些資料來源並讀取其中所含的資料，進行更新的作業。

透過 .NET Framework 資料提供者，ADO.NET 可連接到資料庫、執行新增、刪除、修改命令，或者擷取查詢結果，並將資料放入記憶體中暫存，支援離線處理機制。也可以透過 ADO.NET 的物件操作 XML 文件。ADO.NET 類別包含在 System.Data.dll 中，同時與 System.Xml.dll 中的 XML 類別緊密整合。

在 .NET Framework 中，你可以利用 Microsoft® Visual Studio® 搭配 ADO.NET 設計更有效率的資料存取程式。也就是說實際上資料存取的動作，是透過 ADO.NET 進行的。利用 .NET Framework 所設計的 Web 應用程式可以先透過 ADO.NET 進行資料存取，而 ADO.NET 底層則透過各式各樣的 Provider 對資料儲存體 (Data Store) 進行真正的資料操作。



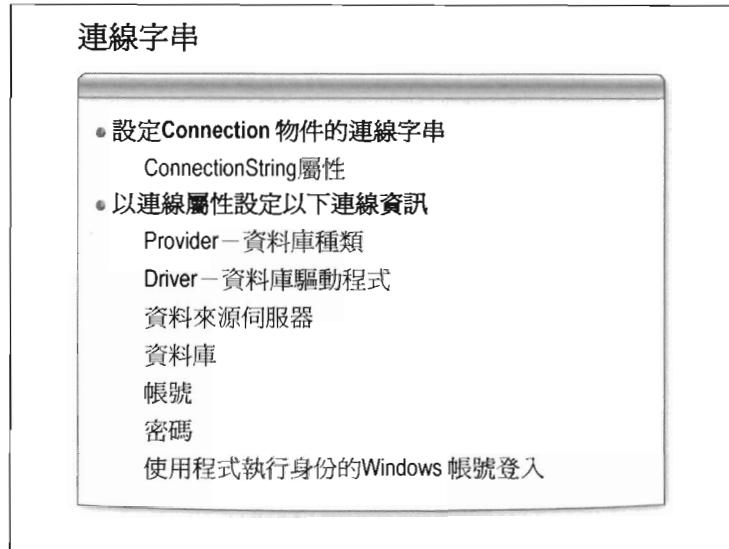
## ADO.NET 物件

在 ADO.NET 裡，Connection 物件用來連接資料來源。要連接時須先設定設定連線字串，以 SQL Server 2008 為例，可使用 SqlConnection 物件以如下程式碼建立連線，稍後會介紹連線字串中的參數：

```
Visual Basic
Dim cn As SqlConnection
cn = New SqlConnection("Data Source=localhost;User Id=sa;Pass
word=;Initial Catalog=northwind")
cn.Open()
```

```
C#
SqlConnection cn ;
cn = new SqlConnection("Data Source=localhost; User Id=sa;Pass
word=;Initial Catalog=northwind");
nwindConn.Open();
```

一旦資料庫連線使用完畢建議儘早關閉，以便讓資料庫連線資源可以歸還給系統給其他有需要的應用程式使用。只要呼叫 Connection 物件的 Close 方法就可以達到歸還資源的效果。



## 連線字串

由於 OleDbConnection 透過 OLE DB 介面可存取多種不同的資料庫，因此需要以 Provider 屬性指定要連接的是那一種資料庫。

若要連接至 Microsoft SQL Server 7.0 以上的版本，可使用.NET Framework Data Provider for SQL Server 提供者中的 SqlConnection 物件。若要連接至 OLE DB 資料來源或 Microsoft SQL Server 6.x 或更早版本，請使用 .NET Framework Data Provider for OLE DB 中的 OleDbConnection 物件。若要連接至 ODBC 資料來源，請使用 .NET Framework Data Provider for ODBC 中的 OdbcConnection 物件。若要連接至 Oracle，可使用 .NET Framework Data Provider for Oracle 中的 OracleConnection 物件。

使用這些資料提供者，可以參考以下連線字串：

- 透過 OleDbConnection 連接 Oracle 資料庫

```
"Provider=MSDAORA; Data Source=ORACLE8i7;Persist
Security Info=False;Integrated Security=yes"
```

- 透過 OleDbConnection 連接 Access 資料庫

```
"Provider=Microsoft.Jet.OLEDB.4.0; Data  
Source=c:\DB\Northwind.mdb"
```

- 透過 OleDbConnection 連接 SQL Server 資料庫

```
"Provider=SQLOLEDB;Data Source=MySQLServer;Integrated  
Security=SSPI"
```

## Driver

由於 OdbcConnection 透過 ODBC 介面可存取多種不同的資料庫，因此需要以 Driver 屬性指定使用哪一種資料庫驅動程式。如：

- 透過 OdbcConnection 連接 Oracle 資料庫

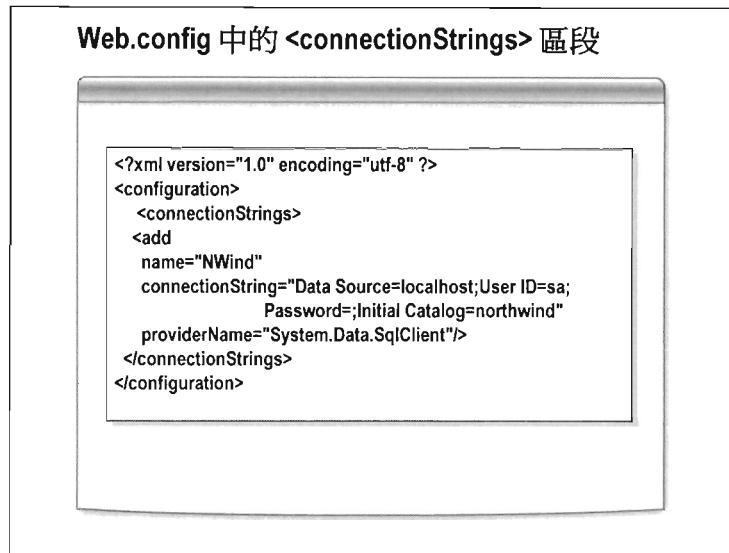
```
"Driver={Microsoft ODBC for Oracle};Server=ORACLE8i7;  
Persist Security Info=False;Trusted_Connection=yes"
```

- 透過 OdbcConnection 連接 Access 資料庫

```
"Driver={Microsoft Access Driver (*.mdb)};DBQ=c:\bin\nwind.  
mdb"
```

- 透過 OdbcConnection 連接 SQL Server 資料庫

```
"Driver={SQL Server};Server=MyServer;Trusted_Connection=  
yes;Database=Northwind;"
```



## Web.config 中的 <connectionStrings> 區段

在 Web 應用程式中，連線字串被視為是一項重要、機密的資訊。因此在.NET Framework 的 Web.config 檔中加入一個新區段：

<connectionStrings>，專門用來存放 Web 應用程式中使用到的連線字串。如下所示：

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <connectionStrings>
    <add
      name="NWind"
      connectionString="Data Source=localhost;User ID=sa;Passwo
rd=;Initial Catalog=northwind"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
</configuration>
```

<connectionStrings>裡的<add> 元素包含以下三個重要屬性：

- name—此連線資訊的名稱。
- connectionString—實際的連線字串。
- providerName—用來存取此連線資料來源的.NET Data Provider。

### 讀取設定檔中的連線字串

- System.Web.Configuration 命名空間
- WebConfigurationManager 類別

Visual Basic

```
Dim cnSection As ConnectionStringsSection =
WebConfigurationManager.GetWebApplicationSection("connectionStrings")
Dim NWind As String =
cnSection.ConnectionStrings("NWInd").ConnectionString
```

C#

```
ConnectionStringsSection cnSection = (ConnectionStringsSection)
WebConfigurationManager.GetWebApplicationSection("connectionStrings");
string NWind = cnSection.ConnectionStrings["NWInd"].ConnectionString;
```

*<Add>*  
*Name*

### 讀取設定檔中的連線字串

.NET Framework 也增加相關功能來讀取設定檔中的連線字串。

在.NET Framework 的 System.Web.Configuration 命名空間下有一個新類別：WebConfigurationManager，可用來存取 Web.config 設定檔的所有內容。

Web.config 檔中每個區段都是繼承自 ConfigurationSection 類別，且每個區段對應的類別名稱就是該區段名稱再加上「Section」。舉例而言：ConnectionStringsSection 類別就對應到 Web.config 設定檔中的 <connectionStrings> 區段。

Visual Basic

```
<%@ Import Namespace ="System.Web.Configuration" %>
...
Dim cnSection As ConnectionStringsSection = WebConfigurationM
anager.GetWebApplicationSection("connectionStrings")
```

C#

```
<%@ Import Namespace ="System.Web.Configuration" %>
...
ConnectionStringsSection cnSection = (ConnectionStringsSection)
WebConfigurationManager.GetWebApplicationSection("connection
Strings");
```

以上的程式會將傳回物件明確地轉型為 `ConnectionStringsSection` 型別。接著再以如下程式取出名為「`NWind`」中的連線字串：

Visual Basic

```
Dim NWind As String = cnSection.ConnectionStrings("NWind").ConnectionString
```

C#

```
string NWind = cnSection.ConnectionStrings["NWind"].ConnectionString
```

## 使用 Command 物件

- **建立Command物件**  
Command類別的建構子  
Connection物件的CreateCommand方法
- **CommandText屬性**  
指令內容
- **CommandType屬性**  
指令類型
- **執行指令**  
ExecuteReader方法  
ExecuteScalar方法  
ExecuteNonQuery方法

## 使用 Command 物件

連線建立之後便可透過 Command 物件修改及查詢資料。建立 Command 物件的方式有二種，一是透過 Command 本身的建構函式，或是藉由 Connection 物件的 CreateCommand 方法。

常用屬性：

- Connection：指定透過那一個資料庫連線物件，執行資料存取動作。
- CommandText：資料來源的存取指令可在 Command 物件的 CommandText 屬性中指定，可能是 SQL 指令或預存程序的名稱。
- CommandType：若要執行預存程序(Stored Procedure)，需用 Command 物件的 CommandType 屬性設定為 StoredProcedure。Parameters 屬性可用來設定 SQL 指令中的參數，或是預存程序中的輸出/入參數及傳回值。不過要注意，呼叫 ExecuteReader 時，傳回值及輸出參數需在關閉相關的 DataReader 後才能存取。

## 執行指令

Command 提供幾個執行指令的方法：

- 若要取回一組資料可呼叫 ExecuteReader 方法傳回一個 DataReader 物件。
- 呼叫 ExecuteScalar 可傳回一個單值。
- 呼叫 ExecuteNonQuery 則是用來執行指令不回傳查詢資料。它會傳回一個整數代表受影響的資料筆數。

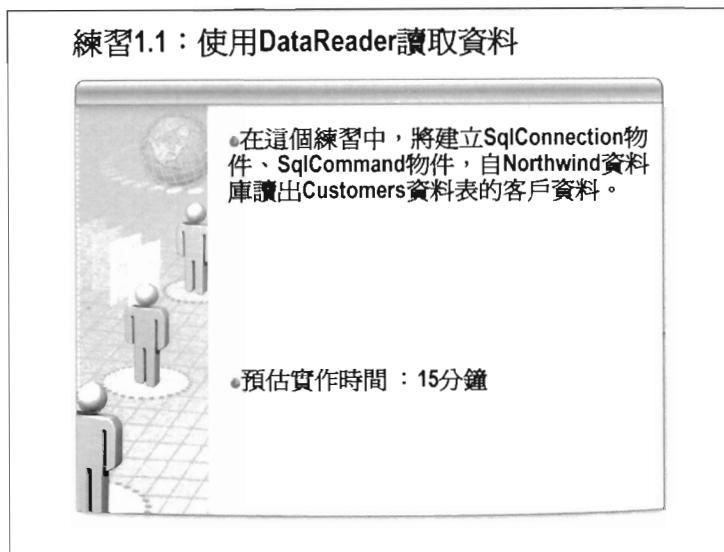
以下程式傳回 Northwind 資料庫的 Categories 資料表內容：

Visual Basic

```
Dim cmd As SqlCommand = New SqlCommand("SELECT CategoryID, CategoryName FROM Categories", cn)
```

C#

```
SqlCommand cmd = new SqlCommand("SELECT CategoryID, CategoryName FROM Categories", cn);
```



## 練習 1.1：使用 DataReader 讀取資料

### 目的：

在這個練習中，將建立 SqlConnection 物件、SqlCommand 物件，自 Northwind 資料庫讀出 Customers 資料表的客戶資料。

### 功能描述：

在練習中，將於「Mod01\_1」網站的網頁，透過程式取回並顯示 Northwind 資料庫中的 Customers 資料。

### 預估實作時間：15 分鐘

### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod01\_1\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod01\_1」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 切換到網頁的「Source」畫面。在 ASPX 網頁最上方匯入以下命名空間：

```
<%@ import Namespace="System.Data" %>
<%@ import Namespace="System.Data.SqlClient" %>
```

5. 在 Page\_Load 事件處理常式，加入以下程式建立 SqlConnection 物件：

<b>Visual Basic</b> <pre>Dim cn As New SqlConnection("Server=.; DataBase=Northwind; Integrated Security=sspi")</pre>
---

<b>C#</b> <pre>SqlConnection cn = new SqlConnection("Server=.; DataBase=Northwind; Integrated Security=sspi");</pre>
---

6. 加入以下程式建立並設定 SqlCommand 物件：

<b>Visual Basic</b> <pre>Dim cmd As New SqlCommand cmd.Connection = cn cmd.CommandText = "select CustomerID, CompanyName from Customers;SELECT FirstName, LastName FROM Employees;"</pre>
--

<b>C#</b> <pre>SqlCommand cmd = new SqlCommand(); cmd.Connection = cn; cmd.CommandText = "select CustomerID, CompanyName from Customers;SELECT FirstName, LastName FROM Employees;"</pre>
--

7. 加入以下程式開啟連線，並呼叫 ExecuteReader 方法將傳回的 Customer 資料表資料逐筆顯示在網頁上；然後呼叫 DataReader 的 NextResult 方法移到查詢的第二個結果集，再將傳回的 Employees 資料表資料逐筆顯示在網頁上：

<b>Visual Basic</b> <pre>cn.Open()</pre>
---

```

Dim dr As SqlDataReader = cmd.ExecuteReader()
If dr.HasRows Then
    Do While dr.Read()
        Response.Write(dr.GetString(0) & "-" & dr.GetString(1) & "<Br>")
    Loop
Else
    Response.Write("沒有資料.")
End If
Response.Write("<hr/>")
dr.NextResult()
If dr.HasRows Then
    Do While dr.Read()
        Response.Write(dr.GetString(0) & "-" & dr.GetString(1) & "<Br>")
    Loop
Else
    Response.Write("沒有資料.")
End If
dr.Close()
cn.Close()

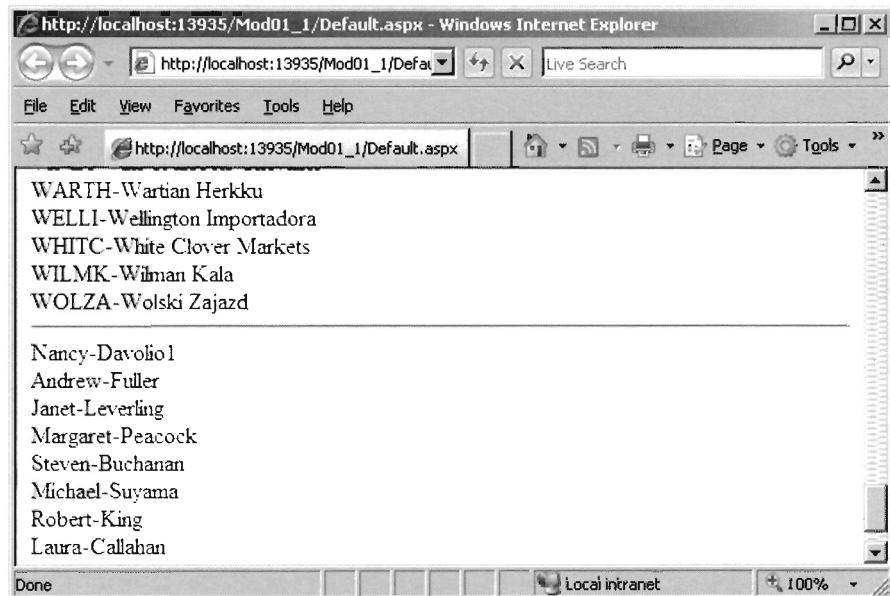
```

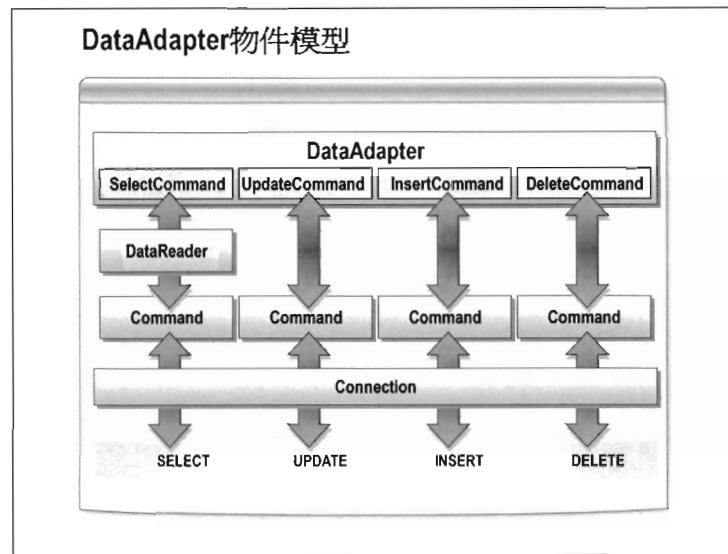
```

C#
cn.Open();
SqlDataReader dr = cmd.ExecuteReader();
if (dr.HasRows)
    while (dr.Read())
        Response.Write(dr.GetString(0) + "-" + dr.GetString(1) + "<Br>");
else
    Response.Write("沒有資料.");
Response.Write("<hr/>");
dr.NextResult();
if (dr.HasRows)
    while (dr.Read())
        Response.Write(dr.GetString(0) + "-" + dr.GetString(1) + "<Br>");
else
    Response.Write("沒有資料.");
dr.Close();
cn.Close();

```

8. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」，確認程式執行結果。參考下圖所示：





## DataAdapter 物件模型

每個.NET Framework 的資料提供者套件中都包含一個 DataAdapter 類別：

- .NET Framework Data Provider for OLE DB：  
OleDbDataAdapter
- .NET Framework Data Provider for SQL Server：  
SqlDataAdapter
- .NET Framework Data Provider for ODBC：OdbcDataAdapter
- .NET Framework Data Provider for Oracle：  
OracleDataAdapter

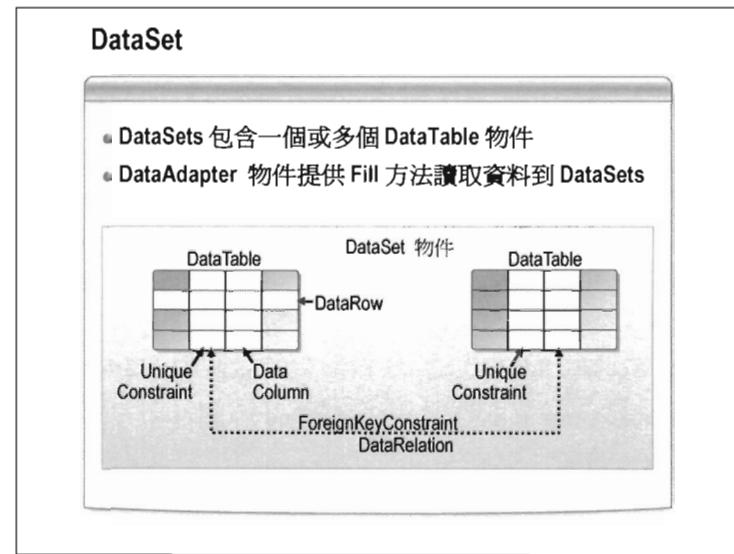
DataAdapter 用來自資料來源收回資料並放入 DataSet 的 DataTable 中；同時，DataAdapter 也負責將 DataSet 更新過的內容寫回資料來源。DataAdapter 也是用 Connection 物件連接資料來源、Command 物件執行資料存取指令。

DataAdapter 裡的 **SelectCommand** 屬性其實是個 **Command** 物件，用來自資料來源取得資料。**InsertCommand** 屬性、**UpdateCommand** 屬性及

DeleteCommand 屬性也都是 Command 物件用來將 DataSet 編輯過的資料新增、更新至資料來源或自資料來源中刪除資料。

如果透過 OleDbConnection 連接到資料庫，必須使用 OleDbDataAdapter 或 OleDbCommand；若使用 SqlConnection 則必須使用 SqlDataAdapter 或 SqlCommand，依此類推。

若只是單純執行 SQL 指令，建議最好使用 Command 物件效能較佳，因為 DataAdapter 包含許多關於處理 DataSet 的功能，反而使執行效能變差。



## DataSet

ADO.NET 的 DataSet 是個存在於記憶體中的資料儲存物件，你可以將任何資料來源取回的資料，放到 DataSet 之中，DataSet 提供一致的關聯式程式設計模型，讓你存取其中的資料。

DataSet 的使用方式有幾種，：

- 以程式在 DataSet 內建立 DataTable、DataRelation 和 Constraint，並填入資料表的資料。
- 使用 DataAdapter 物件，將關聯式資料庫中的資料表填入 DataSet。
- 使用 XML 載入資料到 DataSet。

因為 DataSet 獨立於資料來源之外，因此不見得要與資料來源的架構及內容完全一樣，可以放應用程式所需的資料即可，且每個資料表中的資料可能是來自不同資料來源的。

## 使用 DataSet

編輯過的 DataSet 內容若要與資料來源進行同步，可透過 DataAdapter 物件。你可以利用程式建立 DataSet 物件，DataSet 之內的資料若有更新可以透過 DataAdapter 的 Update 方法將資料更新回資料庫。

DataAdapter 包含 SelectCommand、InsertCommand、UpdateCommand、DeleteCommand 等屬性，你需分別指定 Select、Insert、Update、Delete 等 SQL 指令，透過 Fill 方法 (Method) 將資料讀到 DataSet，或 DataTable。當呼叫 Update 方法 (Method) 會依據 DataSet 或 DataTable 中資料變更的狀況執行 InsertCommand、UpdateCommand、DeleteCommand 等屬性所定義的 SQL 指令更新資料到資料庫。

在 DataSet 中可包含 DataTable(s)，DataTable 可包含 DataColumn(s) 與 DataRow(s)，DataSet 也可直接的建立關聯性，依關聯性來檢視資料。

無論資料自何種資料來源取出，皆可存放至 DataSet 裡。資料放入之前需先建立 DataSet 物件。

您可以呼叫 DataSet 建構函式來建立 DataSet 物件。

```
Visual Basic  
Dim ds As DataSet
```

```
C#  
DataSet ds = new DataSet();
```

建立 DataSet 時，可以選擇性地指定 DataSet 的名稱。如果不指定 DataSet 的名稱，則名稱會設定為「NewDataSet」。

```
Visual Basic  
Dim ds As DataSet = New DataSet("PubsDataSet")
```

```
C#  
DataSet ds = new DataSet("PubsDataSet");
```

## 使用 DataAdapter 將資料填入 DataTable

DataSet 物件可以做為離線的資料來源，DataSet 物件之中主要存放資料的物件，稱之為 DataTable 物件。我們可以使用 DataAdapter 建立 DataSet 物件。使用 DataAdapter 物件可以在建立 DataSet 物件中的同時，建立一個 DataTable。

您可以利用 DataAdapter 的 Fill 方法可用來將 SelectCommand 取回的資料填入 DataSet 的 DataTable 中。呼叫 Fill 方法時需指定 DataTable 物件，或欲新增的 DataTable 名稱。

以下的程式會建立一個 SqlConnection 連結到 Microsoft SQL Server 的 Northwind 資料庫，並將客戶資料取回放入 DataSet 中名為「Customers」的 DataTable 中：

```
Visual Basic
Dim cn As SqlConnection = New SqlConnection("Data Source=localhost;Integrated Security=SSPI;Initial Catalog=northwind")

Dim cmd As SqlCommand = New SqlCommand("SELECT CustomerID, CompanyName FROM Customers", cn)
cmd.CommandTimeout = 30

Dim dap As SqlDataAdapter = New SqlDataAdapter()
dap.SelectCommand = cmd

cn.Open()

Dim ds As DataSet = New DataSet()
dap.Fill(ds, "Customers")

cn.Close()
```

```
C#
SqlConnection cn = new SqlConnection("Data Source=localhost;Integrated Security=SSPI;Initial Catalog=northwind");

SqlCommand cmd = new SqlCommand("SELECT CustomerID, CompanyName FROM Customers", cn);
cmd.CommandTimeout = 30;

SqlDataAdapter dap = new SqlDataAdapter();
dap.SelectCommand = cmd;

cn.Open();

DataSet ds = new DataSet();
```

```
dap.Fill(ds, "Customers");
```

```
|
```

```
cn.Close();
```

### 使用 DataView 物件

- 以不同的檢視方式檢視 DataTable 物件的內容
- 提供功能
  - 設定資料排序方式、設定資料篩選條件、找尋資料
- 建立或取得 DataView 物件
  - 以 DataView 建構函式建立 DataView
  - DefaultView 屬性
- 使用 RowFilter 屬性指定篩選條件

### 使用 DataView 物件

DataView 物件可為一個 DataTable 物件的內容產生不同的檢視結果，讓你以不同的排序方式、篩選條件來檢視資料，或找尋資料。再透過資料繫結機制，展現在應用程式的控制項上。

儘管如此，DataView 的欄位及欄位數一定會與原始的 DataTable 相同，無法新增、移除或變更。

DataView 提供您 DataTable 的動態檢視能力，它和資料庫檢視表 (View) 相當地類似，可以套用不同的排序方式和篩選準則來瀏覽或選取資料。然而，和資料庫檢視表不一樣的地方是：你並無法將 DataView 當成 DataTable 來使用。

### 建立或取得 DataView 物件

有兩種建立或取得 DataView 的方法。一個是透過 DataView 的建構函式，一個是從 DataTable 的 DefaultView 屬性取得 DataView。

如下程式展示如何以 DataView 建構函式建立 DataView，它的原始 DataTable 為 CustomersTable：

**Visual Basic**

```
Dim custDV As DataView = New DataView(CustomersTable)
```

**C#**

```
DataView custDV = new DataView(CustomersTable);
```

如下程式展示如何自 CustomersTable 的 DefaultView 屬性取得 DataView：

**Visual Basic**

```
Dim custDV As DataView = CustomersTable.DefaultView
```

**C#**

```
DataView custDV = CustomersTable.DefaultView;
```

## 使用 DataView 指定篩選條件

DataView 的 RowFilter 屬性決定從 DataView 只會看到哪些資料。

RowFilter 屬性最適用於資料繫結應用程式，將篩選的資料透過繫結控制項顯示。

以下程式篩選出居處在 USA 國家的客戶資料：

**Visual Basic**

```
custDV.RowFilter = "Country = 'USA'"
```

**C#**

```
custDV.RowFilter = "Country = 'USA'" ;
```

若要過濾 FirstName 欄位為“mary”的記錄，可以使用：

**Visual Basic**

```
dv.RowFilter = "FirstName = 'mary'"
```

**C#**

```
dv.RowFilter = "FirstName = 'mary'" ;
```

過濾 FirstName 欄位為‘A’開頭的記錄內容：

```
Visual Basic  
dv.RowFilter ="FirstName Like 'A%' "
```

```
C#  
dv.RowFilter ="FirstName Like 'A%' ";
```

另外，如果要過濾多條件可以使用[and]或[or]運算子。

例如：

```
Visual Basic  
dv.RowFilter ="FirstName Like 'A%' and LastName='Lo'"
```

```
C#  
dv.RowFilter ="FirstName Like 'A%' and LastName='Lo"';
```



## 練習 1.2 : 使用 DataView 進行資料篩選

目的：

這個練習主要是了解如何使用 DataView 物件 RowFilter 屬性，設定篩選條件。

功能描述：

在練習中將使用 DataTable 的 DefaultView 屬性，取得預設關聯到此 DataTable 的 DataView 物件，然後設定以 City 欄位排序，篩選出 city 欄位值為某個字開頭的所有員工。

預估實作時間：12 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選

取「\U9544\Practices\VB 或 CS\Mod05\_1\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod01\_2」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 從「Toolbox」工具箱中拖拉一個 TextBox、Button、GridView 控制項到網頁中。設定 Button 的 Text 屬性為「Search」，畫面看起來如：

Column0	Column1	Column2
abc	abc	abc

目前標籤看起來如下：

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:Button ID="Button1" runat="server" Text="Search" />
<br />
<asp:GridView ID="GridView1" runat="server">
</asp:GridView>
```

5. 在 ASPX 網頁最上方匯入以下命名空間：

```
<%@ import Namespace="System.Data" %>
<%@ import Namespace="System.Data.SqlClient" %>
```

6. 加入 Page\_Load 事件處理程式，利用 SqlDataAdapter 查詢出 Customers 資料表資料，然後存放到 DataSet；再將 DataSet 的內容利用資料繫結技術呈現在 GridView 控制項上，末了將 DataSet 處存到 Session 之中：

Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As Sy
```

```

    stemEventArgs)
    If Not Page.IsPostBack Then
        Dim ds As New DataSet
        Dim da As New SqlDataAdapter("Select * from Customers",
", "Data Source=.;Initial Catalog=Northwind;integrated Security=
sspi;")
        da.Fill(ds)
        GridView1.DataSource = ds
        GridView1.DataBind()
        Session("ds") = ds
    End If

End Sub

```

```

C#
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        DataSet ds = new DataSet();
        SqlDataAdapter da = new SqlDataAdapter("Select * from Customers",
", "Data Source=.;Initial Catalog=Northwind;integrated Security=
sspi;");
        da.Fill(ds);
        GridView1.DataSource = ds;
        GridView1.DataBind();
        Session["ds"] = ds;
    }
}

```

7. 雙擊設計畫面中的 Search 按鈕，產生 Click 事件處理常式，加入以下程式，從 Session 中取出 DataSet 資料，取出 DataSet 中第一個 DataTable 的 DataView 物件，利用 DataView 篩選出 City 欄位值符合文字方塊輸入的內容開頭之資料，然後以 GridView 控制項呈現：

```

Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim ds As DataSet = TryCast(Session("ds"), DataSet)
    Dim dv As DataView = ds.Tables(0).DefaultView
    dv.RowFilter = "City LIKE '" + TextBox1.Text + "%'"
    GridView1.DataSource = dv
    GridView1.DataBind()
End Sub

```

```

C#
protected void Button1_Click(object sender, EventArgs e)
{
}

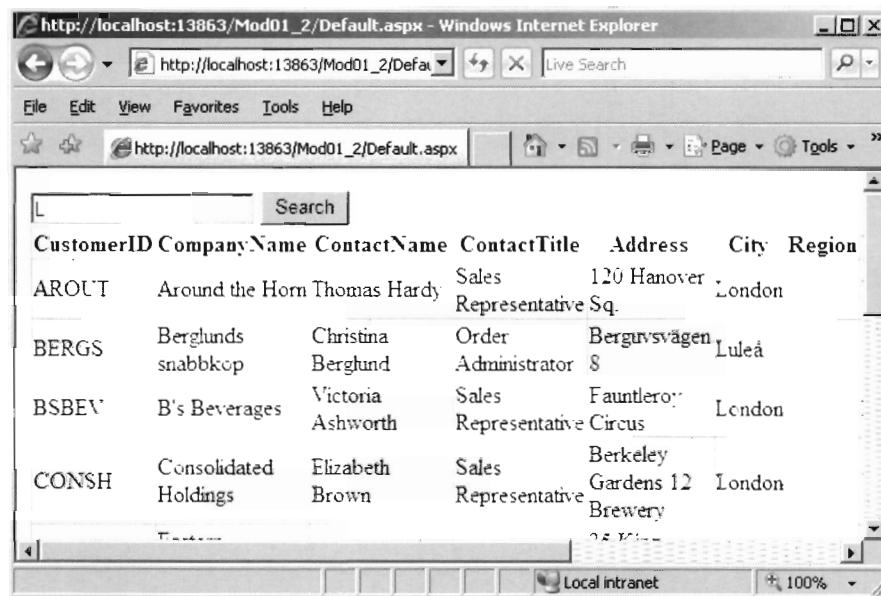
```

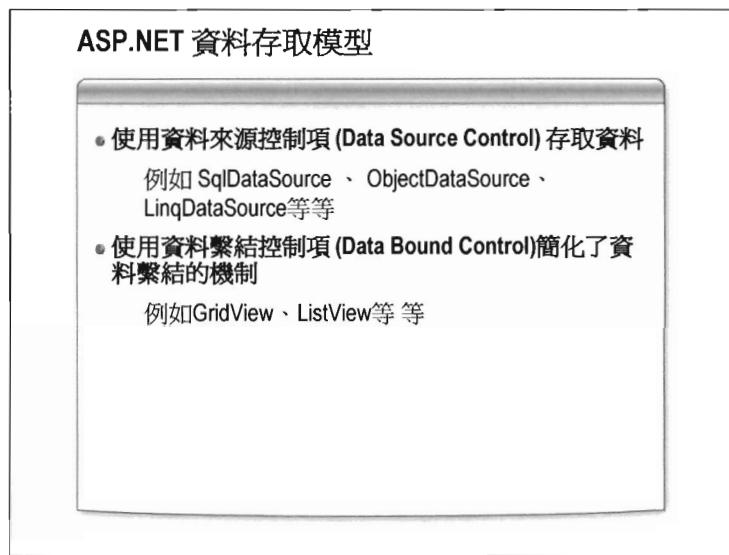
```

        DataSet ds = Session["ds"] as DataSet;
        DataView dv = ds.Tables[0].DefaultView;
        dv.RowFilter = "City LIKE '" + TextBox1.Text + "%'";
        GridView1.DataSource = dv;
        GridView1.DataBind();
    }
}

```

8. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。第一次執行時會顯示所有 Customers 資料表資料；當文字方塊輸入「L」，再按 Search 按鈕，則顯示所有 City 欄位值是以 L 開頭的資料：





## ASP.NET 資料存取模型

在網頁中顯示資料十分頻繁，為簡化網頁開發人員所需撰寫的程式碼。ASP.NET 的資料繫結架構提供許多資料來源控制項及資料繫結控制項可以更方便、更直覺地呈現資料。

### 資料來源控制項

使用資料來源控制項最顯而易見的好處是：宣告式(Declarative)資料繫結方式。這個方式可大幅減少 ASP.NET 網頁中所需撰寫的程式碼。並可讓程式設計人員在規範好的環境中存取、顯示資料，如此也可提升程式的效能及穩定性。

ASP.NET 提供的資料來源控制項有： SqlDataSource 、 ObjectDataSource... 等。

### 資料繫結控制項

無論資料來源控制項是那一種，ASP.NET 的資料繫結控制項都會將它所提供的資料標準化成一個可列舉(Enumerable)的集合。資料繫結控制項可分為三種：

- 標準控制項

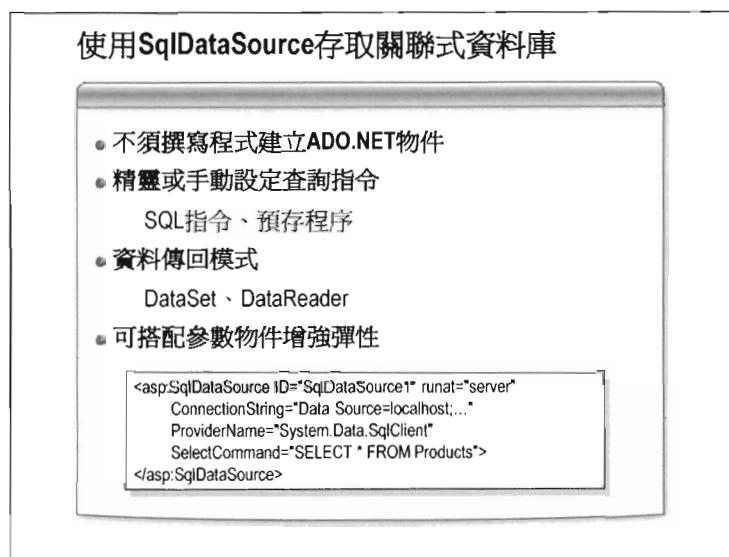
如：TextBox、Label、Button...等。

- 清單式控制項

如：CheckBoxList、DropDownList、ListBox...等。

- 複合式控制項

如：GridView、FormView...等。



## 使用 SqlDataSource 存取關聯式資料庫

程式設計師不需建立任何 ADO.NET 物件即可使用 SqlDataSource 存取關聯式資料庫。

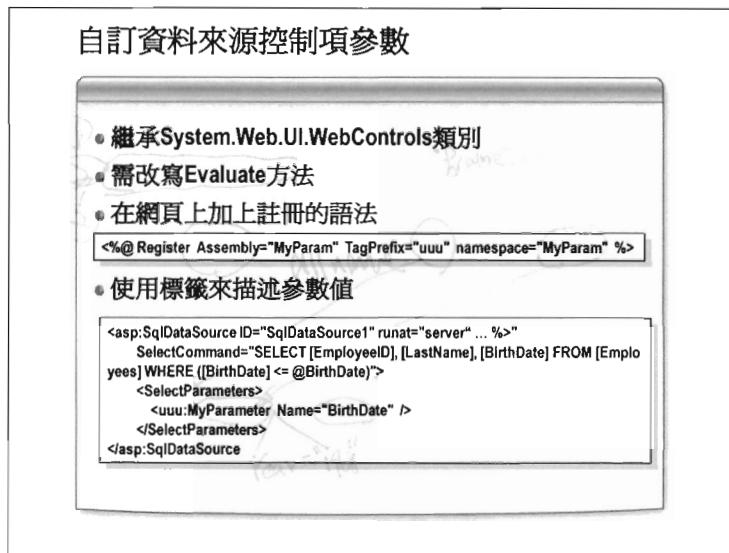
### 查詢指令

除可透過精靈自動建立查詢指令，也可自行修改或輸入 SQL 查詢指令。可視資料來源的種類使用特定的 SQL 語法及指令，像是 T-SQL 或 PL-SQL。另外也可在 SelectCommand 放入預存程序的名稱呼叫預存程序，此時 SelectCommandType 會被設為 StoredProcedured。

### 資料傳回模式

SqlDataSource 控制項的 DataSourceMode 屬性決定資料的回傳模式。預設為 DataSet，自資料來源傳回的資料會整批儲存在 ASP.NET 應用程式伺服器的記憶體，在這個模式下相關的資料展示控制項如：GridView 及 DetailsView 可以提供較豐富的資料展示功能像是排序及分頁。

另外也可將 `DataSourceMode` 屬性設定為 `DataReader`，這表示回傳資料不會整批儲存在 ASP.NET 應用程式伺服器的記憶體。



## 自訂資料來源控制項參數

ASP.NET 提供許多內建的資料來源控制項，能夠快速透過它們來存取資料。使用某些資料來源控制項時，如 SqlDataSource 或 ObjectDataSource，經常會利用參數來進行查詢，以便篩選出適當的資料。參數可以從控制項中輸入的值而來，稱之為 ControlParameter；此外，ASP.NET 還包含 CookieParameter、FormParameter、ProfileParameter、QueryStringParameter 與 SessionParameter。

若有需要，您可以自訂特殊的資料來源控制項參數(Parameter)，自行設定參數相關資訊。要建立自訂的資料來源控制項參數相當的簡單。

### 控制項參數

若要自訂參數控制項，可以繼承 System.Web.UI.WebControls 類別，這個類別提供所有參數控制項基本的功能。參數控制項至少要提供一個 Evaluate 方法，用來回傳參數的值，通常這個方法的實作內容相單的簡單。

Evaluate 包含兩個輸入參數：

- HttpContext：代表 HTTP 請求的相關資訊。

- 參數控制項關聯至資料來源控制項的參考

建立自訂參數時，有時會改寫 Parameter 類別的 Clone 方法，建立參數物件的副本，以便提供設計時期的支援。

以下是自訂 Parameter 的範例：

```

Visual Basic
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports System.Web

Imports System.Web.UI
Imports System.Web.UI.WebControls

Namespace MyParam
    Public Class MyParameter
        Inherits Parameter

        Protected Overrides Function Evaluate(context As HttpContext,
                                              control As Control) As Object
            Dim d As New DateTime(1955, 1, 1)
            Return d
        End Function 'Evaluate
    End Class 'MyParameter
End Namespace 'MyParam

```

```

C#
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

using System.Web.UI;
using System.Web.UI.WebControls;
namespace MyParam
{
    public class MyParameter : Parameter
    {
        protected override object Evaluate(HttpContext context, Control control)
        {
            DateTime d = new DateTime(1955, 1, 1);
            return d;
        }
    }
}

```

## 建立參數

MyParameter 需要支援利用宣告式的語法來設定其值，因此使用時，需要在網頁上加上註冊的語法：

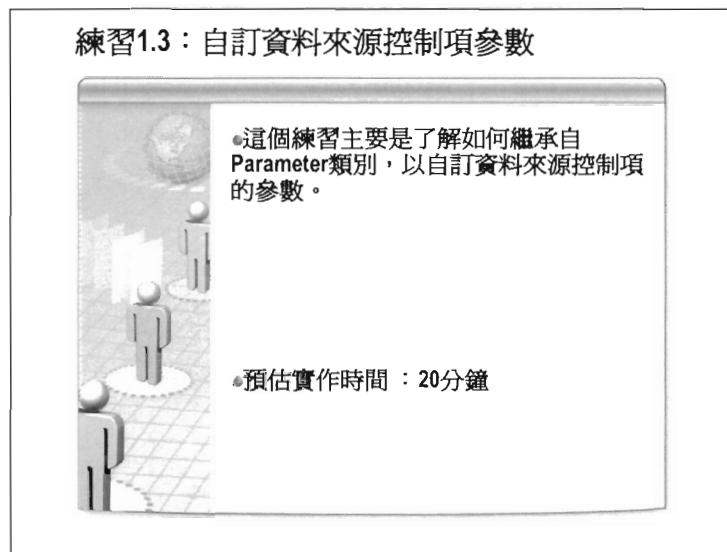
```
<%@ Register Assembly="MyParam" TagPrefix="uuu" namespace="MyParam" %>
```

這樣在 SqlDataSource 或 ObjectDataSources 的參數集合之中，就可以使用 MyParameter 標籤來描述參數值。以下是 GridView 控制項使用 MyParameter 的範例：

```
<asp:GridView ID="GridView1" runat="server"
    AutoGenerateColumns="False"
    DataKeyNames="EmployeeID" DataSourceID="SqlDataSource1"
    EmptyDataText="There are no data records to display.">
    <Columns>
        <asp:BoundField DataField="EmployeeID"
            HeaderText="EmployeeID"
            InsertVisible="False" ReadOnly="True"
            SortExpression="EmployeeID" />
        <asp:BoundField DataField="LastName"
            HeaderText="LastName"
            SortExpression="LastName" />
        <asp:BoundField DataField="BirthDate"
            HeaderText="BirthDate"
            SortExpression="BirthDate" />
    </Columns>
</asp:GridView>

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:NorthwindConnectionString %>" 
    ProviderName="<%$ ConnectionStrings:NorthwindConnectionString.ProviderName %>" 
    SelectCommand="SELECT [EmployeeID], [LastName], [Birth Date] FROM [Employees] WHERE ([BirthDate] <= @BirthDate)">
        <SelectParameters>
            <uuu:MyParameter Name="BirthDate" />
        </SelectParameters>
</asp:SqlDataSource>
```

SelectCommand 標籤中包含一個 BirthDate 名稱的自訂參數，其值為 MyParameter 類別中為傳的 Date 物件(1955 年 1 月 1 日)。因此網頁程式執行時，會把小於此日期的所有員工資料回傳。您可以進一步為 MyParameter 類別提供屬性，以便從網頁中設定年度的參數。



## 練習 1.3：自訂資料來源控制項參數

**目的：**

這個練習主要是了解如何繼承自 Parameter 類別，以自訂資料來源控制項的參數。

**功能描述：**

這個練習會在網站中加入一個類別，繼承自 Parameter 類別並改寫 Evaluate 方法來設定參數值。

**預估實作時間：20 分鐘**

**實作步驟：**

1. 從『Start』 → 『Program』 → 『Microsoft Visual Studio 2008』 → 『Microsoft Visual Studio 2008』，啓動 Visual Studio 2008 開發環境。
2. 從『File』 → 『New Web Site』 → 選取『ASP.NET Web Site』 → 將『Location』設為『File System』並點選『Browse...』按鈕選取『U9544\Practices\VB 或 CS\Mod01\_3\Starter』目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 Visual C#，將站台取名為「Mod01\_3」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Class」，將檔名命名為「MyCustomParameter.vb」或「MyCustomParameter.cs」。若有訊息提示是否將檔案存放在 App\_Code 目錄，請選擇「Yes」。
4. 在「MyCustomParameter.vb」或「MyCustomParameter.cs」檔案上方匯入以下命名空間：

```
Visual Basic
Imports System.Web.UI
Imports System.Web.UI.WebControls
```

```
C#
using System.Web.UI;
using System.Web.UI.WebControls;
```

5. 修改 MyCustomParameter 類別程式碼，使其封裝在 Mod01\_3 命名空間之中：

```
Visual Basic
Namespace Mod01_3
Public Class MyCustomParameter

End Class
End Namespace
```

```
C#
namespace Mod01_3
{
    public class MyCustomParameter : Parameter
    {
    }
}
```

6. 修改 MyCustomParameter 類別程式碼，使其繼承自「Parameter」類別：

```
Visual Basic
Public Class MyCustomParameter
    Inherits Parameter

End Class
```

```
C#
public class MyCustomParameter : Parameter
{
}
```

7. 宣告一個 Year 屬性，屬性的資料保存在 Parameter 類別的 ViewState 集合之中。而在 Evaluate 方法之中便可以使用到這個屬性來設定查詢的年度：

```
Visual Basic
Public Property Year() As Integer
    Get
        Dim obj As Object = MyBase.ViewState("Year")
        If obj Is Nothing Then
            Return 0
        End If
        Return CInt(obj)
    End Get
    Set(ByVal value As Integer)
        If Me.Year <> value Then
            MyBase.ViewState("Year") = value
            MyBase.OnParameterChanged()
        End If
    End Set
End Property
```

```
C#
public int Year
{
    get
    {
        object obj = base.ViewState["Year"];
        if (obj == null)
            return 0;
        return (int)obj;
    }
    set
    {
        if (this.Year != value)
        {
            base.ViewState["Year"] = value;
            base.OnParameterChanged();
        }
    }
}
```

8. 改寫 Evaluate 方法，用來回傳參數的值為一個日期：

## Visual Basic

```
Protected Overrides Function Evaluate(ByVal context As HttpContext, ByVal control As Control) As Object
```

```
    Dim d As New DateTime(Year, 1, 1)
    Return d
End Function
```

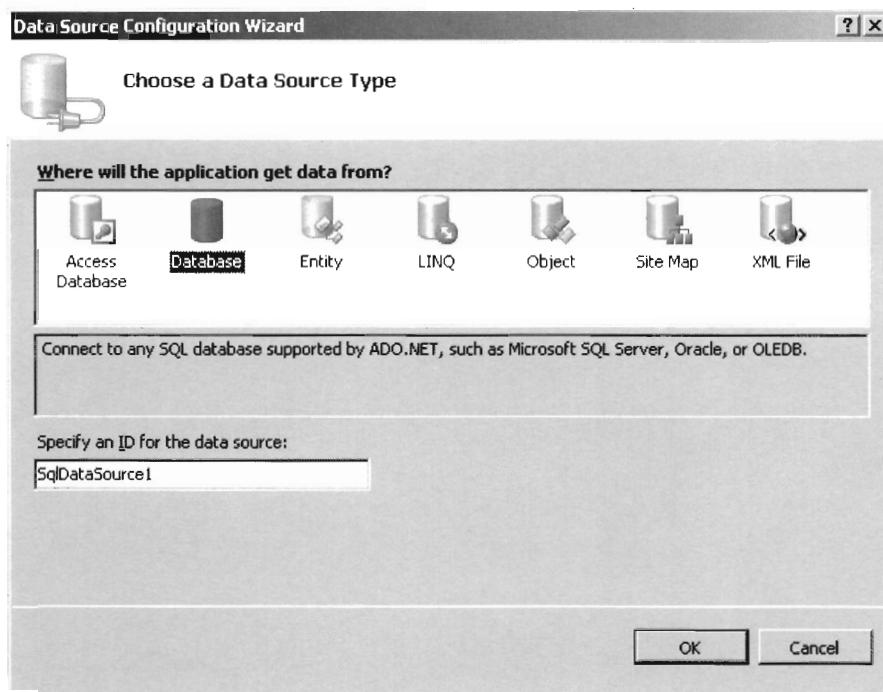
## C#

```
protected override object Evaluate(HttpContext context, Control control)
{
    DateTime d = new DateTime(Year, 1, 1);
    return d;
}
```

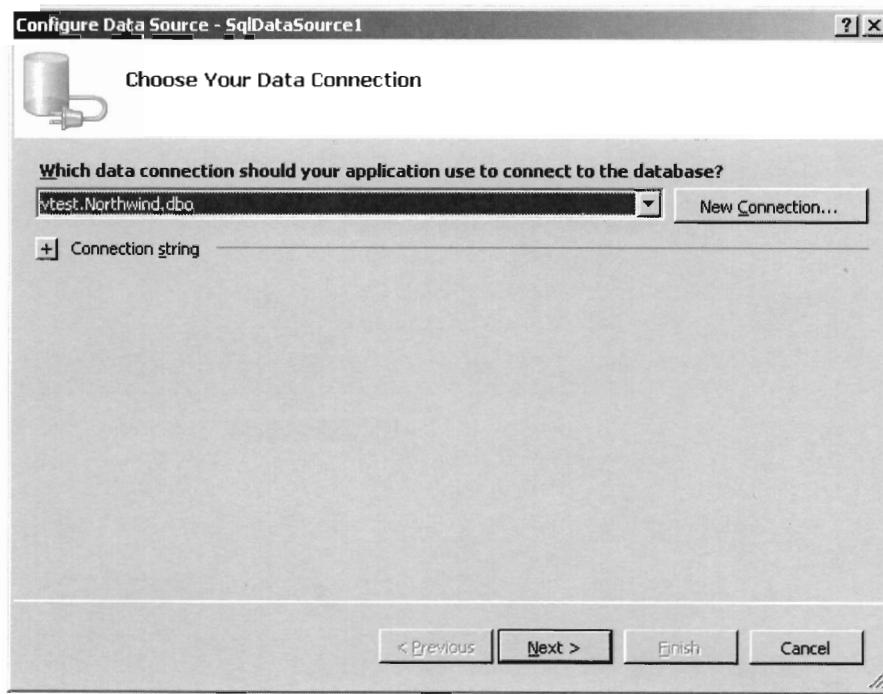
9. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁檔案，使用預設的檔名命名。
10. 在網頁上方註冊欲使用的參數類別所在命名空間，與前置符號：

```
<%@ Register TagPrefix="uuu" namespace="Mod01_3" %>
```

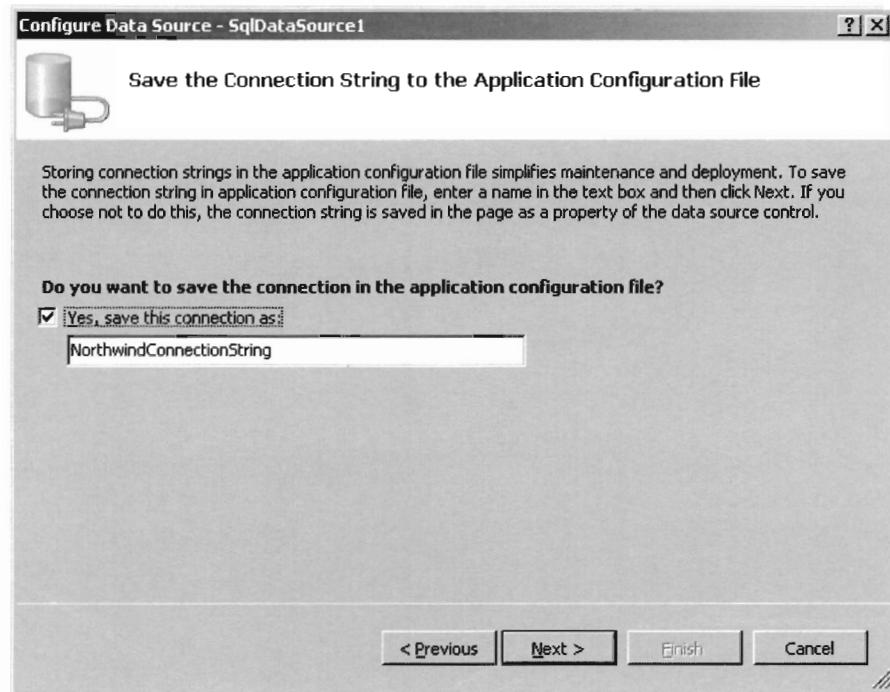
11. 從「Toolbox」工具箱中拖拉一個 GridView 控制項到網頁中。
12. 點選 GridView 控制項的智慧型標籤，從「Choose Data Source」下拉式清單方塊，選取「<New data source>」，選取「Database」當作資料來源，按下「OK」：



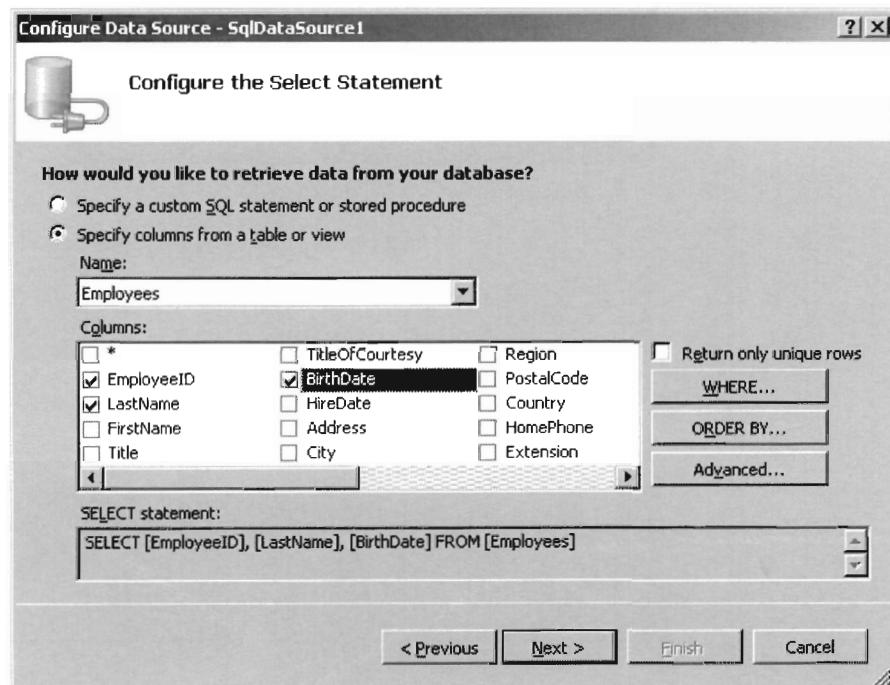
13. 選取 Northwind 資料庫，按下「Next」。



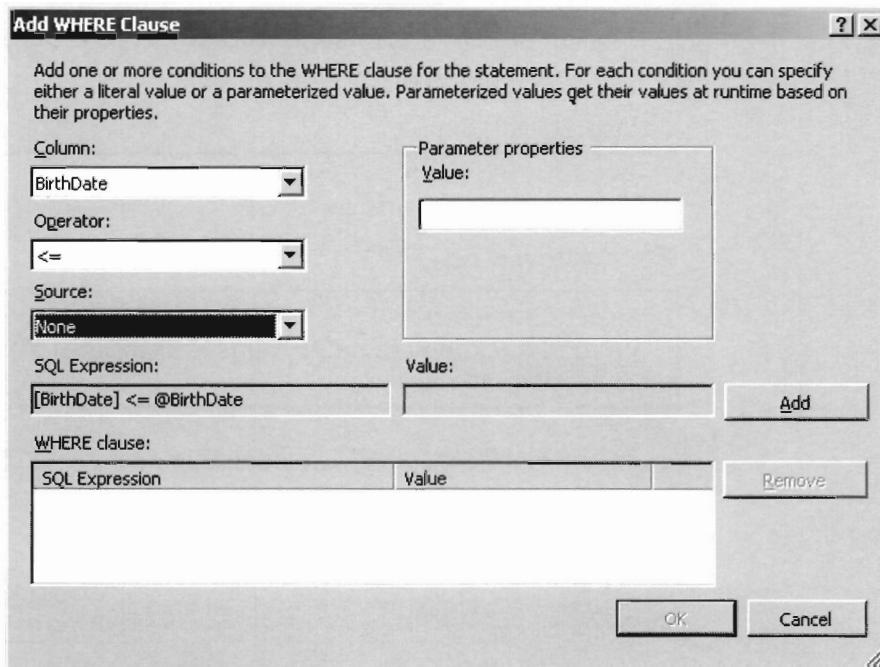
14. 按下「Next」。



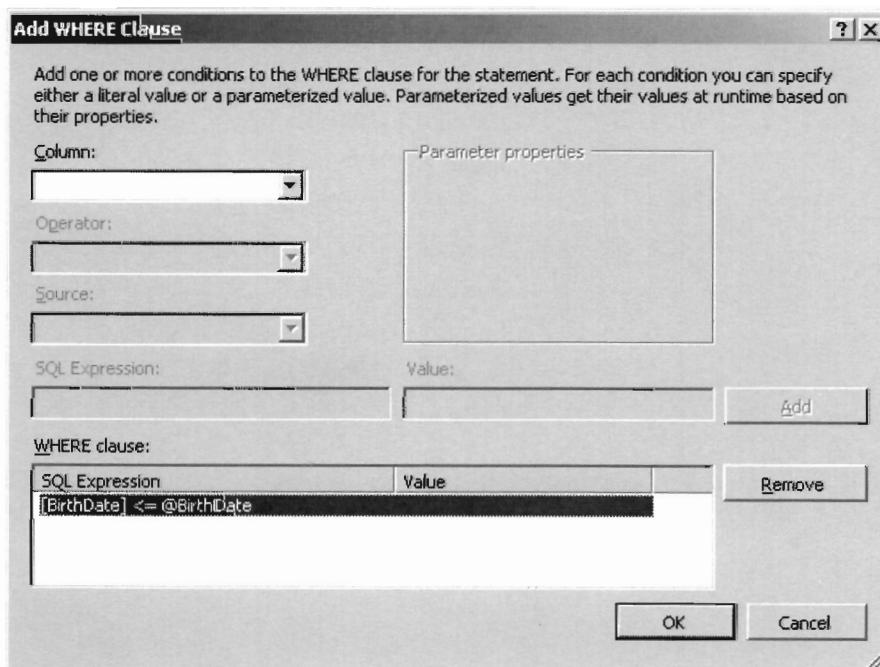
15. 選取查詢 Employees 資料表的 EmployeeID、LastName、BirthDate 欄位：



16. 點選「Where」按鈕，設定 Column 欄位為「Birthdate」；  
Operator 設為「<=」；Source 設為「None」：



然後按下「Add」按鈕，目前畫面看起來為：



按下「OK」按鈕，一直到精靈結束，Visual Studio 會自動產生以下標籤：

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:NorthwindConnectio
nString %>" 
    SelectCommand="SELECT [EmployeeID], [LastName], [BirthDat
e] FROM [Employees] WHERE ([BirthDate] <= @BirthDate)">
    <SelectParameters>
```

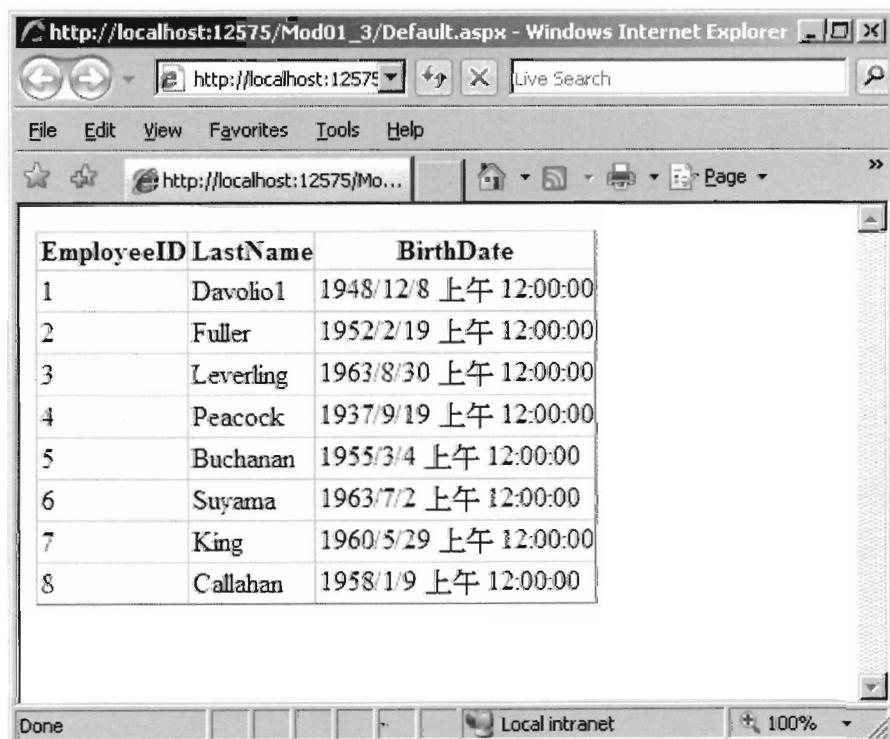
```
<asp:Parameter Name="BirthDate" Type="DateTime" />
</SelectParameters>
</asp:SqlDataSource>
```

17. 修改 SelectParameters 標籤如下，以便在網頁中透過 Year

Attribute 設定年度：

```
<SelectParameters>
<uuu:MyCustomParameter Name="BirthDate" Year="1966"/>
</SelectParameters>
```

18. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。當網頁執行時，便將 1966 之前出生的員工資料表列出來：



The screenshot shows a Microsoft Internet Explorer window displaying a table of employee data. The table has three columns: EmployeeID, LastName, and BirthDate. The data is as follows:

EmployeeID	LastName	BirthDate
1	Davolio	1948/12/8 上午 12:00:00
2	Fuller	1952/2/19 上午 12:00:00
3	Leverling	1963/8/30 上午 12:00:00
4	Peacock	1937/9/19 上午 12:00:00
5	Buchanan	1955/3/4 上午 12:00:00
6	Suyama	1963/7/2 上午 12:00:00
7	King	1960/5/29 上午 12:00:00
8	Callahan	1958/1/9 上午 12:00:00

### 使用 ListView 控制項

- 以樣版為基礎的控制項
- 至少要設定兩個樣版
  - LayoutTemplate：定義使用者介面的配置
  - ItemTemplate：定義資料的展示方式
- 支援資料繫結能力
- 支援新增、刪除、修改、查詢、排序功能
- 能夠將資料分組
- 可搭配 DataPager 分頁

## 使用 ListView 控制項

ListView 控制項是以樣版為基礎的控制項，利用各式各樣的樣版，如 AlternatingItemTemplate、LayoutTemplate、InsertItemTemplate、SelectedItemTemplate、EmptyDataTemplate、EditItemTemplate、ItemTemplate...等等來定義外觀、以及資料在新增、刪除、修改、查詢和選取的過程中，展示資料的樣版，也支援資料繫結能力。 ListView 控制項允許搭配資料來源控制項，如 SqlDataSource、LinqDataSource，來進行資料的新增、刪除、修改動作。

和傳統的 ASP.NET FormView 或 DetailsView 控制項不太一樣的地方是在於，ListView 控制項完全沒有定義使用者介面的配置 (Layout)，您可以完全掌控，讓 ListView 控制項以表格式方式配置資料，或按流水式方式配置。因此，使用 ListView 控制項時，至少要設定兩個樣版：LayoutTemplate 與 ItemTemplate 樣版，其中 LayoutTemplate 樣版用來定義使用者介面的配置，而 ItemTemplate 樣版定義資料的展示方式。同時，你需要在 LayoutTemplate 中預先為資料預留一個控制項，以便讓 ItemTemplate 定義的資料樣版能合併至最終將展示的 HTML 標籤之中。

ListView 控制項還有一項很重要的功能，就是能夠將資料分組，例如想要讓外觀像報紙一樣可以分欄顯示，ListView 控制項也可以很容易的達成。您可以在 LayoutTemplate 樣版中使用伺服器控制項，預留群組的位置，然後設計 GroupTemplate 樣版定義分組資料的展示。

ListView 控制項可以搭配 DataPager 控制項來進行分頁的設計。

DataPager 控制項可以為有實作 IPageableItemContainer 介面的控制項來進行分頁，而目前只有 ListView 控制項有支援，不過您可以實作這個介面，為其他控制項加上分頁的機制。將分頁功能獨立成一個不單獨的 DataPager 控制項，就可以將分頁與展示資料的 ListView 控制項分離，不僅可以重複其他控制項中使用 DataPager 分頁控制項，也可以讓 ListView 控制項負擔較輕，必要時再加入分頁功能。

### ListView 控制項常用事件

- ListView 控制項在執行的過程中會觸發許多資料異動的事件
- 常用事件
  - ItemCommand
  - ItemDataBound
  - ItemDeleted
  - ItemInserted
  - ItemUpdated

## ListView 控制項常用事件

ListView 控制項在執行的過程中會觸發許多資料異動的事件，您可以在程式中捕捉這些事件，以做因應的處理，例如在新增資料完成後，顯示提示訊息。ListView 控制項常用事件如下：

- ItemCommand：當 ListView 控制項中的按鈕被按下觸發。
- ItemDataBound：當 ListView 控制項進行資料繫結的動作時觸發。
- ItemDeleted：在 ListView 控制項刪除資料之後發生。
- ItemDeleting：ListView 控制項刪除資料之前發生。
- ItemEditing：ListView 控制項編輯資料之前發生。
- ItemInserted：ListView 控制項新增資料之後發生。
- ItemInserting：ListView 控制項新增資料之前發生。
- ItemUpdated：ListView 控制項更新資料之後發生。
- ItemUpdating：ListView 控制項更新資料之前發生。

### 以程式動態設定樣式

在資料繫結動作發生時利用**ItemDataBound**事件進行客製化動作

使用**ListViewItem**的**FindControl**方法找尋控制項

```
Visual Basic
Protected Sub ListView1_ItemDataBound(ByVal sender As Object, ByVal e As System.Web.UI.WebControls.ListViewItemEventArgs)
    Dim lbl As Label = CType(e.Item.FindControl("UnitsInStockLabel"), Label)
    lbl.ForeColor = System.Drawing.Color.Red
End Sub
```

```
protected void ListView1_ItemDataBound(object sender, ListViewItemEventArgs e) {
    Label lbl = (Label)e.Item.FindControl("UnitsInStockLabel");
    lbl.ForeColor = System.Drawing.Color.Red;
}
```

### 以程式動態設定樣式

若 ListView 控制項設定了資料來源，ListView 控制項在輸出展示的標籤之前，進行資料項目的繫結動作。此時會觸發 **ItemDataBound** 事件，此事件可以讓你進行一些客製化的動作，如修改資料項目的展示樣式，或其中的值。

在 **ItemDataBound** 事件觸發時，**ListViewEventArgs** 物件會傳入 **ItemDataBound** 事件處理常式中，您可以透過此物件來存取每一筆繫結中的資料項目。使用 **ListViewItem** 的 **FindControl** 方法可以找尋 ListView 中的控制項。

例如以下程式片斷所示，在 **ItemDataBound** 事件處理常式中，找尋到 **Label** 控制項，將其前景顏色設定為紅色：

```
Visual Basic
Protected Sub ListView1_ItemDataBound(ByVal sender As Object,
ByVal e As System.Web.UI.WebControls.ListViewItemEventArgs)
    Dim lbl As Label = CType(e.Item.FindControl("UnitsInStockLabel"), Label)
    lbl.ForeColor = System.Drawing.Color.Red
End Sub
```

```
C#
```

```
protected void ListView1_ItemDataBound(object sender, ListViewEventArgs e)
{
    Label lbl = (Label)e.Item.FindControl("UnitsInStockLabel");
    lbl.ForeColor = System.Drawing.Color.Red;
}
```

### 練習1.4：以程式動態設定展示樣式

這個練習主要是了解如何善用 ListView 控制項的事件，以程式來動態設定 ListView 中資料的展示樣式。

•預估實作時間：12分鐘

### 練習 1.4：以程式動態設定展示樣式

#### 目的：

這個練習主要是了解如何善用 ListView 控制項的事件，以程式來動態設定 ListView 中資料的展示樣式。

#### 功能描述：

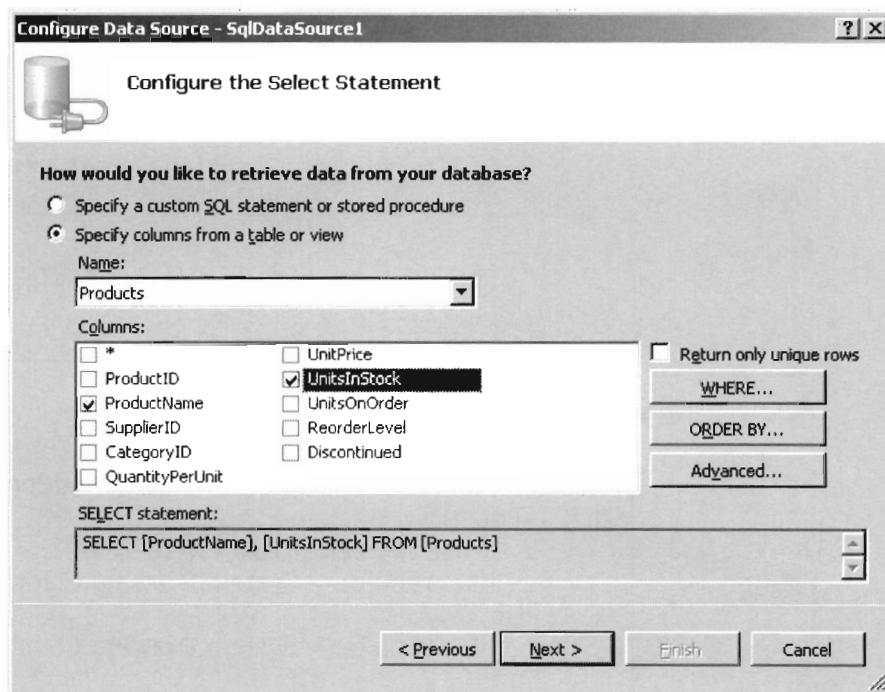
這個練習會為 ListView 控制項加入 ItemDataBound 事件處理常式，以程式動態設定 ListView 控制項中資料的展示樣式。練習將從 Northwind 資料庫查詢出 Products 資料表資料，若產品的 UnitsInStock 值大於 10 便以紅色字顯示，若 UnitsInStock 值小於等於 10 便以綠色字顯示。

預估實作時間：12 分鐘

#### 實作步驟：

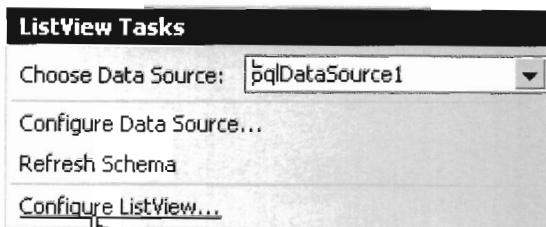
1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。

2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod01\_4\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod01\_4」。
3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 從「Toolbox」工具箱中拖拉一個 ListView 控制項到網頁中。
5. 點選 ListView 控制項的智慧型標籤，從「Choose Data Source」下拉式清單方塊，選取「<New data source>」，選取「Database」當作資料來源，按下「OK」。
6. 按「Next」，直到「Configure the Select Statement」視窗，選取查詢 Products 資料表的 ProductName 與 UnitsInStock 兩個欄位：

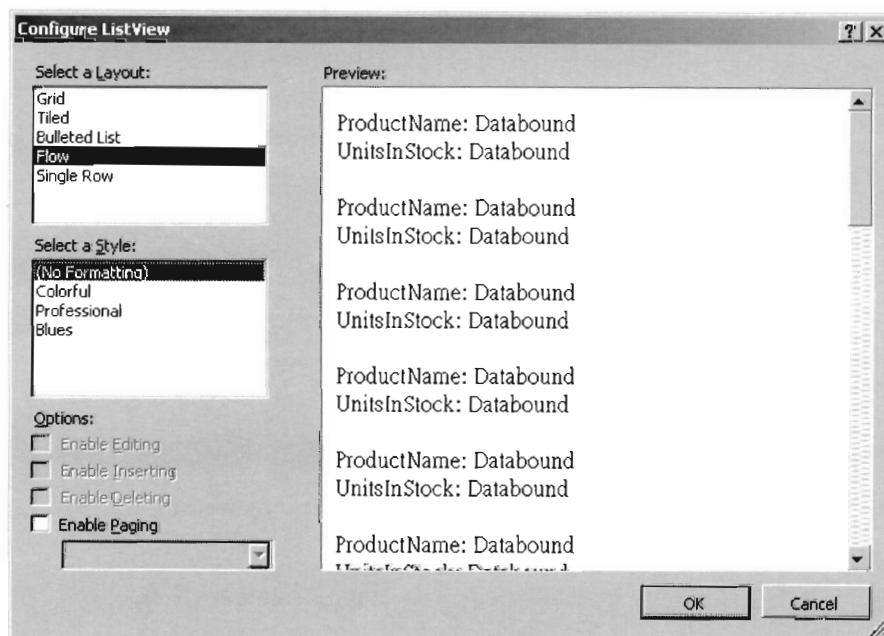


7. 按「Next」直到精靈結束。

8. 點選 ListView 控制項的智慧型標籤，點選「Configure ListView」：



設定 Layout 為 Flow：



9. 切換到設計畫面，產生 ListView 控制項的 ItemDataBound 事件處理常式，加入以下程式碼：

#### Visual Basic

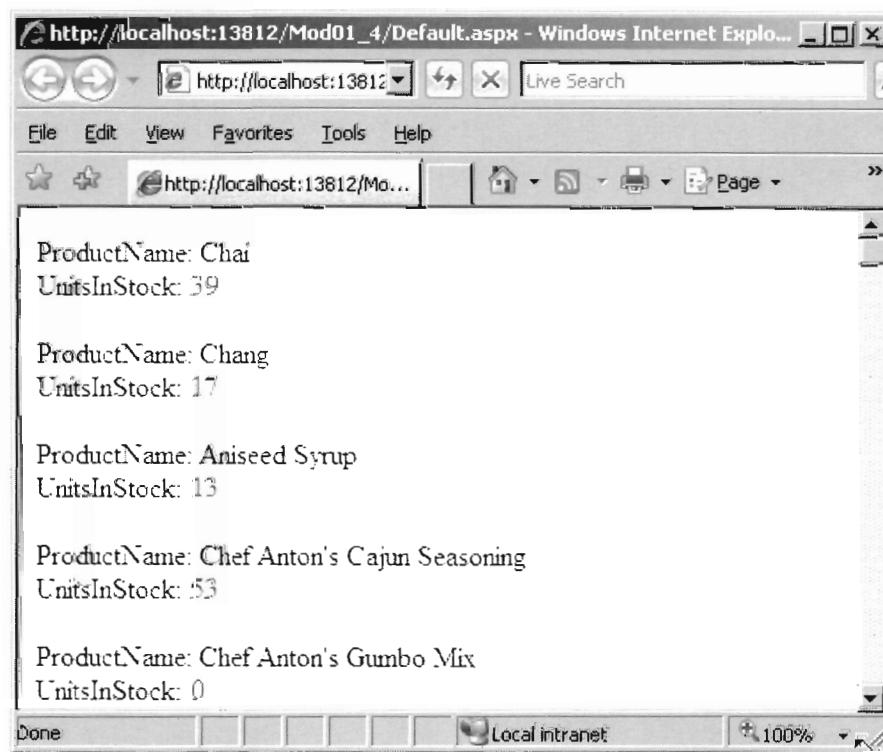
```
Protected Sub ListView1_ItemDataBound(ByVal sender As Object,
ByVal e As System.Web.UI.WebControls.ListViewEventArgs)
    Dim lbl As Label = CType(e.Item.FindControl("UnitsInStockLabel"), Label)
    If Integer.Parse(lbl.Text) > 10 Then
        lbl.ForeColor = System.Drawing.Color.Red
    Else
        lbl.ForeColor = System.Drawing.Color.Green
    End If
End Sub
```

#### C#

```
protected void ListView1_ItemDataBound(object sender, ListViewI
```

```
temEventArgs e)
{
    Label lbl = (Label)e.Item.FindControl("UnitsInStockLabel");
    if (int.Parse(lbl.Text) > 10)
    {
        lbl.ForeColor = System.Drawing.Color.Red;
    }
    else
    {
        lbl.ForeColor = System.Drawing.Color.Green;
    }
}
```

10. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。當按鈕被按下時，便查詢出產品資料，若 UnitsInStock 值大於 10 便以紅色字顯示，若 UnitsInStock 值小於等於 10 便以綠色字顯示，參考：



## 總結

- ADO.NET 資料存取架構
- 連線字串的設計與存取
- ADO.NET物件介紹
- ASP.NET的資料繫結架構
- 設計SqlDataSource資料來源參數
- 進階使用ListView

---

## 第二章: 資料存取進階設計

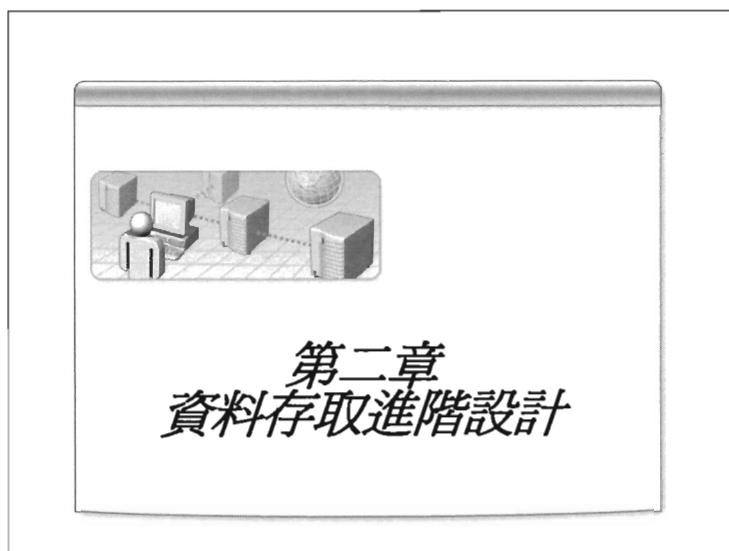
### 本章大綱

設計資料元件 .....	6
透過類別提供的方法維護資料 .....	7
獨立的類別庫 .....	9
練習 2.1 : 設計資料元件 .....	10
在資料元件的 SelectMethod 裡加入參數 .....	17
ObjectDataSource 設定參數 .....	18
練習 2.2 : 設定資料元件的查詢參數 .....	19
排序參數設定 .....	23
練習 2.3 : 設定 ObjectDataSource 排序 .....	24
ObjectDataSource 編輯資料 .....	28
DataObjectMethod 屬性標籤 .....	28
練習 2.4 : 設計 ObjectDataSource 的更新功能 .....	31
何謂 LINQ .....	40
LINQ 專案類型 .....	41
LINQ To SQL Classes .....	42
ASP.NET 中的 LinqDataSource .....	45
練習 2.5 : 設定 LINQ to SQL 連接到資料庫 .....	47

作者：

趙敏翔





---

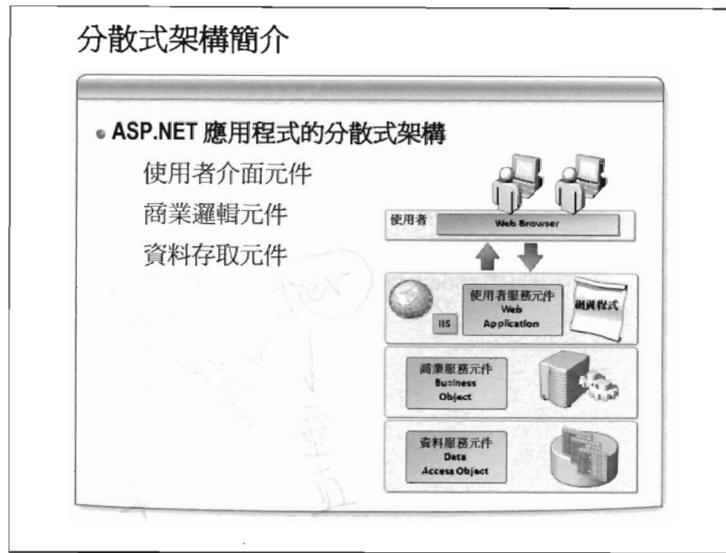
## 大綱

- 分散式架構簡介
- 設計資料元件 ObjectDataSource
- 資料元件的異動
- 設計 LINQ to SQL
- 使用 LinqDataSource
- 總結

本章將介紹資料存取的進階技巧，先簡介使用 ASP.NET 設計分散式架構的概念，以及利用 ObjectDataSource 搭配可讓網站應用程式存取資料的商業共用類別等技術，此外，也將進一步探討 ASP.NET 與 LINQ 整合的進階議題。

在這一章中將學習到

- 設計 ObjectDataSource 資料元件對實際的資料來源存取。
- 設計 ObjectDataSource 資料元件參數
- 設計 ObjectDataSource 資料元件異動功能
- 在 ASP.NET 中設計 LINQ to SQL 類別。
- 使用 LinqDataSource 擷取資料。



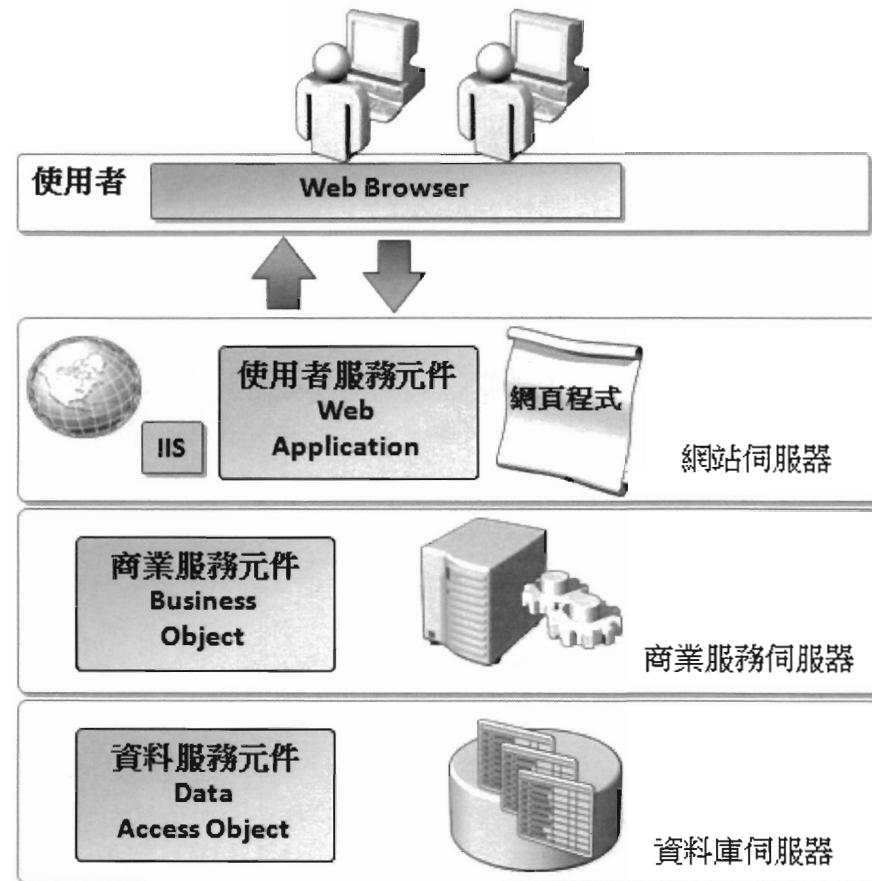
## 分散式架構中的元件角色

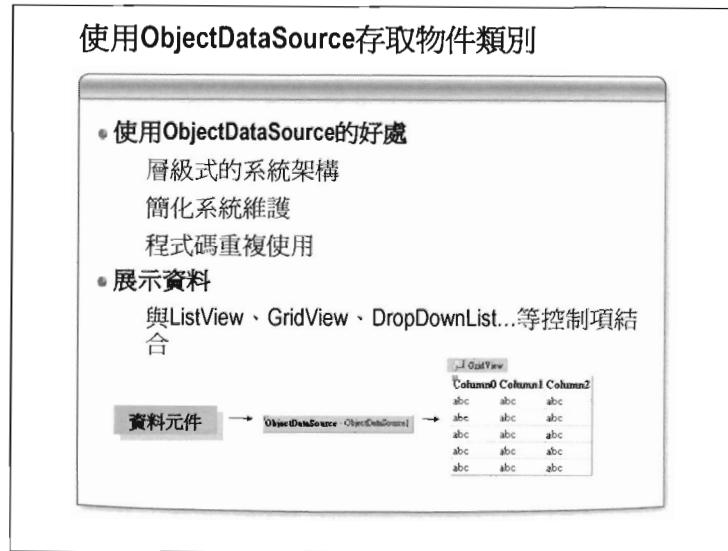
在業界不管是企業級的 ERP 系統或是人力資源 HR 系統、財會系統等等都有可能包含著無數的網站應用程式、Windows 應用程式、Web Service 甚至 PDA 和 Mobile Phone 等用戶端應用程式，因此通常在系統分析時就會將一些共用的商業邏輯或是常見功能獨立設計成元件來使用。

設計分散式應用程式並不是一件簡單的工作，在架構、設計及實作階段，都必須作許多決策。這些決策會影響應用程式的安全性、擴展性、可用性及管理性等「能力」，也會影響目標基礎結構的架構、設計及實作。目前最常見的應用程式架構可以依照應用程式的邏輯設計來簡單分為三層，分別為：

- 使用者介面元件
- 商業邏輯元件
- 資料存取元件

邏輯架構觀點的價值在於識別任何系統一般的服務類型，或是定義層與層之間的介面，多層式的設計能讓開發者在實作各層時製作更周密的架構，並建置更好維護的應用程式。





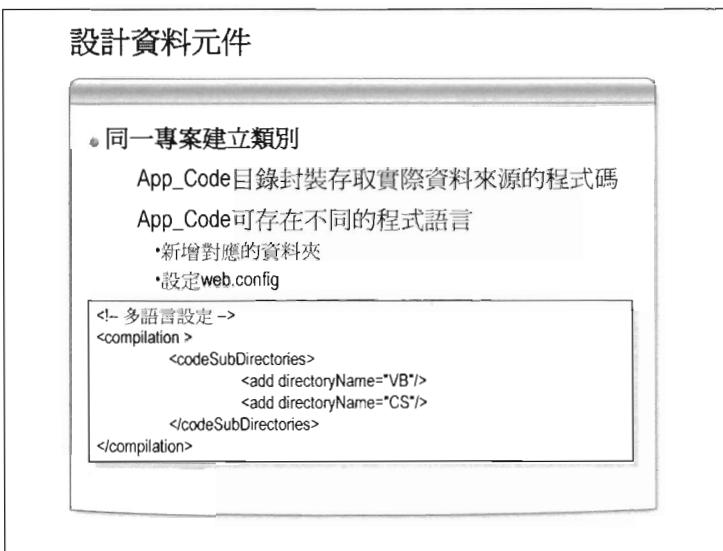
## 設計資料元件

針對 ASP.NET 網頁設計的架構來說，以 SqlDataSource 提供網頁資料，這樣的架構是二層式的。在 ASP.NET 若要建構 N-Tier 的 Web 應用程式，則需透過 ObjectDataSource 建置商務元件，它同樣可讓前端的資料繫結控制項很容易地存取資料，但又可兼顧封裝商業邏輯的好處。

在一個簡單的 Web 應用程式裡，的確沒有什麼理由不用 SqlDataSource，但隨著應用程式愈來愈複雜，把資料存取放在獨立的資料元件中會讓系統維護變得簡單許多。將商業邏輯集中在商業邏輯層物件，不同的網頁都可使用這些程式，也間接達到程式碼共用的好處。

.NET Framework 的 ObjectDataSource 控制項，可與許多下控制項結合進行資料展現及編輯。以下為常用的資料展現控制項：

- ListView
- GridView
- DropDownList
- TreeView



## 設計資料元件

所有資料存取的程式碼可封裝在此資料元件的類別中，外界要存取資料只要呼叫相關的方法，即可對實際的資料來源進行新增、刪除、修改、查詢。.NET 以類別的方式實作資料元件。類別可在 ASP.NET 應用程式的根目錄之下建立名為 App\_Code 的子目錄，App\_Code 目錄下的程式碼，ASP.NET 應用程式執行時會被自動編譯。

再則 App\_Code 下可以透過子目錄分類來置放不同的程式語言類別，例如可以同時存在使用 Visual Basic 或是 C# 所設計的類別檔，設計方式有兩個步驟：

1. 先在 App\_Code 目錄中建立兩個子目錄，分別命名為：VB 和 CS。
2. 設定 web.config 組態檔的<codeSubDirectories>項目，設定如下：

```
<compilation>
  <codeSubDirectories>
    <add directoryName="VB"/>
    <add directoryName="CS"/>
  </codeSubDirectories>
```

</compilation>

### 不同專案建立類別

- 不同專案建立類別
  - 使用類別函式庫專案  
Bin目錄下的 \*.dll
- 透過類別提供的方法維護資料
  - 自訂查詢、刪除、修改、新增資料等靜態方法
- 類別設計時
  - 用 System.ComponentModel.DataObject() 設定

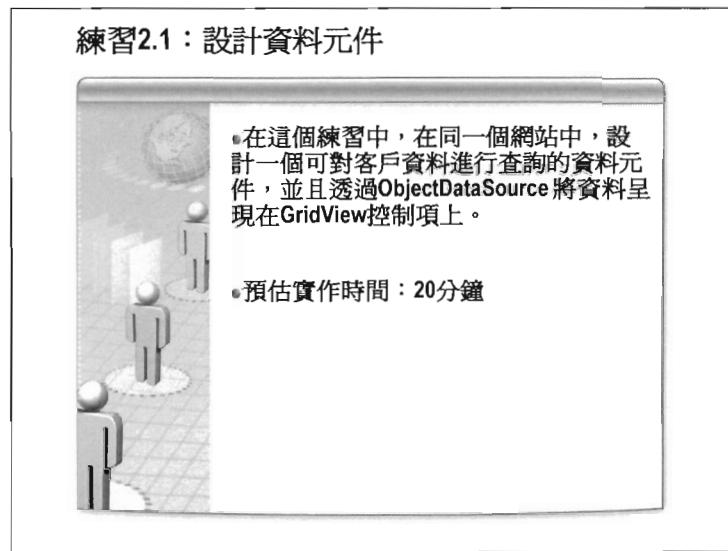
### 不同專案建立類別

ObjectDataSource 的資料來源也可以是獨立的類別庫(Class Library)專案中所建立的類別，開發者在設計時，可以將它編譯成單獨的組件，以便在其他應用程式裡共用。以 ASP.NET 應用程式為例，獨立的組件可放在 Bin 目錄下，附檔名為 dll。

不過值得注意的是，如果設計的類別是明確宣告為要讓 ObjectDataSource 控制項來使用對應，那麼在類別設計時，必須先加上 System.ComponentModel 命名空間下的 DataObject Attribute，程式碼如下：

```
Visual Basic
<System.ComponentModel.DataObject()> _
Public Class CustomerInfo
    ...
End Class
```

```
C#
[System.ComponentModel.DataObject()]
public class CustomerInfo
{
    ...
}
```



## 練習 2.1：設計資料元件

### 目的：

這個練習主要是了解如何使用 ObjectDataSource 控制項，整合自訂用來封裝商業邏輯的類別，將資料呈現在網頁上。

### 功能描述：

在這個練習中，在同一個網站中，使用兩個不同的語言 Visual Basic 和 Visual C#，分別設計一個可對客戶資料進行查詢的資料類別元件，並使用兩個不同的網頁程式，分別抓取 Visual Basic 和 Visual C#的類別，透過 ObjectDataSource 將資料呈現在 GridView 控制項上。

### 預估實作時間：20分鐘

### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。

2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod02\_1\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod02\_1」。
3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 在「Solution Explorer」視窗中選取專案，點選滑鼠右鍵，選擇「Add ASP.NET Folder」→點選「App\_Code」。
5. 點選 App\_Code 資料夾，點選滑鼠右鍵，選擇「New Folder」新增一個子資料夾，命名為：「VB」，並且「VB」資料夾中加入一個類別檔。
6. 點選 VB 資料夾，自主選單「Web Site」下「Add New Item...」，選取「Class」，命名為「CustomerInfo」，Language 為 Visual Basic。
7. 修改 CustomerInfo.vb 程式碼中類別 class 的名稱為 CustomerInfoVB，程式碼如下：

```

Visual Basic
Imports Microsoft.VisualBasic
Public Class CustomerInfoVB

End Class

```

8. 編譯網站，點選專案，自主選單「Build」下選取「Build Web Site」。
9. 在 CustomerInfoVB.vb 類別檔中匯入以下命名空間：

```

Visual Basic
Imports System.Data
Imports System.Data.SqlClient

```

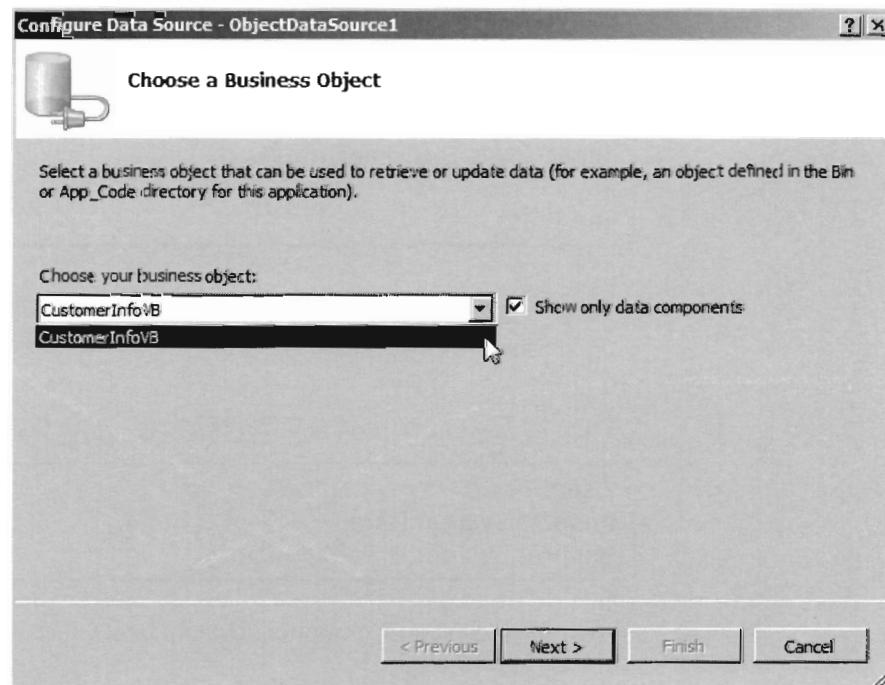
10. 使用 System.ComponentModel.DataObject 設定類別，並在類別中加入一個 GetCustomers 方法，擷取資料庫的客戶資料，並回傳一個 DataTable 物件，程式碼設計如下：

```

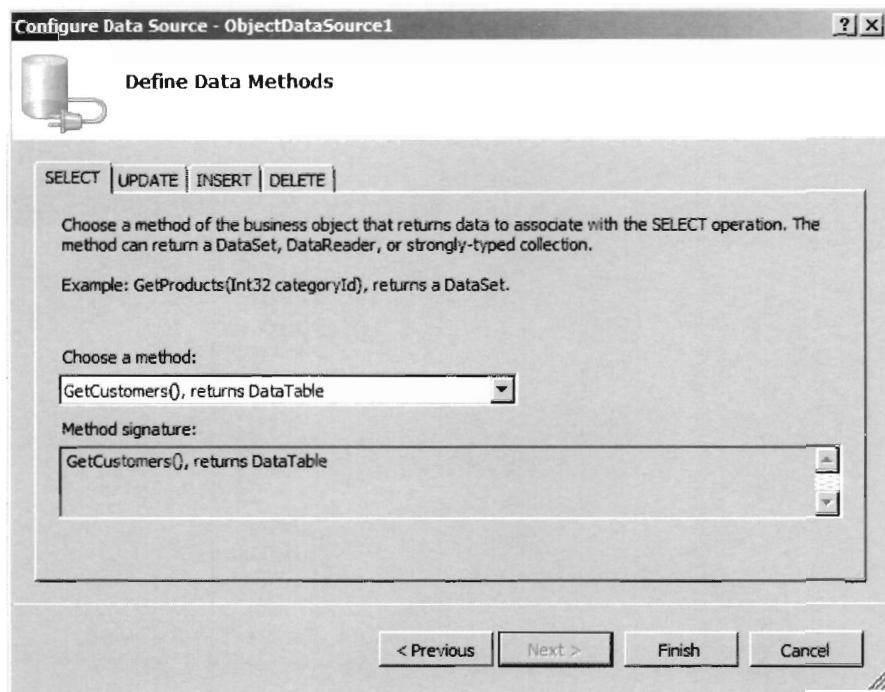
Visual Basic
<System.ComponentModel.DataObject()> _
Public Class CustomerInfoVB
    Public Function GetCustomers() As DataTable
        Dim dt As New DataTable()
        Using cn As New SqlConnection("Data Source=.;Initial Catalog
=Northwind;Integrated Security=True")
            Dim da As New SqlDataAdapter("Select * from Customers", c
n)
            da.Fill(dt)
        End Using
        Return dt
    End Function
End Class

```

11. 從「Toolbox」工具箱中拖拉一個 GridView 控制項到網頁中。
12. 點選 GridView 控制項的智慧型標籤，從「Choose Data Source」下拉式清單方塊，選取「<New data source>」，選取「ObjectDataSource」當作資料來源，按下「OK」。
13. 選擇「CustomerInfoVB」做為其 business Object，按下「OK」。



14. 「GetCustomers」做為其「Select」方法，按「Next」按鈕。

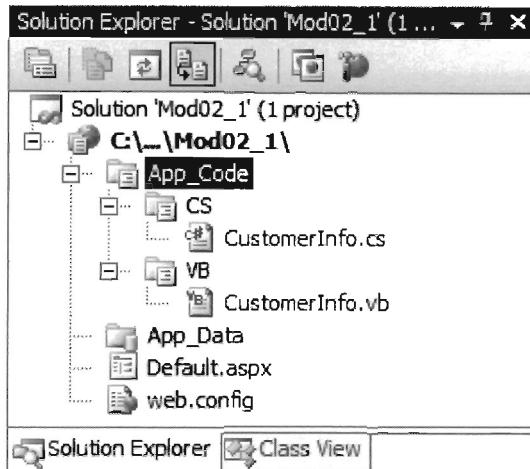


15. 執行網頁測試。在『Solution Explorer』→點選網頁→按滑鼠右鍵→選「View In Browser」。網頁呈現透過 ObjectDataSource 所擷取的客戶資料：

	CustomerID	CompanyName	ContactName	ContactTitle	Address	City
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitucion 2222	Mexico D.F.	
ANTON	Antonio Moreno Taqueria	Antonio Moreno	Owner	Mataderos 2312	Mexico D.F.	
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London	
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvagen 8	Luleå	
BLAUS	Blauer See	Hanna Moos	Sales	Forsterstr. 57	Mannheim	

16. 點選 App\_Code 資料夾，點選滑鼠右鍵，選擇「New Folder」新增一個子資料夾，命名為：「CS」，並且「CS」資料夾中加入一個類別檔。

17. 點選 CS 資料夾，自主選單「Web Site」下「Add New Item...」，選取「Class」，命名為「CustomerInfo」，Language 為 Visual C#。



18. 修改 CustomerInfo.cs 程式碼中類別 class 的名稱為 CustomerInfoCS，程式碼如下：

```
C#
public class CustomerInfoCS
{
}
```

19. 在 CustomerInfoCS.cs 類別檔中匯入以下命名空間：

```
C#
using System.Data;
using System.Data.SqlClient;
```

20. 使用 System.ComponentModel.DataObject 設定類別，並在類別中加入一個 GetCustomers 方法，擷取資料庫的客戶資料，並回傳一個 DataTable 物件，程式碼設計如下：

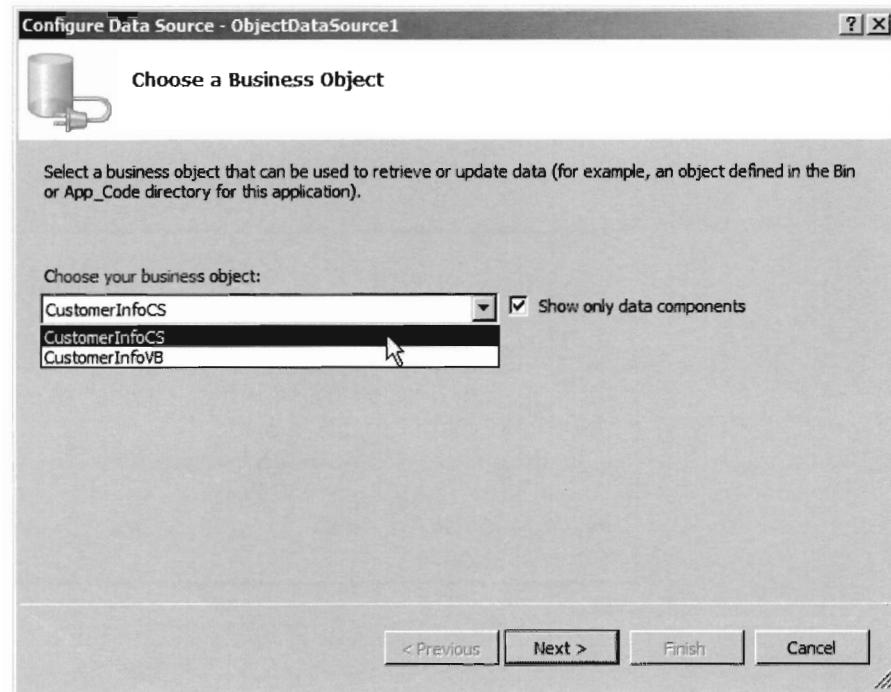
```
Visual C#
[System.ComponentModel.DataObject()]
public class CustomerInfoCS
{
    public DataTable GetCustomers()
    {
        DataTable dt = new DataTable();
        using (SqlConnection cn =
            new SqlConnection("Data Source=.;Initial Catalog=Northwind;Integrated Security=True"))
        {
            SqlDataAdapter da = new SqlDataAdapter("Select * from Cust
```

```
omers", cn);
        da.Fill(dt);
    }
    return dt;
}
}
```

21. 開啓 web.config 組態檔，找到<compilation>項目，在此項目中加入允許多種語言同時存在的設定：

```
<compilation debug="false" strict="false" explicit="true">
  <codeSubDirectories>
    <add directoryName="VB"/>
    <add directoryName="CS"/>
  </codeSubDirectories>
</compilation>
```

22. 編譯網站，點選專案，自主選單「Build」下選取「Build Web Site」。
  23. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增另一個網頁，使用預設的檔名命名。
  24. 從「Toolbox」工具箱中拖拉一個 GridView 控制項到網頁中。
  25. 點選 GridView 控制項的智慧型標籤，從「Choose Data Source」下拉式清單方塊，選取「<New data source>」，選取「ObjectDataSource」當作資料來源，按下「OK」。
  26. 這個網頁選擇「CustomerInfoCS」做為其 business Object，按下「OK」。



27. GetCustomers」做為其「Select」方法，按「Next」按鈕。

28. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。網頁透過另一個語言的物件設計，呈現透過 ObjectDataSource 所擷取的客戶資料：

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region
ALFKI	Alfredsadsas	Maria Anders	Sales Representative	Obere Str. 57	Berlin	
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitucion 2222	Mexico D.F.	
ANTON	Antonio Moreno Taqueria	Antonio Moreno	Owner	Mataderos 2312	Mexico D.F.	
ESOT			Sales	120 Hanover		

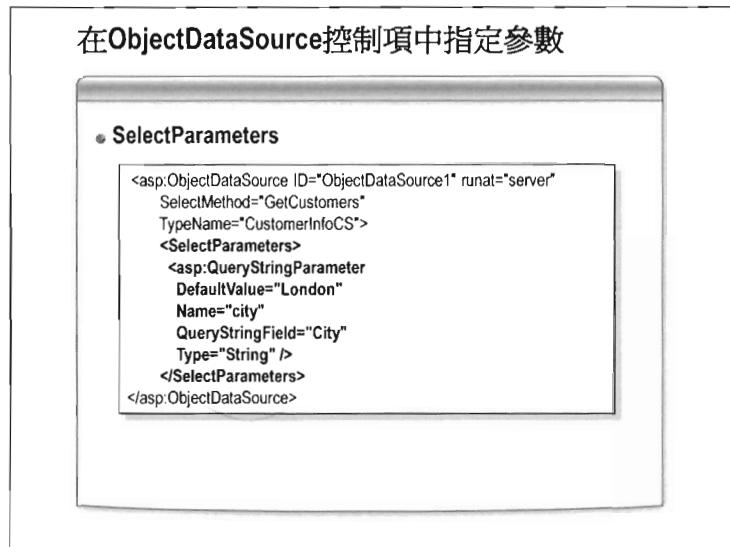


## 參數的設計

為了讓自訂的商務物件更有彈性，可以修改原本的 GetCustomers 方法加入一個名為「City」的字串型別參數，以便在呼叫時可根據客戶所在的城市進行查詢：

```
Visual Basic
Public Function GetCustomers(ByVal city As String) As DataTable
...
End Function
```

```
C#
public DataTable GetCustomers(string city)
{
    ...
}
```



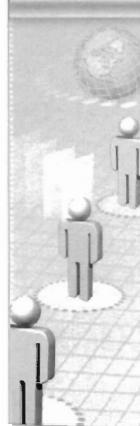
## 在 ObjectDataSource 控制項中指定參數

ObjectDataSource 控制項的要設定參數時，可根據搭配的執行方法而有不同的參數設定，例如 SelectMethod 方法若有參數，所有的參數都要在<SelectParameters>…</SelectParameters>中指定。

在 ObjectDataSource 中所存在的 SelectParameters 參數設定如下：

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server">
    SelectMethod="GetCustomers"
    TypeName="CustomerInfoCS">
    <SelectParameters>
        <asp:QueryStringParameter
            DefaultValue="London"
            Name="city"
            QueryStringField="City"
            Type="String" />
    </SelectParameters>
</asp:ObjectDataSource>
```

### 練習2.2：設計資料元件的查詢參數



- 在這個練習中，設計對客戶資料進行查詢時，加入一個查詢參數。
- 預估實作時間：10分鐘

## 練習 2.2 : 設定資料元件的查詢參數

### 目的：

這個練習主要是了解如何設計 ObjectDataSource 的商務物件允許接受使用者端所傳入的參數來變化運用。

### 功能描述：

在這個練習中，設計對客戶資料進行查詢時，加入一個查詢參數，使用客戶所在的城市當作查詢條件。

### 預估實作時間：10 分鐘

### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「Open Web Site」→選取「ASP.NET Web Site」→選擇「File System」設定 Folder 選取「\U9544\Practices\VB 或 CS\Mod02\_2\Starter\Mod02\_2」目錄，點選「Open」。

3. 打開 App\_Code 中子目錄(VB 或是 CS)下的 CustomerInfo.vb 或是 CustomerInfo.cs。

4. 加入一個接收 City 字串參數的方法 GetCustomers：

Visual Basic

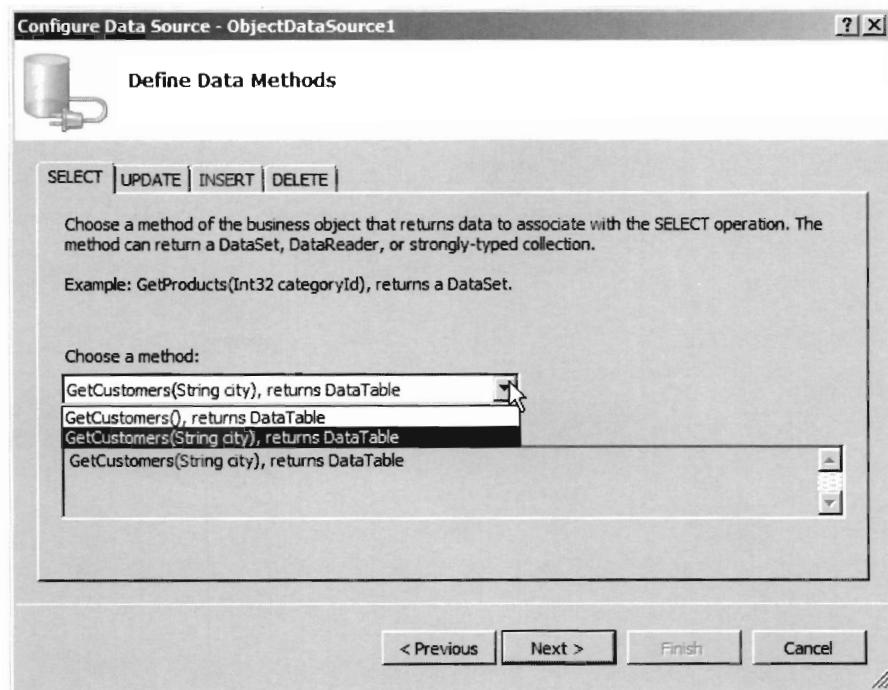
```
Public Function GetCustomers(ByVal city As String) As Data
Table
    Dim dt As New DataTable()
    Using cn As New SqlConnection("Data Source=.;Initial Catalog
=Northwind;Integrated Security=True")
        Dim da As New SqlDataAdapter("Select * from Customers Wh
ere City=@city", cn)
        da.SelectCommand.Parameters.AddWithValue("@city", city)
        da.Fill(dt)
    End Using
    Return dt
End Function
```

C#

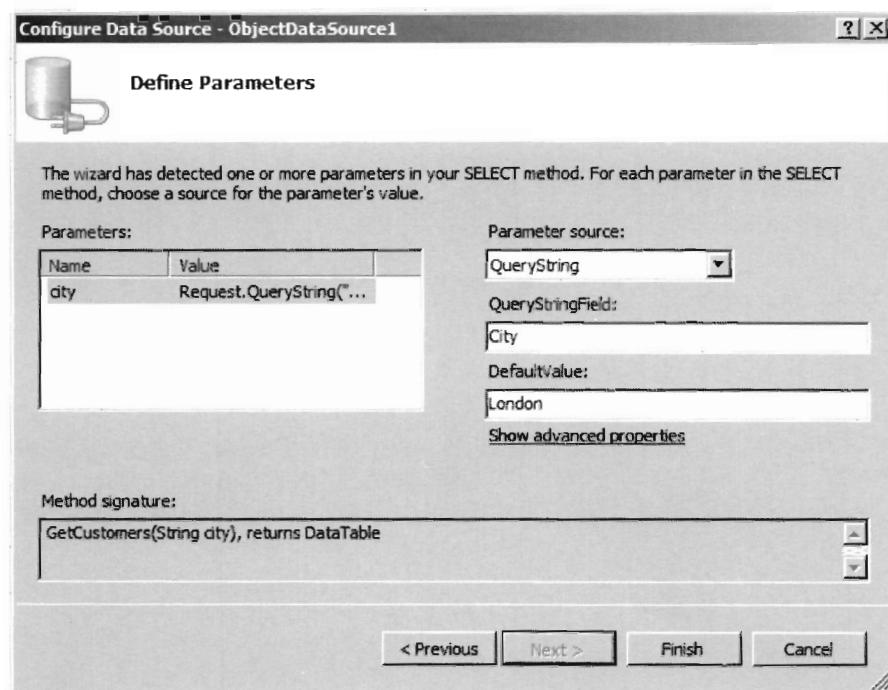
```
public DataTable GetCustomers(string city)
{
    DataTable dt = new DataTable();
    using (SqlConnection cn =
        new SqlConnection("Data Source=.;Initial Catalog=Northwind;I
ntegrated Security=True"))
    {
        SqlDataAdapter da = new SqlDataAdapter("Select * from Custo
mers Where City=@city", cn);
        da.SelectCommand.Parameters.AddWithValue("@city", city);
        da.Fill(dt);
    }
    return dt;
}
```

5. 儲存或關閉 CustomerInfo.vb 或 CustomerInfo.cs。
6. 編譯網站，點選專案，自主選單「Build」下選取「Build Web
Site」。
7. 切換到網頁的「Design」畫面。
8. 點選 ObjectDataSource1 智慧標籤中的「Configure Data
Source...」，選擇「CustomerInfoVB」或是
「CustomerInfoCS」做為其「Business Object」，按
「Next」。

9. 挑選有參數的方法，做為其「Select」方法，按「Next」。

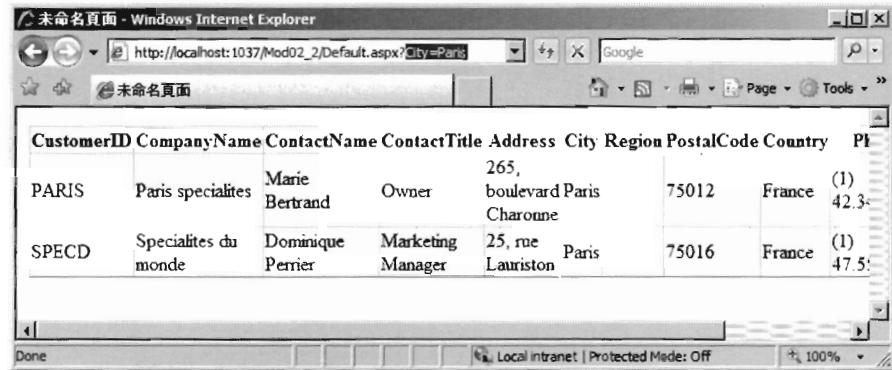


10. 「City」的參數來源設為「QueryString」、QueryStringField 設為「City」、DefaultValue 設為「London」。



11. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。

12. 當網頁執行時，設定網址中的 QueryString，設定 City 為 Paris，確認程式執行結果是否只顯示特定之 Paris 城市的客戶資料：



The screenshot shows a Windows Internet Explorer window displaying a grid of customer data. The URL in the address bar is `http://localhost:1037/Mod02_2/Default.aspx?City=Paris`. The grid has columns for CustomerID, CompanyName, ContactName, ContactTitle, Address, City, Region, PostalCode, Country, and Phone. Two rows of data are visible:

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone
PARIS	Paris specialités	Marie Bertrand	Owner	265, boulevard Charonne	Paris	(1)	75012	France	42.3.
SPECD	Specialités du monde	Dominique Perrier	Marketing Manager	25, rue Lauriston	Paris	(1)	75016	France	47.5.

### 使用 ObjectDataSource 的排序設計

#### SortParameterName 屬性

指定 Select 方法的排序參數，參數型別是字串

```
<asp:ObjectDataSource
  ID="EmployeesObjectDataSource"
  runat="server"
  TypeName="NorthwindEmployee"
  SortParameterName="SortColumns"
  SelectMethod="GetAllEmployees" >
</asp:ObjectDataSource>
```

- 多欄位排序

資料行名稱會以逗號分隔

• 如："LastName, BirthDate DESC"

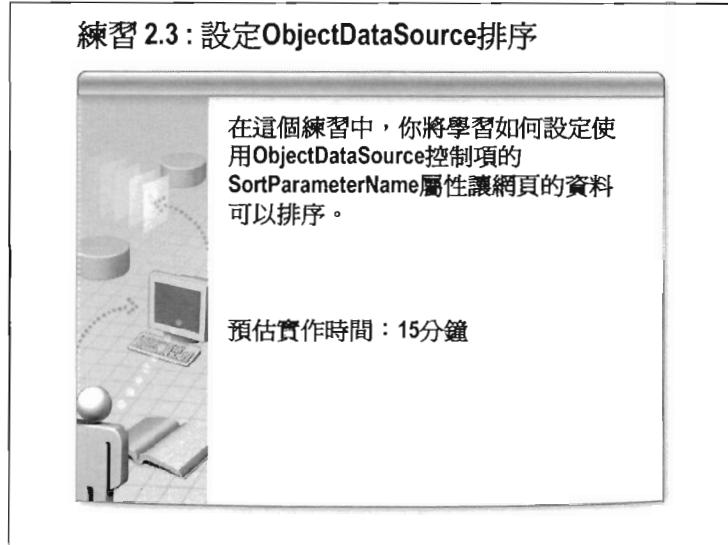
## 使用 ObjectDataSource 的排序設計

如果想透過 ObjectDataSource 排序資料，可使用 ObjectDataSource 控制項的 SortParameterName 屬性，指定 Select 商務物件方法的排序參數。

SortParameterName 屬性會識別用來將排序資料行名稱，傳遞給商務物件方法的參數名稱，且參數型別為字串。

```
<asp:ObjectDataSource
  ID="EmployeesObjectDataSource"
  runat="server"
  TypeName="NorthwindEmployee"
  SortParameterName="SortColumns"
  SelectMethod="GetAllEmployees" >
</asp:ObjectDataSource>
```

如果排序時想要使用多個欄位來排序資料，那麼資料行名稱就要設定以逗號分隔。若要指定遞減排序，排序運算式會包含之後跟隨 DESC 即可。如：在遞增排序時會是 "LastName, BirthDate"，而在遞減排序時會是 "LastName, BirthDate DESC"。



## 練習 2.3 : 設定 ObjectDataSource 排序

### 目的：

這個練習主要是了解如何設計 ObjectDataSource 的排序功能，透過 SortParameterName 屬性的指定，讓商務物件允許接受使用者端所傳入的排序欄位。

### 功能描述：

在這個練習中，設計對客戶資料進行排序時，透過 GridView 控制項本身排序的功能，搭配 ObjectDataSource 針對客戶進行排序。

**預估實作時間：15分鐘**

### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「Open Web Site」→選取「ASP.NET Web Site」→選擇「File System」設定 Folder 選取「U9544\Practices\VB 或 CS\Mod02\_3\Starter\Mod02\_3」目錄，點選「Open」。

3. 打開 App\_Code 中子目錄(VB 或是 CS)下的 CustomerInfo.vb 或是 CustomerInfo.cs。

4. 加入一個接收 sortColumns 字串參數的方法

GetSortCustomers :

Visual Basic

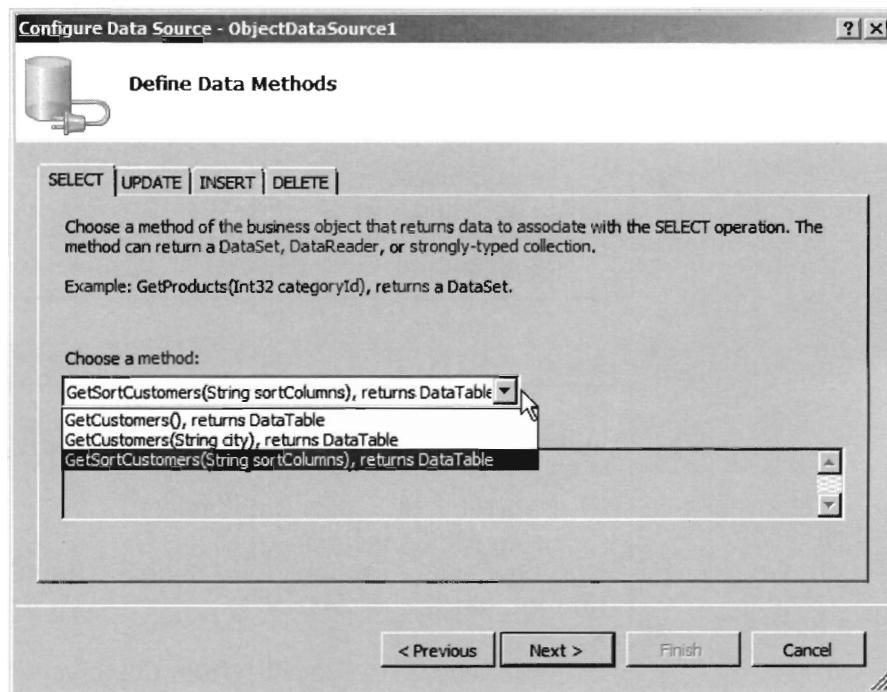
```
Public Function GetSortCustomers(ByVal sortColumns As String)
As DataTable
    Dim dt As New DataTable()
    Using cn As New SqlConnection("Data Source=.;Initial Catalog=Northwind;Integrated Security=True")
        Dim tsql As String = "Select * from Customers "
        If sortColumns.Trim() = "" Then
            tsql += "ORDER BY CustomerID"
        Else
            tsql += "ORDER BY " + sortColumns
        End If
        Dim da As New SqlDataAdapter(tsql, cn)
        '執行過程輸出在畫面上
        HttpContext.Current.Response.Write(tsql)
        da.Fill(dt)
    End Using
    Return dt
End Function
```

C#

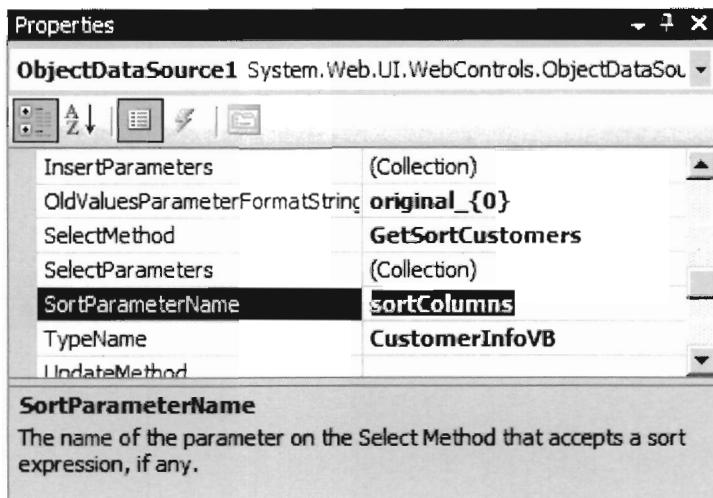
```
public DataTable GetSortCustomers(string sortColumns)
{
    DataTable dt = new DataTable();
    using (SqlConnection cn =
        new SqlConnection("Data Source=.;Initial Catalog=Northwind;Integrated Security=True"))
    {
        string tsql = "Select * from Customers ";
        if (sortColumns.Trim() == "")
        {
            tsql += "ORDER BY CustomerID";
        }
        else
        {
            tsql += "ORDER BY " + sortColumns;
        }
        SqlDataAdapter da = new SqlDataAdapter(tsql, cn);
        //執行過程輸出在畫面上
        HttpContext.Current.Response.Write(tsql);
        da.Fill(dt);
    }
}
```

5. 儲存或關閉 CustomerInfo.vb 或 CustomerInfo.cs。

6. 編譯網站，點選專案，自主選單「Build」下選取「Build Web Site」。
7. 切換到網頁的「Design」畫面。
8. 點選 ObjectDataSource1 智慧標籤中的「Configure Data Source...」，選擇「CustomerInfoVB」或是「CustomerInfoCS」做為其「Business Object」，按「Next」。
9. 挑選有 sortColumns 字串參數的方法，做為其「Select」方法，按「Next」→按「Finish」。



10. 點選 ObjectDataSource1 在屬性視窗中設定 SortParameterName 為 sortColumns。

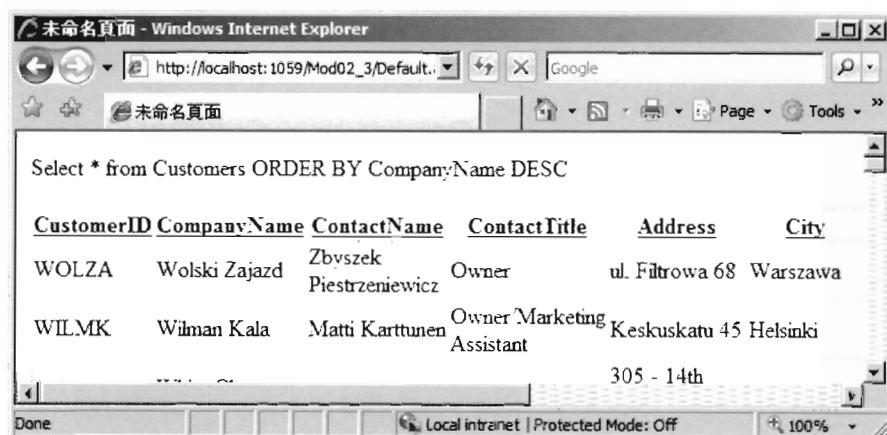


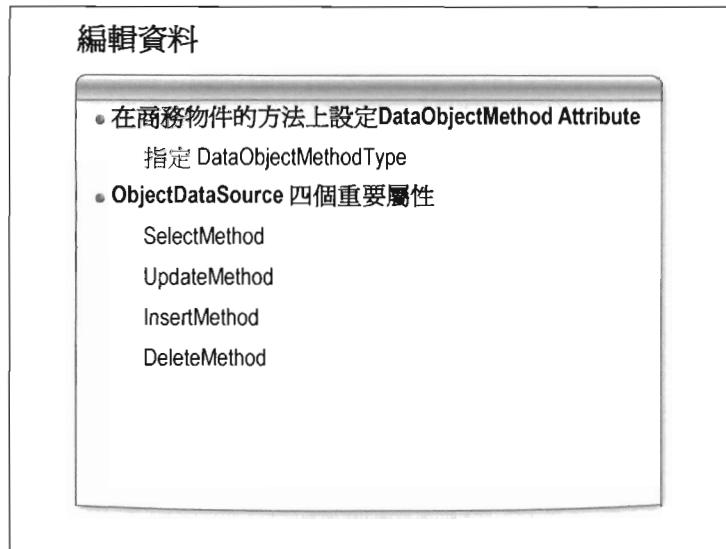
11. 設定 GridView1 控制項的 AllowSorting 屬性為 True。

```
<asp:GridView ID="GridView1" runat="server"
    DataSourceID="ObjectDataSource1"
    AllowSorting="True"

```

12. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。當網頁執行時，點選欄位排序客戶資料：





## DataObjectMethod Attribute 標籤

使用 DataObjectMethodAttribute 可以用來把資料元件中的方法做分類，例如：哪一個方法是用來做更新，那一個是用來做刪除？

ObjectDataSource 控制項等元件會檢查這個 Attribute 的值 (如果有的話)，以協助判斷在執行階段所要呼叫的資料方法。

設定 DataObjectMethodAttribute 時會指定 DataObjectMethodType 列舉型別，DataObjectMethodType 列舉型別的值有：

- Fill：表示這是將資料填入 DataSet 物件的運作方式。
- Select：表示這是擷取資料的運作方式。
- Update：表示這是更新資料的運作方式。
- Insert：表示這是插入資料的運作方式。
- Delete：表示這是刪除資料的運作方式。

## ObjectDataSource 編輯資料

ObjectDataSource 控制項包含四個重要屬性：

- SelectMethod：指定資料元件的資料查詢方法。

- UpdateMethod：指定資料元件的資料更新方法。
- InsertMethod：指定資料元件的資料新增方法。
- DeleteMethod：指定資料元件的資料刪除方法。

透過這四個屬性，便可對底層的資料進行新增、修改、刪除、查詢的操作。

若要讓 ObjectDataSource 擁有編輯資料的功能，可參考如下方法，在資料元件 CustomerInfo 類別加入 UpdateCustomer、DeleteCustomer 方法：

```
Visual Basic
Public Shared Sub UpdateCustomer (ByVal CustomerId As String,
ByVal CompanyNameName As String)
    '更新資料...
End Sub

Public Shared Sub DeleteCustomer(ByVal CustomerId As String)
    '刪除資料...
End Sub
```

```
C#
public static void UpdateCustomer(string CustomerId,
        string CompanyNameName)
{
    //更新資料...
}

public static void DeleteCustomer(string CustomerId)
{
    //刪除資料...
}
```



## 透過 GridView 整合編輯

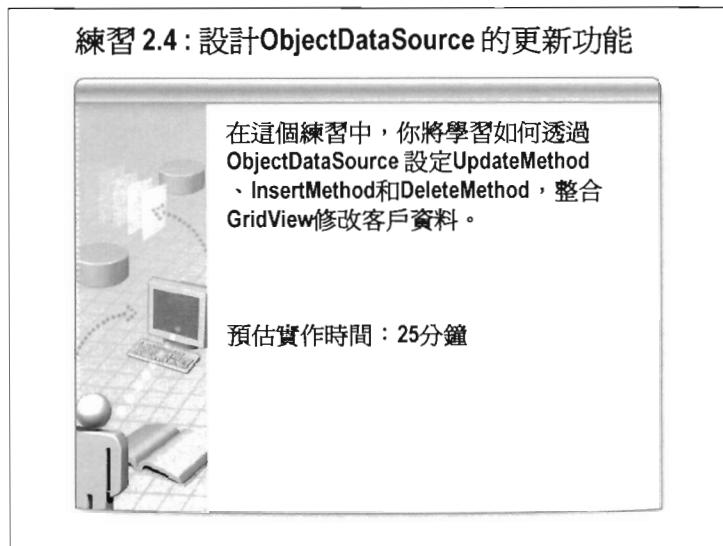
若要讓使用者可編輯資料，GridView 也要跟著調整，  
ShowEditButton、ShowDeleteButton 設為 True，以便顯示編輯(Edit)及  
刪除(Delete)按鈕。

## 指定 GridView 欲顯示的欄位

GridView 的「Auto-generate fields」選項會自動建立取得的資料欄位，若要自訂可清除此選項，自行加入欲顯示的欄位。

## 設定主索引鍵

即使 GridView 上沒有顯示主索引鍵，也要設定 GridView 控制項的 DataKeyNames 屬性設為主索引鍵欄位，以便對應到 ObjectDataSource 所需要的關鍵資料。



## 練習 2.4 : 設計 ObjectDataSource 的更新功能

### 目的：

在這個練習中，將學習如何透過 ObjectDataSource 設定 UpdateMethod 、InsertMethod 和 DeleteMethod ，整合 GridView 修 改客戶資料。

### 功能描述：

在練習中，將在原本的 CustomerInf 類別，加入二個方法分別對 Northwind 資料庫的 Customers 客戶資料進行修改及刪除。

### 預估實作時間：25 分鐘

### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「Open Web Site」→選取「ASP.NET Web Site」→選擇「File System」設定 Folder 選取「\U9544\Practices\VB 或 CS\Mod02\_4\Starter\Mod02\_4」目錄，點選「Open」。

3. 打開 App\_Code 中子目錄(VB 或是 CS)下的 CustomerInfo.vb 或是 CustomerInfo.cs。
4. 在類別中加入一個名為 UpdateCustomer 的靜態方法，它需要三個字串參數：CustomerID、CompanyName、Country，透過這三個參數可依 CustomerID 更新客戶資料。並且透過 DataObjectMethod 屬性表示此方法為更新的行為，程式碼如下：

Visual Basic

```
<System.ComponentModel.DataObjectMethod(Component
Model.DataObjectType.Update)> _
Public Shared Sub UpdateCustomer(ByVal CustomerID As String, B
yVal CompanyName As String, ByVal Country As String)
    Using cn As New SqlConnection("Data Source=.;Initial Catalog=Northwind;Integrated Security=True")
        Dim updateString As String = "UPDATE Customers SET C
ompanyName=@CompanyName, Country=@Country WHERE Cust
omerID=@CustomerID"
        Dim cmd As New SqlCommand(updateString, cn)
        cmd.Parameters.AddWithValue("@CompanyName", Comp
anyName)
        cmd.Parameters.AddWithValue("@Country", Country)
        cmd.Parameters.AddWithValue("@CustomerID", Custome
rID)
        cn.Open()
        cmd.ExecuteNonQuery()
        cn.Close()
    End Using
End Sub
```

C#

```
[System.ComponentModel.DataObjectMethod(System.Com
ponentModel.DataObjectType.Update)]
public static void UpdateCustomer(string CustomerID,
    string CompanyName, string Country)
{
    using (SqlConnection cn =
        new SqlConnection("Data Source=.;Initial Catalog=Northwi
nd;Integrated Security=True"))
    {
        string updateString = "UPDATE Customers SET CompanyN
ame =@CompanyName, Country=@Country WHERE CustomerID=
@CustomerID";
        SqlCommand cmd = new SqlCommand(updateString, cn);
        cmd.Parameters.AddWithValue("@CompanyName", Compa
nyName);
        cmd.Parameters.AddWithValue("@Country", Country);
        cmd.Parameters.AddWithValue("@CustomerID", CustomerI
D);
```

```

        cn.Open();
        cmd.ExecuteNonQuery();
        cn.Close();
    }
}

```

5. 再加入一個名為 DeleteCustomer 的靜態方法，它需要一個字串參數：CustomerID，透過這個參數可依 CustomerID 刪除客戶資料，並且透過 DataObjectMethod 屬性表示此方法為刪除的行為。程式碼如下：

Visual Basic

```

<System.ComponentModel.DataObjectMethod(Component
Model.DataObjectType.Delete)>
Public Shared Sub DeleteCustomer(ByVal CustomerID As String)
    Using cn As New SqlConnection("Data Source=.;Initial Catalog=Northwind;Integrated Security=True")
        Dim deleteString As String = "DELETE Customers WHERE CustomerID =@CustomerID "
        Dim cmd As New SqlCommand(deleteString, cn)
        cmd.Parameters.AddWithValue("@CustomerID", CustomerID)
        cn.Open()
        cmd.ExecuteNonQuery()
        cn.Close()
    End Using
End Sub

```

C#

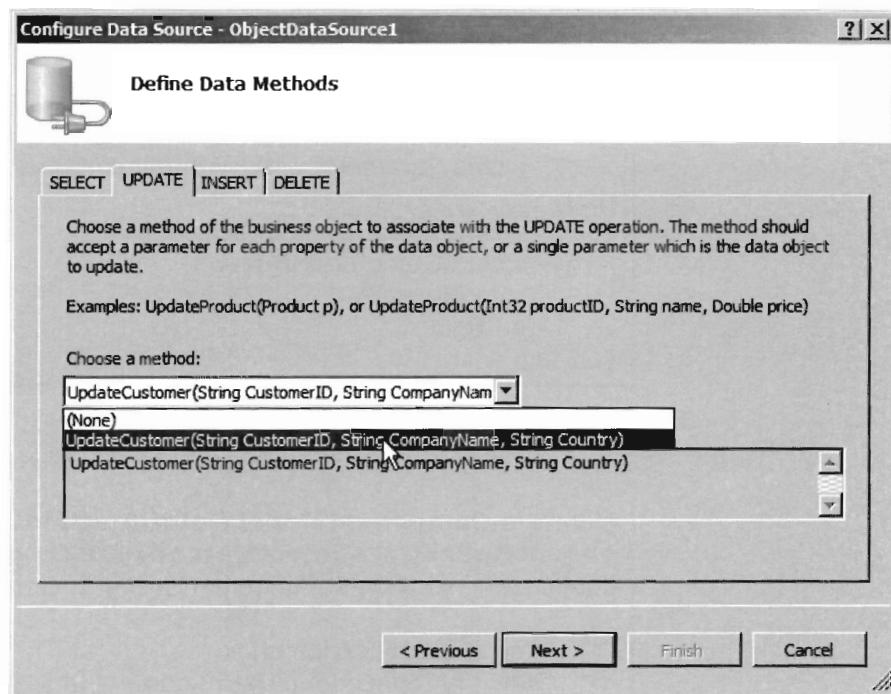
```

[System.ComponentModel.DataObjectMethod(System.Com
ponentModel.DataObjectType.Delete)]
public static void DeleteCustomer(string CustomerID)
{
    using (SqlConnection cn =
        new SqlConnection("Data Source=.;Initial Catalog=Northwin
d;Integrated Security=True"))
    {
        string deleteString = "DELETE Customers WHERE Custome
rID =@CustomerID ";
        SqlCommand cmd = new SqlCommand(deleteString, cn);
        cmd.Parameters.AddWithValue("@CustomerID", CustomerI
D);
        cn.Open();
        cmd.ExecuteNonQuery();
        cn.Close();
    }
}

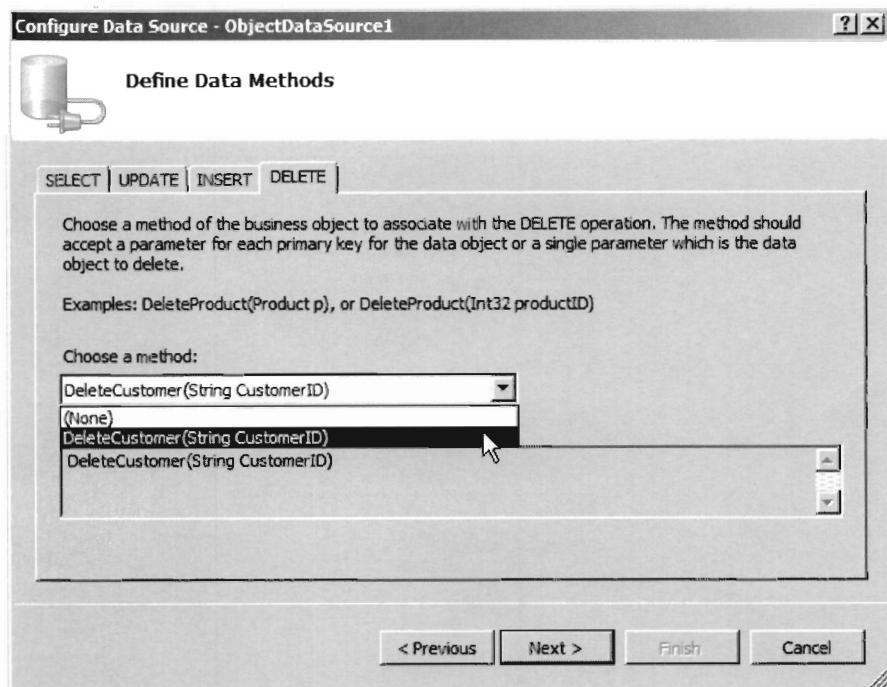
```

6. 儲存或關閉 CustomerInfo.vb 或 CustomerInfo.cs

7. 編譯網站，點選專案，自主選單「Build」下選取「Build Web Site」。
8. 切換到網頁的「Design」畫面。
9. 點選 ObjectDataSource1 智慧標籤中的「Configure Data Source...」，選擇「CustomerInfoVB」(C#語言請選擇「CustomerInfoCS」)做為其「Business Object」，按「Next」按鈕。
10. 增加「UpdateCustomer」做為其「Update」方法，按下「Next」。



11. 「DeleteCustomer」做為其「Delete」方法，按下「Next」。



12. 點選 GridView 智慧標籤，勾選「Enable Editing」及「Enabled Deleting」啓用編輯和刪除的功能。

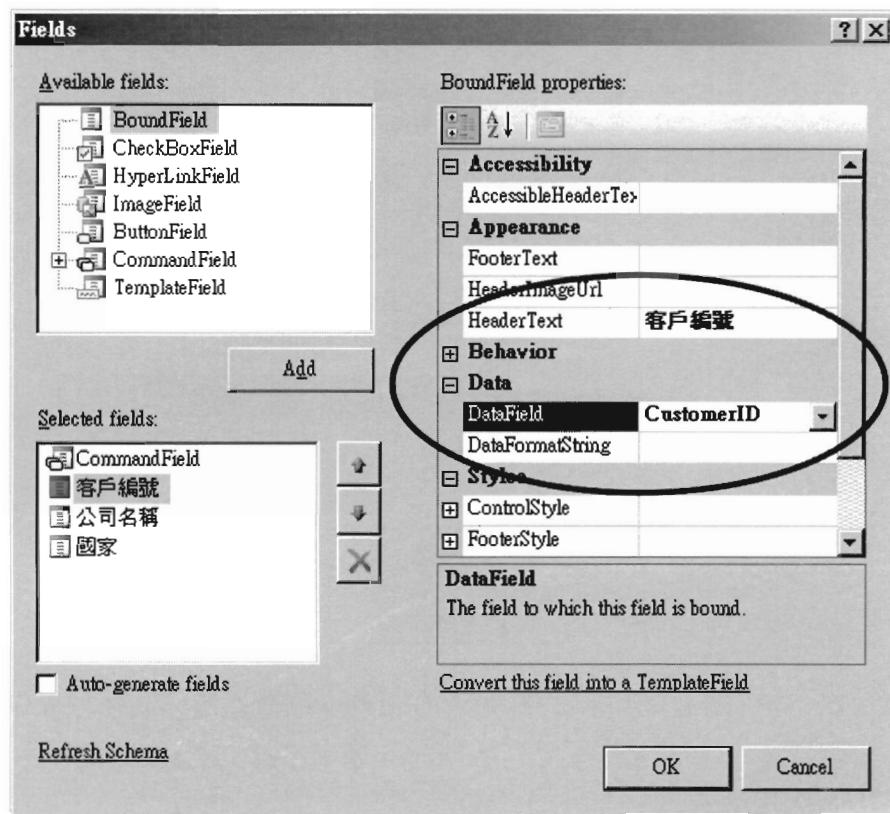
	Databound Col0	Databound Col1	Databound Col2
<a href="#">Edit</a> <a href="#">Delete</a> abc	0	abc	
<a href="#">Edit</a> <a href="#">Delete</a> abc	1	abc	
<a href="#">Edit</a> <a href="#">Delete</a> abc	2	abc	
<a href="#">Edit</a> <a href="#">Delete</a> abc	3	abc	
<a href="#">Edit</a> <a href="#">Delete</a> abc	4	abc	

ObjectDataSource - ObjectDataSource1

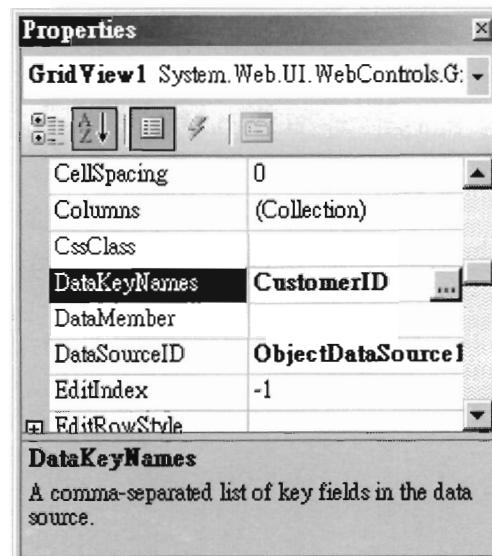
13. 點選「GridView」智慧標籤中的「Edit Columns..」。清除「Auto-generate fields」選項。

14. 加入三個「BoundField」，將其「HeaderText」分別設為：客戶編號、公司名稱、國家；「DataField」分別輸入：

CustomerID、CompanyName、Country。



15. 在「Properties」屬性視窗將「GridView」的  
「DataKeyNames」輸入「CustomerID」。



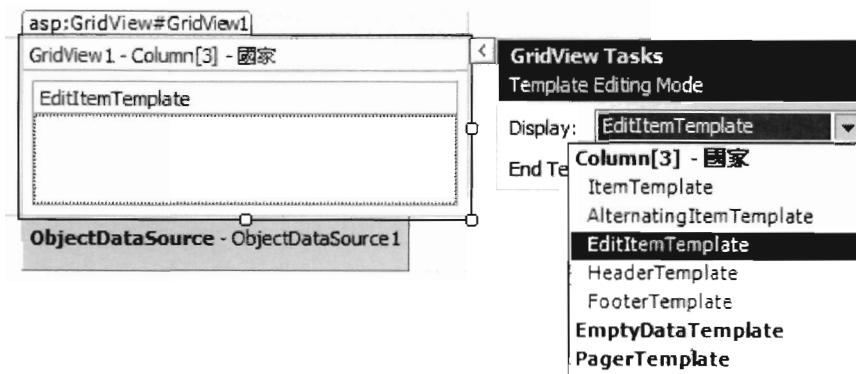
16. 最後切換到 Source 原始檔視窗中，找到  
<asp:ObjectDataSource>控制項，請將<asp:ObjectDataSource>

控制項中設計工具自動產生的  
OldValuesParameterFormatString="original\_{0}"屬性刪除。

17. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。當網頁執行時，點選「Edit」更新客戶資料，或是點選「Delete」，刪除客戶資料：

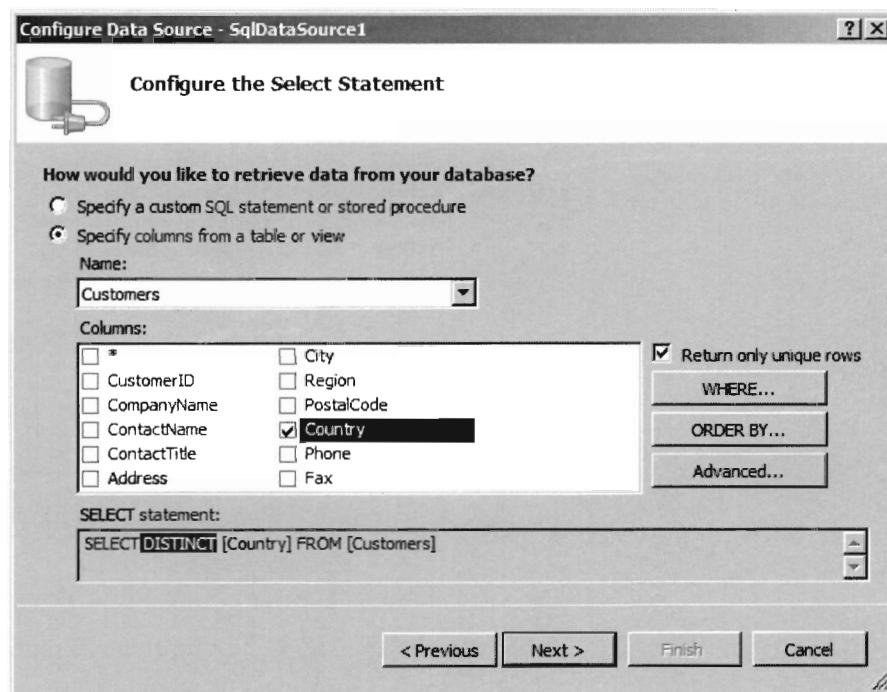


18. 回到 Visual Studio 2008，在 GridView1 智慧標籤，點選「Edit Columns...」，在視窗的左下角「Selected fields:...」中選取「國家」，並按下「Convert this field into a TemplateField」，
19. 在 GridView1 智慧標籤，點選「Edit Templated」後，在 Display: 中挑選「國家」的 EditItem Template，把 EditItem Template 中原本的 TextBox 控制項刪除。

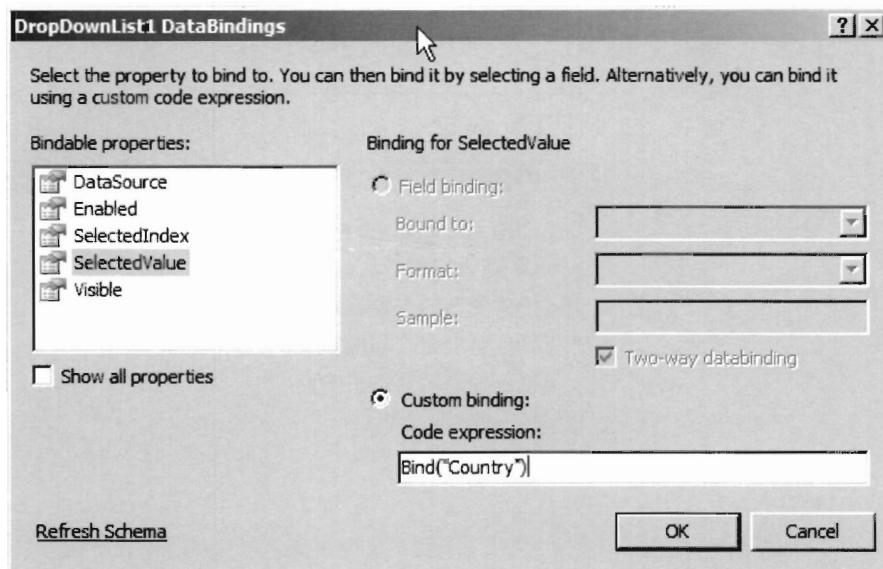


20. 從「Toolbox」工具箱中拖拉一個 DropDownList 控制項到 EditItem Template 視窗中取代刪除的 TextBox 控制項。

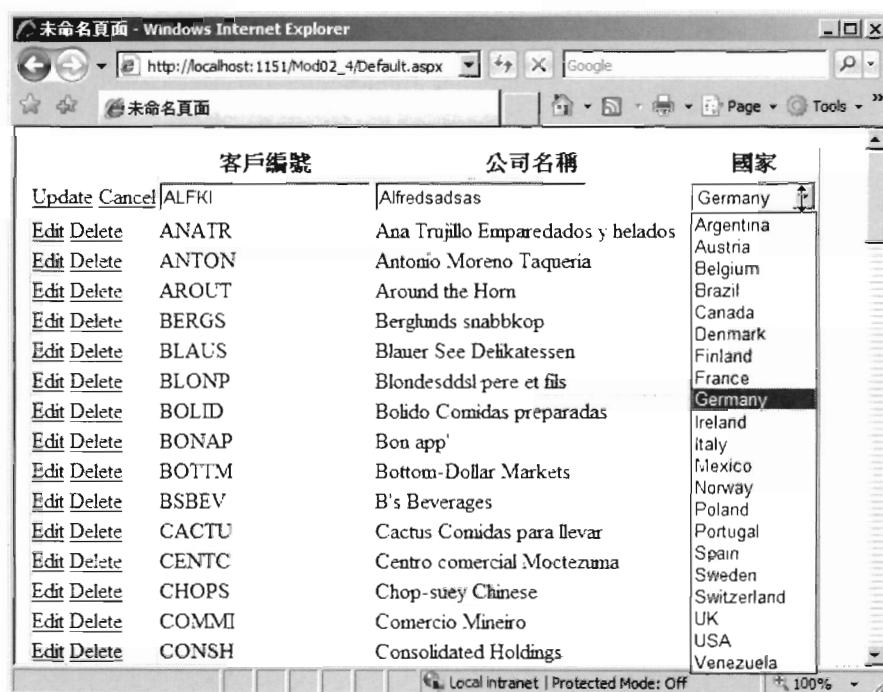
21. 點選 DropDownList 控制項的智慧型標籤，選取「Choose Data Source」，從「Choose Data Source」下拉式清單方塊，選取「<New data source>」，選取「Database」當作資料來源，按下「OK」。
22. 按「Next」連結到 Northwind 資料庫，直到「Configure the Select Statement」視窗，選取查詢 Customers 資料表的 Country 欄位，並把「Return only unique rows」選項勾選起來。

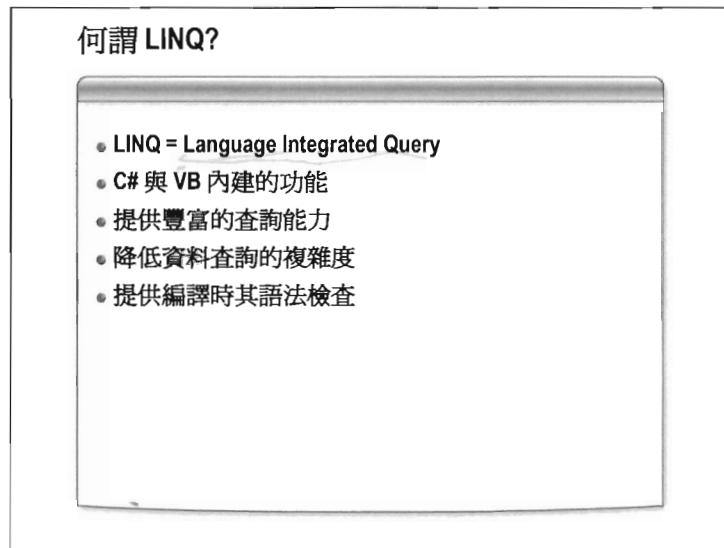


23. 按「Next」直到精靈結束。
24. 回到 DropDownList 控制項的資料來源選取視窗，設定 display 為 Country 以及 value 為 Country，按下「OK」。
25. 點選 DropDownList 控制項智慧標籤的「Edit DataBindings...」，設定 SelectedValue 的 Code expression 為 Bind("Country")，按下「OK」。



26. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」，確認程式執行結果，在畫面中按下「Edit」，可見國家名稱可供挑選。參考下圖所示：



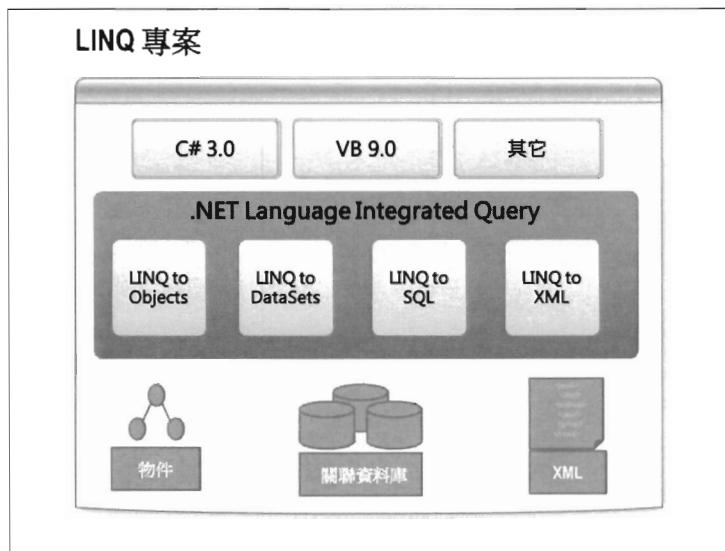


## 何謂 LINQ

LINQ (Language Integrated Query) 是.NET Framework 3.5 中絕對不能忽視的新成員，它的目標是以一致的方式，直接利用程式語言本身的語法存取各種不同類型的資料。LINQ 的全名為 Language Integrated Query，直接解釋，就是與程式語言整合在一起的語法。

因為 LINQ 強大而具彈性的設計技術，以下為 LINQ 所具備的優點：

- C# 與 Visual Basic 內建的功能
- 提供豐富的查詢能力
- 降低資料查詢的複雜度
- 提供編譯時其語法檢查



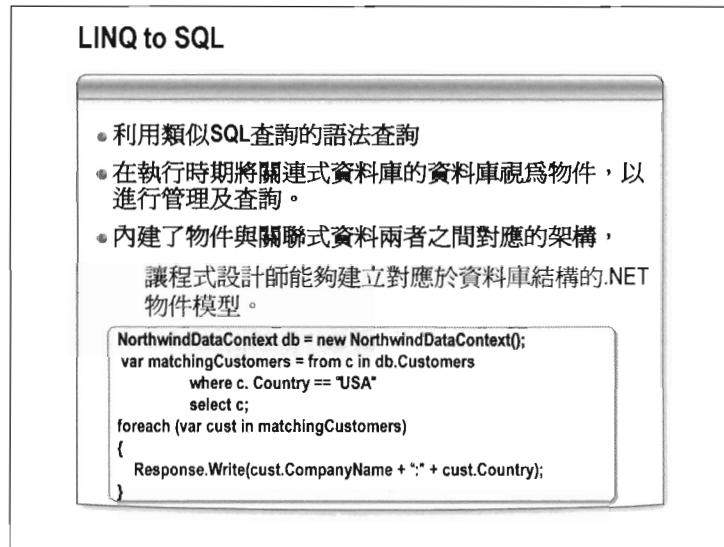
## LINQ 專案類型

以目前的情況為例，寫程式不是只要熟悉程式語言即可，面對關聯式資料庫得用 SQL 語法、面對 XML 得再學另一套查詢 XML 的 XPath 語法。這些不同的語法，讓程式開發人員在處理不同類型資料時，形成許多大大小小的障礙。

而微軟推出 LINQ 的目的就是為了統一各種類形資料的存取語法，讓它們在程式中都可用一致的方式查詢、新增、刪除、修改。

LINQ 包含以下部份：

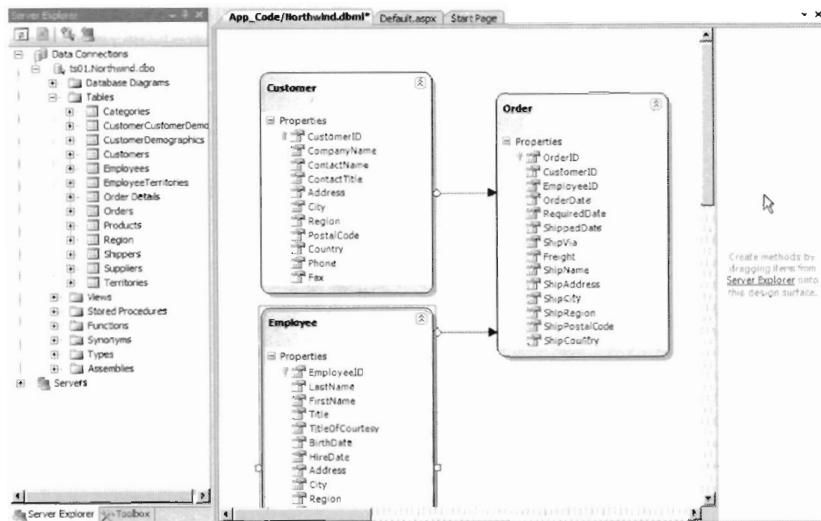
- LINQ to Object：可查詢集合與陣列中的資料
- LINQ to DataSet：查詢 DataSet/DataTable 中的資料。
- LINQ to SQL：查詢或異動微軟 SQL Server 資料庫的資料。
- LINQ to XML：查詢 XML 節點。



## LINQ To SQL

在 Visual Studio 2008 中新增一個圖形化設計工具：「Object Relational Designer」又簡稱為 O/R Designer，「O/R Designer」提供一個圖形化的方式，讓程式設計師描繪資料表的內容及資料表之間的關聯性。

透過這張設計圖，將會產生一個可以在實體類別和資料庫之間傳送和接收資料的強型別 (Strongly Typed) 的 DataContext，以用來在實體類別與資料庫之間傳送和接收資料。



## DataContext 類別

DataContext 類別是一個 LINQ to SQL 類別，可以做為 SQL Server 資料庫與該資料庫對應之 LINQ to SQL 實體類別之間的管道。

「O/R Designer」除會建立 DataContext 類別外，另外還會為資料表建立對應的類別，這些類別稱為「Entity 類別」。每個資料表會對應成一個「Entity 類別」，而資料表中的欄位就對應成「Entity 類別」的屬性，資料表中的每筆資料就是「Entity 類別」的實例。

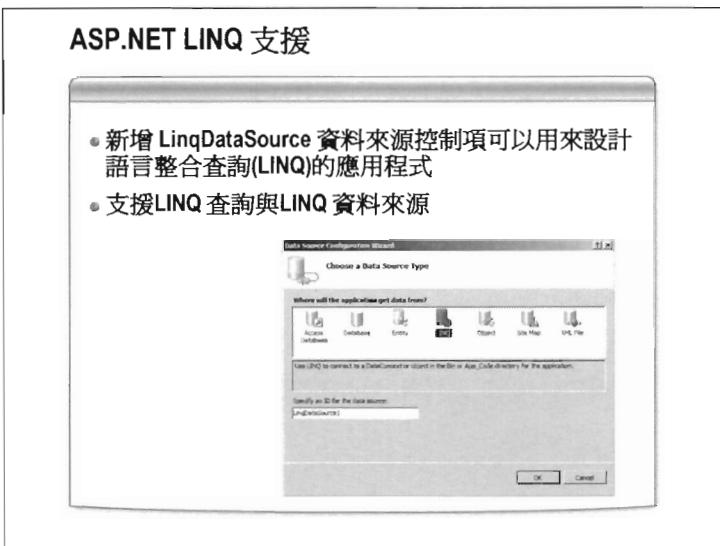
以下程式碼為建立 DataContext 實體物件，並透過屬性方式取出對應於資料庫中資料表資料的 LINQ 語法：

```
NorthwindDataContext db = new NorthwindDataContext();

var matchingCustomers = from c in db.Customers
                      where c.Country == "USA"
                      select c;

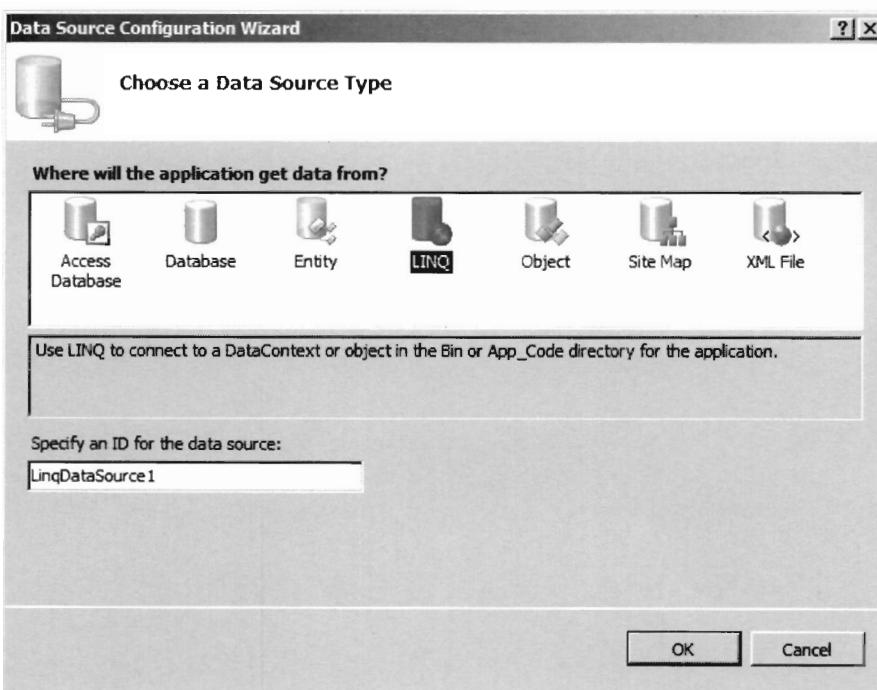
foreach (var cust in matchingCustomers)
{
    Response.Write(cust.CompanyName + ":" + cust.Country);
}
```

簡單來說，「O/R Designer」的工作就是將關聯式資料庫，轉成一個  
DataContext 物件模型，以便在程式中用 LINQ 語法存取。



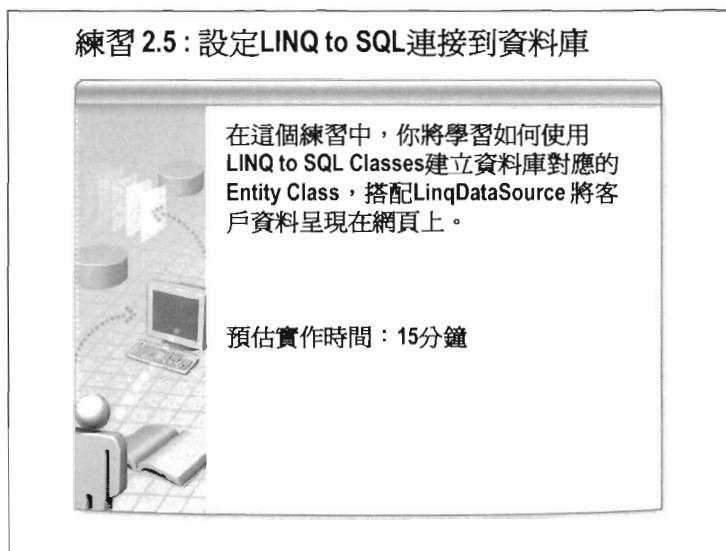
## ASP.NET 中的 LinqDataSource

LinqDataSource 控制項透過 ASP.NET 資料來源控制項架構，將 Language-Integrated Query (LINQ) 開放給 Web 開發人員使用。



在網頁上加入 LinqDataSource 控制項，透過 LinqDataSource 控制項可連接來自資料庫或記憶體中資料集合(如陣列)的資料。另外也可以設定 LinqDataSource 控制項來更新、插入及刪除資料。這樣開發者可以不必額外撰寫 SQL 命令來執行資料處理這些工作，自動的交由控制項處理就可以了。

而在網頁上使用 LinqDataSource 控制項時，LinqDataSource 控制項並不會直接連接到資料庫，而是設定對應到「Entity 類別」。



## 練習 2.5: 設定 LINQ to SQL 連接到資料庫

### 目的：

在這個練習中，你將學習如何新增一個 LINQ to SQL Classes，並透過 O/R Designer 建立資料庫對應的 DataContext 物件，並搭配 LinqDataSource 以及 ListView 控制項 將客戶資料呈現在網頁上。

### 功能描述：

這個練習中會先透過 LINQ to SQL Classes 建立 Northwind 資料庫所對應的 DataContext 物件，在網頁上再透過設定 LinqDataSource 將資料來源指定為 DataContext 物件，以便擷取資料。

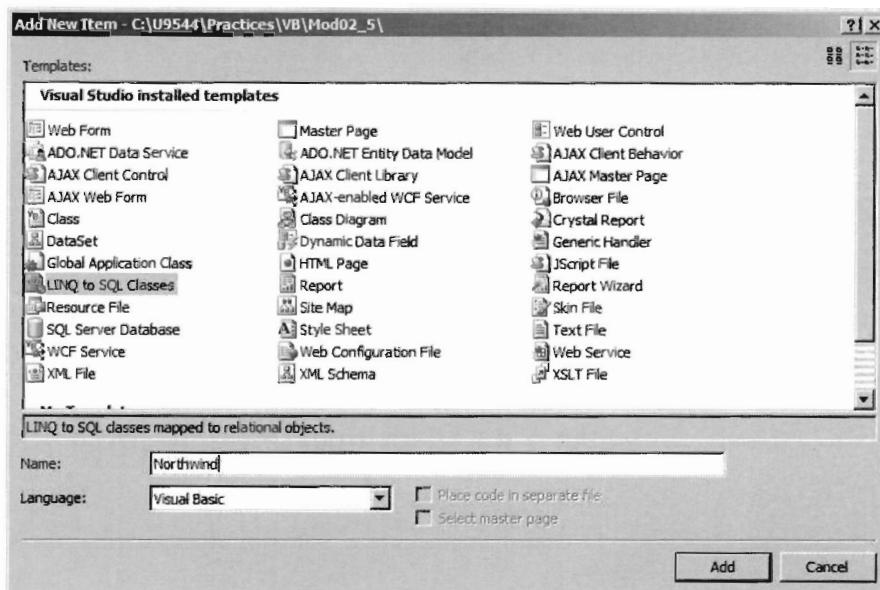
### 預估實作時間：15 分鐘

### 實作步驟：

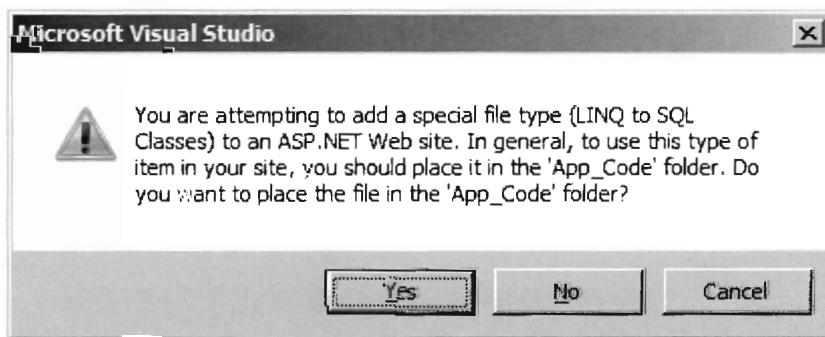
1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod02\_5\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod02\_5」。

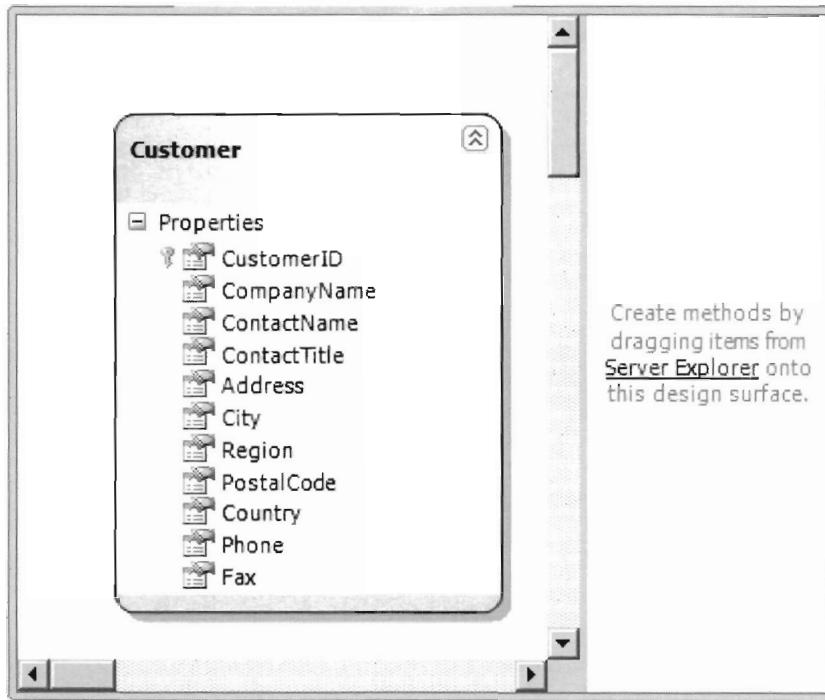
3. 自主選單「Web Site」下「Add New Item...」，選取「LINQ To SQL Classes」，設定 Name 為「Northwind」，與使用的程式語言 (Language)，如 Visual Basic 或 C#，按下「Add」。



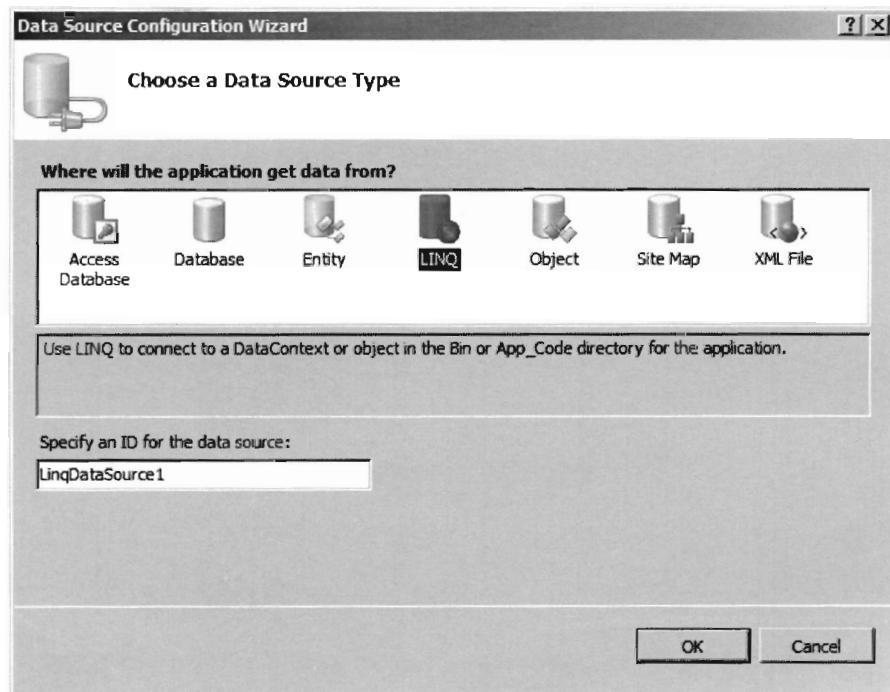
4. 這時會彈出一個視窗，詢問是否將檔案放在 App\_Code 中，請按下「Yes」。



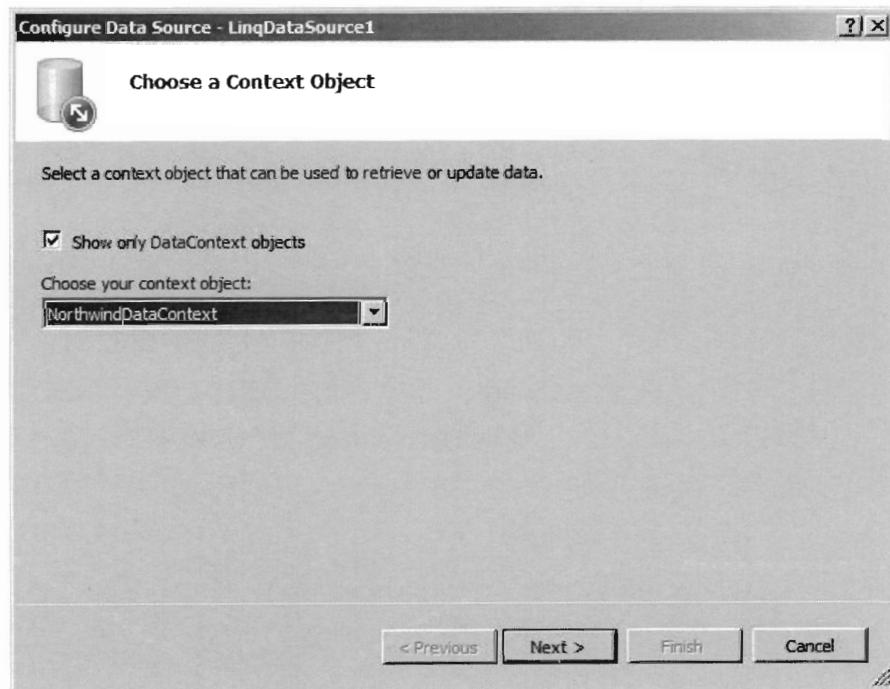
5. 點選 Northwind.dbml 的設計介面窗中的 Server Explorer 開啓「Server Explorer」視窗，新增一個 Data Connection，設定連結到 Northwind 資料庫。
6. 將 Northwind 的 Customers 資料表直接拖拉到 Northwind.dbml 的設計介面中，Visual Studio 2008 將會自動產生對應的 Entity 類別。



7. 編譯網站，點選專案，自主選單「Build」下選取「Build Web Site」。
8. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
9. 從「Toolbox」工具箱中拖拉一個 ListView 控制項到網頁中。
10. 點選 ListView 控制項的智慧型標籤，從「Choose Data Source」下拉式清單方塊，選取「<New data source>」，選取 LinqDataSource 當作資料來源，按下「OK」。

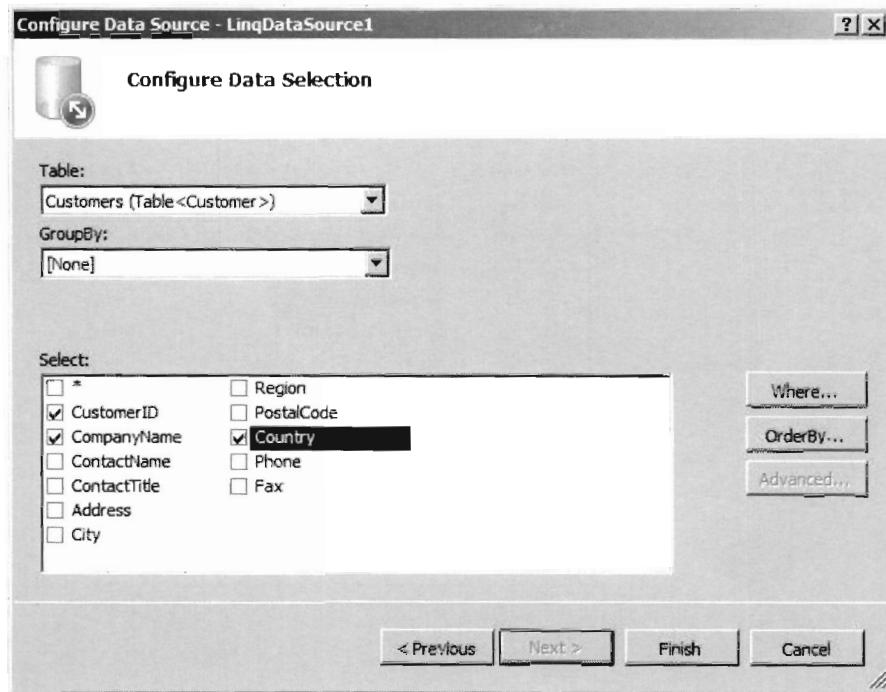


11. 選擇資料來源為「NorthwindDataContext」。



12. 到「Configure the Select Statement」視窗，選取查詢

Customers 資料表的 CustomerID、CompanyName 與 Country  
三個欄位：



13. 按「Next」直到精靈結束。
14. 點選 ListView 控制項的智慧型標籤，點選「Configure ListView」，設定 Layout 為 Grid。



15. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。執行畫面參考：

The screenshot shows a Windows Internet Explorer window displaying a table of customer information. The table has three columns: CustomerID, CompanyName, and Country. The data is as follows:

CustomerID	CompanyName	Country
ALFKI	Alfreds Futterkiste	Germany
ANATR	Ana Trujillo Emparedados y helados	Mexico
ANTON	Antonio Moreno Taqueria	Mexico
AROUT	Around the Horn	UK
BERGS	Berglunds snabbköp	Sweden
BLAUS	Blauer See Delikatessen	Germany
BLONP	Blondesððsl pere et fils	France
BOLID	Bólido Comidas preparadas	Spain
BONAP	Bon app'	France
BOTTM	Bottom-Dollar Markets	Canada

## 總結

- 分散式架構簡介
- 設計資料元件 ObjectDataSource
- 資料元件的異動
- 設計LINQ to SQL
- 使用LinqDataSource

# 第三章：網頁多國語言 程式設計

## 本章大綱

全球化與當地語系化.....	4
語言喜好設定 .....	5
取得用戶端選用語系.....	6
設定語系 .....	7
練習 3.1 : 判斷用戶端語言喜好設定 .....	9
自動化取得用戶端選用語系.....	13
練習 3.2 : 用戶端設定語言喜好設定 .....	14
網頁當地語系化(Localization) .....	17
建立 Local 資源檔.....	20
當地語系化運算式 .....	21
練習 3.3 : 使用資源檔 .....	23
使用程式取回 Local 資源.....	26
建立 Global 資源檔 .....	27
使用程式存取 Global 資源檔.....	29
練習 3.4 : 使用 Global 資源檔 .....	31
明確指定資源檔繫結屬性 .....	33
使用 Localize 控制項 .....	34
練習 3.5 : 明確資源檔繫結與 Localize 控制項.....	35
總結 .....	38

作者：  
許薰尹





---

## 大綱

- 什麼是當地語系化?
- 設定語言喜好設定
- 取得用戶端選用語系
- 網頁當地語系(Localization)

    認識資源檔

    如何建立Local 資源檔

    如何建立Global 資源檔

- 總結

---

國際化的網頁日趨重要，ASP.NET 支援多國語言網頁，本章將介紹多國語言網頁的設計。

在這個章節中將學習到：

- 什麼是全球化?
- 什麼是當地語系化?
- 設定語言喜好設定
- 取得用戶端選用語系
- 網頁當地語系化(Localization)
  - 認識資源檔
  - 如何建立 Local 資源檔
  - 如何建立 Global 資源檔

### 全球化與當地語系化

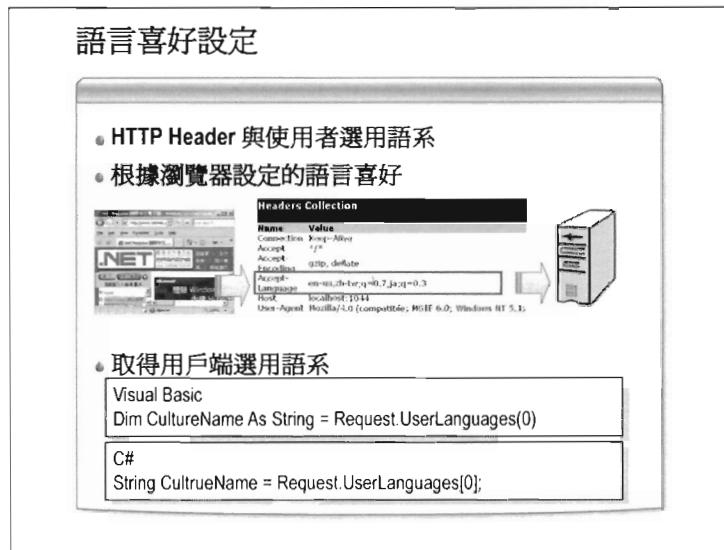
- **全球化**  
    設計、開發適用於多種文化的應用程式之過程
- **當地語系化**  
    網頁必須隨著使用者語系選擇，顯示該地區文化特性
- **ASP.NET 簡化當地語系化設計的複雜性**  
    自動偵測瀏覽器使用語系  
    控制項屬性可以繫結到資源檔  
    程式使用強型別存取資源  
    自動產生及關聯到資源檔  
    支援在設計階段建立資源檔

## 全球化與當地語系化

講到多國語言便會談到兩個名詞：全球化(Globalization)與當地語系化(Localization)。全球化是指設計、開發適用於多種文化的應用程式之過程。當地語系化指的是特定的文化特性 (Culture) 和地區設計應用程式的過程。

在 ASP.NET 基本架構的設計已包含多國語言的功能，開發人員大多數只需使用設定的方式便可完成多國語言網頁的設計，它包含了以下功能：

- 自動偵測瀏覽器使用語系
- 控制項屬性可以繫結到資源檔
- 程式使用強型別存取資源
- 自動產生及關聯到資源檔
- 支援在設計階段建立資源檔



## 語言喜好設定

當地語系化(Localization)就是網頁或應用程式必須依據使用者的語言喜好來顯示該地區的語言及文化特性(例如日期、時間、金額、數字的顯示格式)。多國語系網頁設計的第一步，必須先取得使用者的語言喜好設定，這一節中將學會如何在 IE 設定語言喜好設定，以及 ASP.NET 中判斷使用者選用語系。

當使用者進行 Request 動作時，IE 的語言喜好設定會存放在 HTTP 封包表頭(HTTP Header)的 Accept-Language 值傳送到伺服器端。伺服器端的程式可以從 HTTP Header 中的 Accept-Language 值中獲取語言喜好設定。

### 設定語言喜好設定

使用者可以利用 IE 的「網際網路選項」設定語言喜好設定，設定步驟如下：

1. 開啓 Internet Explorer。
2. 選單上選取「Tools」、「Internet Options」。

在「Internet Options」的「General」頁籤點選「Language」按鈕。

3. 在「Language Preference」畫面，按「Add」按鈕加入喜好語言。
4. 可點選「Move Up」、或「Move Down」按鈕調整喜好語言的順序。
5. 可點選「Remove」刪除某個語言。

## 取得用戶端選用語系

Request 物件的 UserLanguages 屬性集合讀取 HTTP Header 中的 Accept-Language 值。在程式中取出 UserLanguages 屬性集合索引 0，代表取出優先權最高的語系。

```
Visual Basic  
Request.UserLanguages(0)
```

```
C#  
Request.UserLanguages[0];
```



## 設定語系

若要設定網頁使用的語系可以依據瀏覽器的語言喜好設定或利用程式的方式建立 CultureInfo 物件指定特定語系。在 ASP.NET Web 網頁中，您可以利用 Page 指示詞或程式設定兩個文化特性值：Culture 和 UICulture。Culture 用來設定與文化特性相關功能，如日期、數字和貨幣格式等)。UICulture 用來決定網頁載入的資源。

使用程式碼自訂文化設定必須引用以下二個命名空間：

- System.Globalization：包含多種全球化程式所需的類別，其中常見的 CultureInfo 類別便是設定某文化資訊的類別。
- System.Threading：System.Threading 命名空間則是包含一些關於執行緒等類別像是 Thread 類別。

### 套用瀏覽器語言喜好的語系

下列範例是使用 Request 的 UserLanguages 屬性的索引 0 判斷瀏覽器是否有傳送語系到伺服器，如果有就記錄使用語系到目前執行緒的 CurrentCulture 屬性。

Visual Basic

```
If (Request.UserLanguages(0) IsNot Nothing) Then
    Thread.CurrentThread.CurrentCulture =
        CultureInfo.CreateSpecificCulture(
            Request.UserLanguages(0))
End If
```

C#

```
if (Request.UserLanguages[0] != null)
{
    Thread.CurrentThread.CurrentCulture =
        CultureInfo.CreateSpecificCulture(
            Request.UserLanguages[0]);
}
```

## 使用 **CultureInfo** 物件指定特定語系

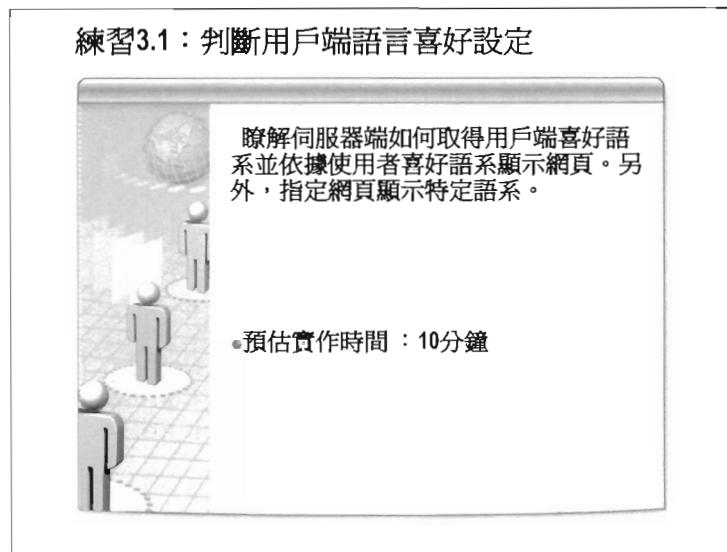
在程式中設定其他語系的方式非常簡單，只要將目前執行緒的文化資訊指定為某地方語系即可。此範例將套用日本地區文化慣例，程式如下：

Visual Basic

```
Thread.CurrentThread.CurrentCulture = New CultureInfo("ja-JP")
```

C#

```
Thread.CurrentThread.CurrentCulture = new CultureInfo("ja-JP");
```



## 練習 3.1：判斷用戶端語言喜好設定

目的：

瞭解伺服器端如何取得用戶端喜好語系並依據使用者喜好語系顯示網頁。另外，指定網頁顯示特定語系。

功能描述：

網頁顯示目前伺服器端語系、用戶端語系以及利用 CultureInfo 類別指定文化資訊。

預估實作時間：10 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「Open」→「Web Site」→選取「磁碟:\U9544\Practices\VB 或 CS\Mod03\_1\Starter\Mod03\_1」。

3. 從「Solution Explorer」打開「Default.aspx」，在頁面最上方加上必要的命名空間。

```
<%@ Import Namespace="System.Globalization" %>
<%@ Import Namespace="System.Threading" %>
```

4. 找到「TODO: 取得瀏覽器端的語言」位置，其後判斷用戶端語系，並將目前語系指定為用戶端目前設定語系。程式碼如下：

```
Visual Basic
If Request.UserLanguages(0) IsNot Nothing Then
    Thread.CurrentThread.CurrentCulture =
        CultureInfo.CreateSpecificCulture(Request.UserLanguages
(0))
End If
```

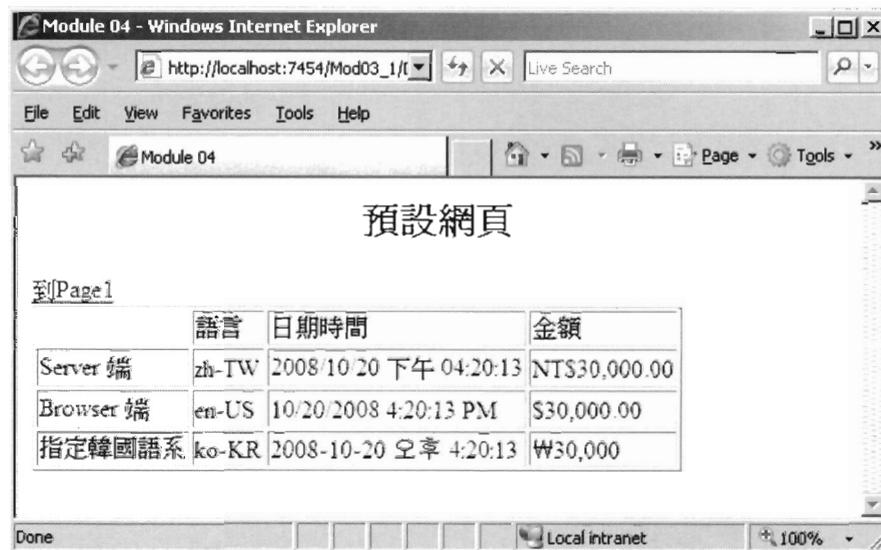
```
C#
if (Request.UserLanguages[0] != null)
{
    Thread.CurrentThread.CurrentCulture =
        CultureInfo.CreateSpecificCulture(Request.UserLanguages
[0]);
}
```

5. 找到「TODO: 指定韓國語系」位置，使用 CultrueInfo 類別指定為韓國文化資訊。

```
Visual Basic
Thread.CurrentThread.CurrentCulture = New CultureInfo("ko-KR")
```

```
C#
Thread.CurrentThread.CurrentCulture = new CultureInfo("ko-KR
");
```

6. 執行「Default.aspx」，結果如下：

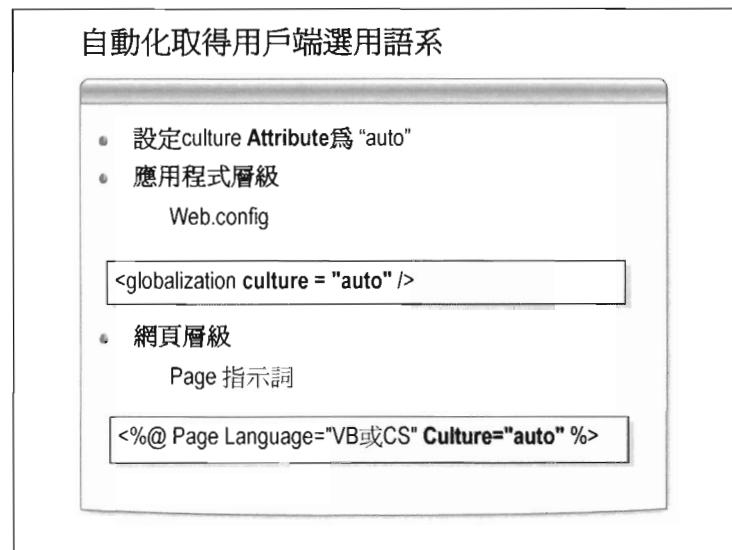


註：Server 端出現的語言是根據 Server 端控制台「地區及語言選項」而改變。

7. 開啓 IE 並設定語言喜好設定。

- 選單上選取「工具」→「網際網路選項」，
  - 在「網際網路選項」的「一般」頁籤點選「語言按鈕」。
  - 在「語言喜好設定」畫面，按「新增」按鈕加入「日文」(或任何國家)，接著點選「上移」，設定「日文」為第一優先，然後點選「確定」。
  - 關閉 IE。
8. 執行 Default.aspx，Browser 端語言應該會變成 ja-JP，以及日期、時間、金額格式都會變成日本人慣用格式。





## 自動化取得用戶端選用語系

在 ASP.NET 套用用戶端選用語系是件很容易的事，只需要在設定 Web.config 的 globalization 區段設定 **culture** Attribute 為 **auto**，或是在網頁(@Page 指示詞)的 **culture** Attribute 為 **auto** 即可。

你也可以利用程式來設定選用語系，只要使用 Request 物件的 UserLanguages 屬性集合就可以讀取 HTTP 封包的 Accept-Language 值。

設定 culture Attribute 為 "auto" 的方式又分為二種：

1. 設定在 Web.config 上可以套用到應用程式的所有頁面上：

```
<configuration>
  <system.web>
    <globalization culture = "auto" />
  </system.web>
</configuration>
```

2. 設定在@Page 指示詞上，套用在單頁上：

```
<%@ Page Language="VB或CS" Culture="auto" %>
```

### 練習3.2：用戶端設定語言喜好設定

•用戶端在IE上設定語言喜好設定，並確認網頁是否套用用戶端指定語系。

•預估實作時間：10分鐘

### 練習 3.2：用戶端設定語言喜好設定

#### 目的：

這個練習目的在學習如何使用自動化方式取得用戶端語系。

#### 功能描述：

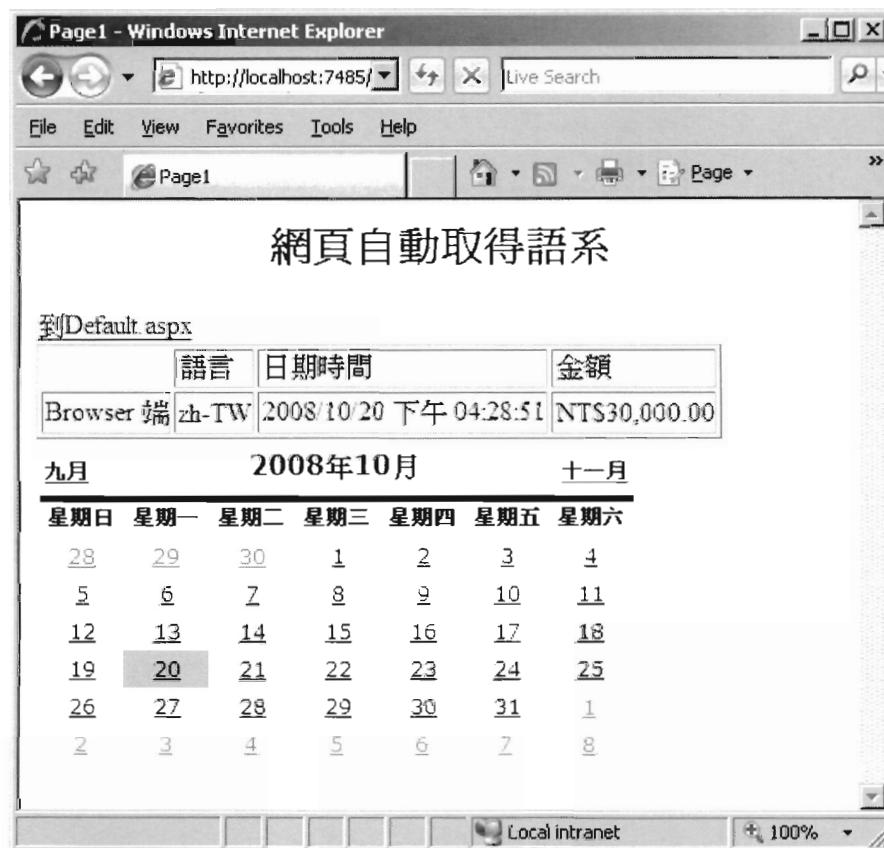
延續上一個練習，設定 IE 設定語言喜好設定並確認前一個練習的網頁是否套用指定語系。

#### 預估實作時間：10 分鐘

#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「Open」→「Web Site」→選取「磁碟:\U9544\Practices\VB 或 CS\Mod03\_2\Starter\Mod03\_1」或繼續前一個專案。

3. 打開 IE。確認目前 IE 語言喜好設定第一優先是否為「日文」，如果不是請設定「日文」。
4. 執行 Default.aspx 頁面確認畫面 Browser 端語言應該是 ja-JP 以及日期、時間、金額格式都會變成日本人慣用格式。。
5. 接著點選頁面上的「到 Page1」超連結，會看到 Page1 的內容如下圖，它目前顯示的內容是依據伺服器的語系，而不是 IE 的語言喜好設定。



因為 Page1 沒有做任何語系識別的程式碼，也沒有做任何設定，所以不會依據 IE 的語言喜好設定。

6. 從「Solution Explorer」打開「Page1.aspx」，找到<% @Page ...%>，並設定 Culture 屬性為 Auto，如下：

```
<%@Page Language="VB或CS" Culture="Auto"%>
```

7. 點選 IE 的「重新整理」，確認 Page1.aspx 是否套用 ja-JP 語系的格式(日期、時間、金額...)。



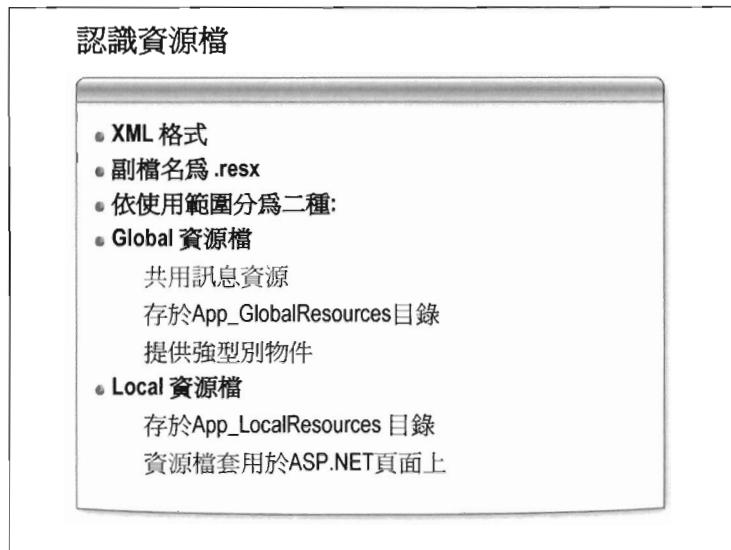
### 網頁當地語系化(Localization)

- 認識資源檔
- 如何建立Local 資源檔
- Local 資源檔的繫結屬性
- 如何建立Global 資源檔
- 程式存取Global資源檔
- 明確指定資源檔繫結屬性

### 網頁當地語系化(Localization)

當地語系化(Localization)指的是為應用程式指定特定的文化特性(Culture)和地區設定(Locale)的過程。除了日期、時間、金額格式必須符合當地文化習慣之外，網頁上的文字也必須是當地語言。若要建立一個網頁，可適用於不同國家的人都能以自己的喜好語言來閱讀的網頁。傳統上的網頁設計方式可能會為相同內容但不同語言撰寫多個網頁，不過，這個方法相當耗費維護上的負擔、容易出錯。

ASP.NET 可讓您建立根據瀏覽器的語言喜好設定，或利用程式碼來取得喜好的語言顯示的網頁內容。ASP.NET 支援資源檔 (.resx)，使用 Visual Studio 的設計工具可以輕易的編輯 Resource File，將要顯示的不同語言的內容存在不同的資源檔，在 ASP.NET 網頁中，利用控制項從資源取得其屬性值，在執行時期便可將資源檔中的不同語言組成的資源讀取出來以做顯示。



## 認識資源檔

資源檔是一個 XML 格式的檔案，一個資源檔可以記載一個語系資源，內容包含字串、路徑、圖案。資源檔之中包含索引鍵/值配對組成的資料，每一配對組成的資料都是一個個別的資源。索引鍵名稱不區分大小寫。資源檔的副檔名為.resx，檔案命名規則是：

- 「檔名.resx」代表預設語系
- 「檔名.CultureInfo 名稱.resx」代表 CultureInfo 名稱語系的資源檔

例如：系統訊息的資源檔，檔名為 SystemMessage，它們多國語言資源檔分別是：

- SystemMessage.resx，預設語系
- SystemMessage.zh-TW.resx，台灣語系
- SystemMessage.ja-JP.resx，日本語系

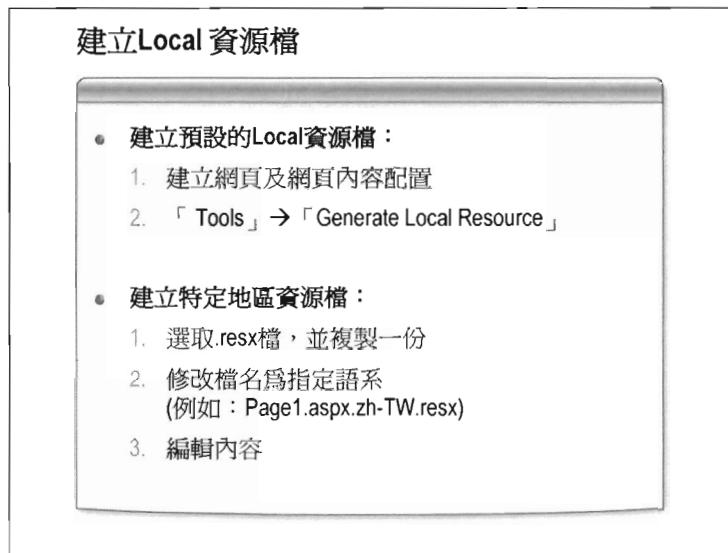
如果目前程式的語系設定是台灣地區，系統會套用 SystemMessage.zh-TW.resx 的資訊。如果是日本地區，則是套用 SystemMessage.ja-JP.resx。如果不是這二種語系，則是套用預設語

系—SystemMessage.resx。資源檔在執行階段，會將 .resx 檔動態編譯成組件來執行。

資源檔依使用方式分為二種： Global 資源檔(全域資源檔)與 Local 資源檔(本機資源檔)。

Global 資源檔存在應用程式根路徑下的 App\_GlobalResources 目錄，用於一般的系統訊息，網站中的任何網頁或程式碼都可以讀取 Global 資源檔。例如，多國語言的錯誤訊息或提示訊息。當程式要引用時它是強型別物件，這對於開發者來說非常方便，因為程式編輯器對於資源檔內的 Key 會有 IntelliSense 的功能。

Local 資源檔存在 App\_LocalResources 目錄。它是套用在網頁上，用來儲存一個 ASP.NET 網頁 (.aspx 檔)、使用者控制項(\*.ascx 檔)、或.master 等副檔名的資源。例如，Default.aspx，它的預設資源檔就是 Default.aspx.resx，台灣地區資源檔則為 Default.aspx.zh-TW.resx。和 Global 資源檔不一樣的地方是，App\_LocalResources 目錄可以位於應用程式的任何目錄中。



## 建立 Local 資源檔

Local 資源檔套用在網頁上，在 Visual Studio 中有一個自動化產生 Local 資源檔的功能。它的步驟如下：

1. 編寫網頁及網頁內容配置，例如：Page1.aspx。
2. 在選單選取「Tools」→「Generate Local Resource」。

資源檔將自動存放在 App\_LocalResources 目錄，例如：  
Page1.aspx.resx。

## 建立特定地區資源檔：

完成預設資源檔之後，便可利用它來產生其他地區資源檔。它的步驟如下：

1. Solution Explorer 選取中.resx 檔，例如：Page1.aspx.resx，並複製一份。
2. 將複製後的檔名修改為指定語系，例如：Page1.aspx.zh-TW.resx。
3. 編輯指定語系的文字。

## 當地語系化運算式

- 使用隱含式運算式語法設定繫結
    - @ Page 上指示網頁參考的資源檔
- ```
<%@ Page Culture="auto" meta:resourcekey="PageResource1"
  UICulture="auto" %>
```
- 控制項上的繫結屬性
- ```
<asp:Label ID="Label1" runat="server" Text="Name."
  meta:resourcekey="Label1Resource1"></asp:Label>
```
- 使用明確式運算式語法設定繫結
- ```
<asp:Label ID="Label5" runat="server"
  Text="<%$ Resources:Label1Resource1.Text %>"></asp:Label>
```

## 當地語系化運算式

建立 Local 資源檔之後，您可以使用隱含式運算式或明確式運算式兩種語法來存取資源檔的內容。但使用時要注意，您可以為控制項指定明確資源運算式或隱含資源運算式，但只能選擇其中一種，不可同時指定兩者。

### 使用隱含式運算式語法設定繫結

若要使用隱含當地語系化，本機資源檔中的資源用命名規則如下：

|             |
|-------------|
| 鍵值.Property |
|-------------|

例如以下描述 Button1 按鈕的 Text 屬性其在本機資源檔中的索引鍵為：

|              |
|--------------|
| Button1.Text |
|--------------|

Visual Studio 工具「Generate Local Resource」選項除了產生資源檔之外，在網頁上它也留下一些對映資源檔的資訊。

分成二個部份：

- @Page 的指示詞

@Page 指示器上多了 meta:resourcekey attribute，它是用來指示這個頁面所對映的資源檔命名空間。

PageResource1 是資源檔的名稱，這個名稱是固定的。

```
<%@ Page Culture="auto"
    meta:resourcekey="PageResource1"
    UICulture="auto" %>
```

- 在控制項上的繫結屬性

每個控制項也會對一個 meta:resourcekey attribute。在控制項上的 meta:resourcekey 是用來對照資源檔的 Key 名稱。Key 的名稱是使用控制項的 ID 之後加上 Resource1。

```
<asp:Label ID="Label1" runat="server" Text="Name:"
    meta:resourcekey="Label1Resource1"></asp:Label>
```

### 使用明確式運算式語法設定繫結

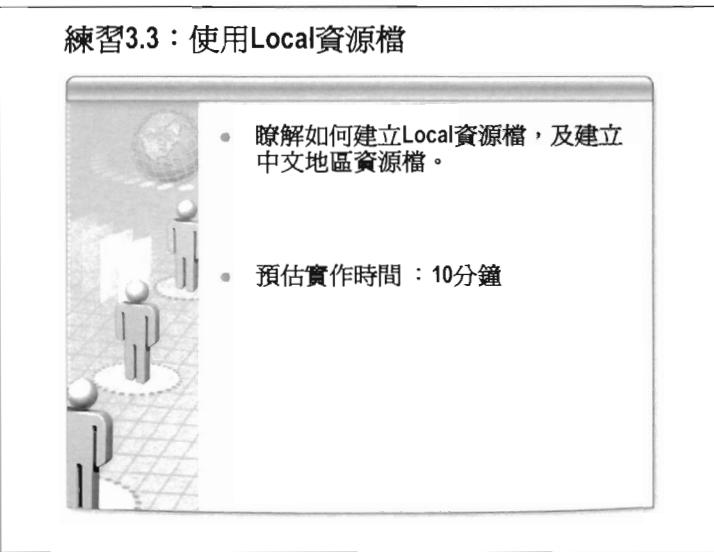
若想要直接地控制屬性的當地語系化設定，您可以使用明確式語法來取代隱含式當地語系化。若使用明確式語法來處理當地語系化，當網頁執行時，會透過運算式從資源檔中讀取值，然後使用此值來設定屬性。通常明確式語法都是應用在網頁需要使用大量需當地語系化的文字訊息時。

明確式本機資源運算式使用下列語法：

```
<%$ Resources : ResourceID%>
```

例如以下範例使用明確式運算式語法，從資源檔中，取出 Name 為 Label1Resource1.Text 對應的 Value：

```
<asp:Label ID="Label5" runat="server"
    Text="<%$ Resources:Label1Resource1.Text %>">
</asp:Label>
```



## 練習 3.3：使用資源檔

目的：

瞭解如何建立 Local 資源檔，及建立中文地區資源檔。

功能描述：

設定 Page2.aspx 為多國語言網頁，讓網頁依據使用者喜好語言選取網頁。

預估實作時間：10 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「Open」→「Web Site」→選取。「Web Site ...」→選取「磁碟:\U9544\Practices\VB 或 CS\Mod03\_3\Starter\Mod03\_1」。
3. 在「Solution Explorer」找到 Page2.aspx，並打開它的設計視窗。

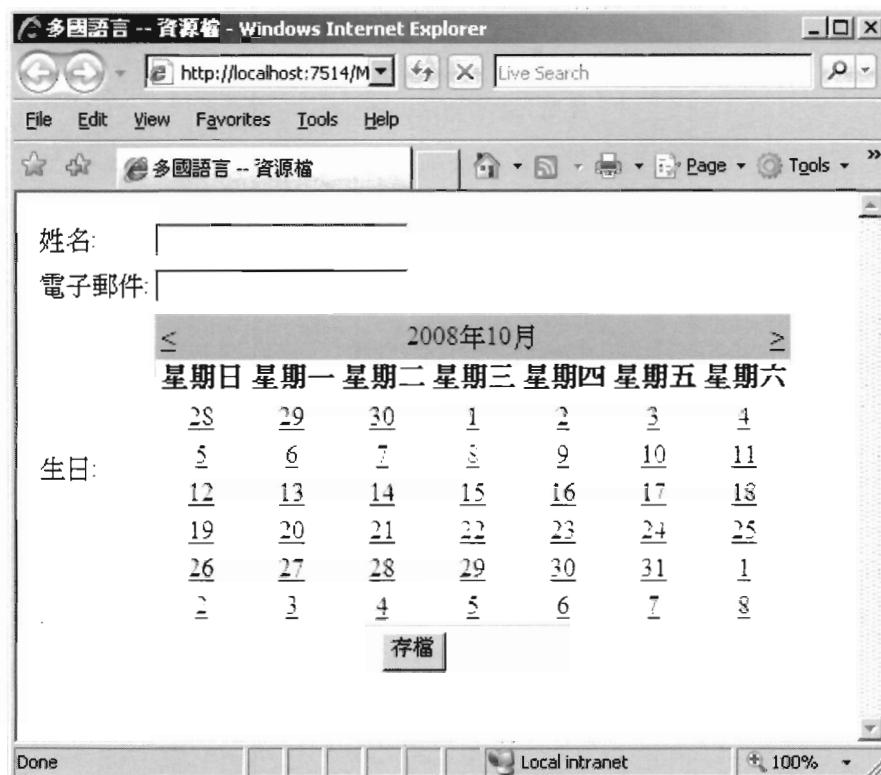
4. 在選單選取「Tools」→「Generate Local Resource」。

完成這個動作後，將會在「Solution Explorer」看到新增了一個「App\_LocalResources」目錄，以及目錄下多一個 Page2.aspx.resx。

5. 在「Solution Explorer」選取 Page2.aspx.resx 資源檔，然後按右鍵複製並且修改檔名為「Page2.aspx.zh-TW.resx」。
6. 打開 Page2.aspx.zh-TW.resx，將其中的英文訊息修改為中文訊息，如下圖。

| Name                               | Value       | Comment |
|------------------------------------|-------------|---------|
| Button1Resource1.Text              | 存檔          |         |
| Button1Resource1.ToolTip           |             |         |
| Calendar1Resource1.Caption         |             |         |
| Calendar1Resource1.NextMonthText   | &gt;        |         |
| Calendar1Resource1.PrevMonthText   | &lt;        |         |
| Calendar1Resource1.SelectMonthText | &gt;&gt;    | ↓       |
| Calendar1Resource1.SelectWeekText  | &gt;        |         |
| Calendar1Resource1.ToolTip         |             |         |
| Label1Resource1.Text               | 姓名:         |         |
| Label1Resource1.ToolTip            |             |         |
| Label2Resource1.Text               | 電子郵件:       |         |
| Label2Resource1.ToolTip            |             |         |
| Label3Resource1.Text               | 生日:         |         |
| Label3Resource1.ToolTip            |             |         |
| PageResource1.Title                | 多國語言 .. 資源檔 |         |
| TextBox1Resource1.Text             |             |         |
| TextBox1Resource1.ToolTip          |             |         |

7. 執行 Page2.aspx，當 IE 的語言設定為 zh-TW 地區時，出現的畫面將會套用 Page2.aspx.zh-TW.resx。IE 語系為非 zh-TW 地區，則此頁會套用 Page2.aspx.resx。



### 使用程式取回Local資源

- 使用HttpContext物件的GetLocalResourceObject 方法
- 需傳入網頁路徑與resourcekey 當參數

Visual Basic

```
string s = HttpContext.GetLocalResourceObject("MyPage.aspx",
"Label1Resource1.Text").ToString();
```

C#

```
string s = HttpContext.GetLocalResourceObject("MyPage.aspx",
"Label1Resource1.Text").ToString();
```

## 使用程式取回 Local 資源

如果你需要在網頁之中，使用程式來取得本機資源檔(Local Resource)中的值，例如在網頁設計階段尚無法確定資源的值，而需要在執行階段，依不同情況設定資源值，您可以使用 HttpContext 物件的 GetLocalResourceObject 方法。

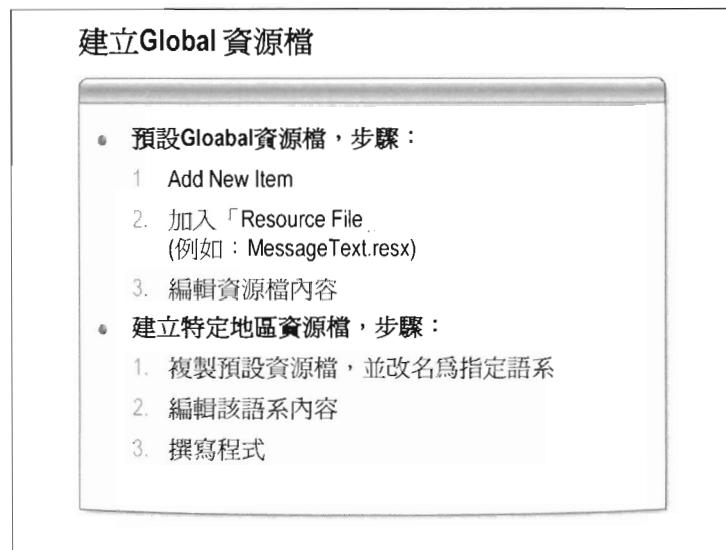
GetLocalResourceObject 方法需要使用 ResourceKey 來指定想存取的資源名稱。如下範例所示：

Visual Basic

```
string s = HttpContext.GetLocalResourceObject("MyPage.aspx", "L
abel1Resource1.Text").ToString()
```

C#

```
string s = HttpContext.GetLocalResourceObject("MyPage.aspx", "L
abel1Resource1.Text").ToString();
```



## 建立 Global 資源檔

一般不在畫面上的系統訊息，像是錯誤訊息、提示文字的多國語言，它們不是固定在畫面上的資訊，而是隨著使用者操作而動態顯示的訊息，它們的多國語言是存在 Global 資源檔中。使用 Visual Studio 的建立步驟如下：

1. 在 Solution Explorer，專案名稱滑鼠按右鍵選取「Add New Item」。
2. 在 Add New Item 視窗中選取「Resource File」，並給予適當檔名，然後點選「Add」。
3. 編輯資源檔內容。

## 建立特定地區資源檔

其他地區資源檔可以直接從預設資源檔複製後再編輯為該地區語言，這樣可以確保 Key 名稱的一致性。它的步驟如下：

1. 複製預設資源檔，並改名為指定語系。
2. 編輯該語系內容。

3. 撰寫程式使用資源檔。

### 使用程式存取 Global 資源檔

- 使用強型別物件  
使用Resources命名空間與相關類別

|                     |                                                    |
|---------------------|----------------------------------------------------|
| <b>Visual Basic</b> | Dim message As String = Resources.MessageText.save |
| <b>C#</b>           | string message = Resources.MessageText.save;       |

- 使用GetGlobalResourceObject方法

|                     |                                                                             |
|---------------------|-----------------------------------------------------------------------------|
| <b>Visual Basic</b> | Dim message As String = GetGlobalResourceObject("MessageText", "save")      |
| <b>C#</b>           | string message =(string)GetGlobalResourceObject(" MessageText ", " save "); |

## 使用程式存取 Global 資源檔

Global 資源檔的訊息如何在程式中取出？您可以使用兩種方法：

- 使用強型別物件，透過編譯時期自動產生的 Resources 命名空間語相關類別。
- 使用 HttpContext 物件的 GetGlobalResourceObject 方法。

### 使用強型別物件

當編寫完資源檔時，資源檔就是一個強型別物件，因此可以直接使用 Resources 命名空間及資源檔名(成為強型別物件)中的 Key 名稱做為屬性，以取出對應的值。使用 Resources 命名空間存取資源檔資訊，範例程式如下：

|                     |                                                    |
|---------------------|----------------------------------------------------|
| <b>Visual Basic</b> | Dim message As String = Resources.MessageText.save |
|---------------------|----------------------------------------------------|

|           |                                              |
|-----------|----------------------------------------------|
| <b>C#</b> | String message = Resources.MessageText.save; |
|-----------|----------------------------------------------|

Resources 為命名空間(完整名稱是:System.Resources)，MessageText 為資源檔名，save 為 MessageText 資源檔的 Key 名稱。

## 使用 **GetGlobalResourceObject** 方法

若要以程式設計方式擷取 Global 資源檔中的值，可以在 HttpContext 或 TemplateControl 類別 (Class) 中使用 GetGlobalResourceObject 方法。該方法取回的值是一個物件，因此，您必須將取回的資源轉換為適當的型別，才能在網頁中使用之。

GetGlobalResourceObject 方法需傳入資源類別的名稱和資源 ID。類別名稱是根據 .resx 的檔名而定。例如，MessageText.resx 資源檔案的類別名稱為 MessageText。使用 HttpContext 的 GetGlobalResourceObject 方法之範例程式如下：

Visual Basic

```
Dim message As String = GetGlobalResourceObject("MessageText", "save")
```

C#

```
string message =(string)GetGlobalResourceObject(" MessageText", " save ");
```

### 練習3.4：使用Global資源檔



- 瞭解如何存取Global資源檔的訊息，並使用程式取出資源檔文字。
- 預估實作時間：10分鐘

### 練習 3.4：使用 Global 資源檔

#### 目的：

瞭解如何存取 Global 資源檔的訊息，並使用程式取出資源檔文字。

#### 功能描述：

繼續前一個練習，建立 Global 資源檔，並使用程式取出資源檔文字。

#### 預估實作時間：10 分鐘

#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「Open」→「Web Site」→選取「磁碟:\U9544\Practices\VB 或 CS\Mod03\_4\Starter\Mod03\_1」或繼續前一個專案。

3. 在「Add New Item」視窗中選取「Resource File」，檔名取為「MessageText.resx」，然後點選「Add」。

4. 編輯資源檔，內容如下表：

| Name       | Value              |
|------------|--------------------|
| page2Title | Member Information |
| save       | Save Done.         |

5. 複製 MessageText.resx 到新的檔案，並改名為 MessageText.zh-TW.resx。

6. 編輯 MessageText.zh-TW.resx 資源檔，內容如下表：

| Name       | Value  |
|------------|--------|
| page2Title | 會員基本資料 |
| save       | 存檔.    |

7. 接下來，程式存取 Global 資源檔。

8. 開啓 Page2.aspx，在「Save」按鈕上雙擊二下，回到 Source 畫面。

9. 在頁面上方加入命名空間。

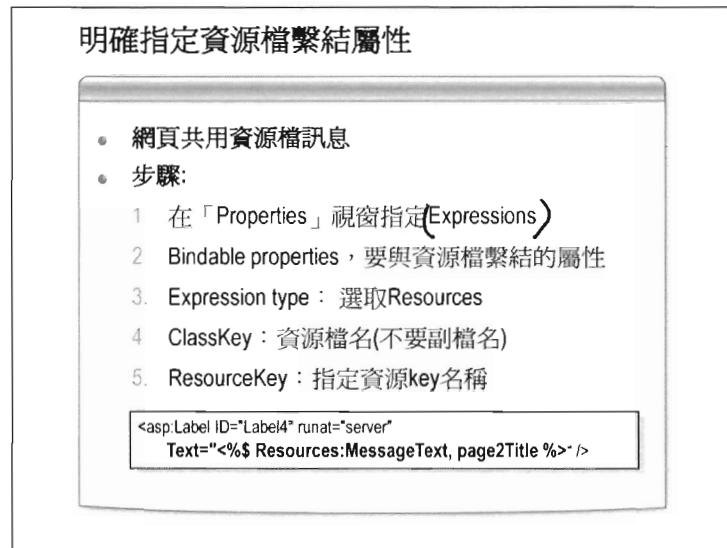
```
<%@ Import Namespace="System.Threading" %>
<%@ Import Namespace="System.Globalization" %>
```

10. 在 Button1\_Click 事件中撰寫程式碼如下：

```
Visaul Basic
Dim message As String
message = Resources.MessageText.save
Dim script As String = "alert(\"" & message & "\");"
Page.ClientScript.RegisterClientScriptBlock(Me.GetType(), _
"MyKey", script, True)
```

```
C#
String message := Resources.MessageText.save;
String script = "alert(\"" + message + "\");";
Page.ClientScript.RegisterClientScriptBlock(
    this.GetType(), "mykey", script,true );
```

11. 執行 Page2.aspx，點按「Save」或「存檔」按鈕，可以讀取英文或中文的訊息方塊。



## 明確指定資源檔繫結屬性

Global 資源檔屬於網站共用訊息，也就是多個頁面共用的訊息可以存放在 Global 資源檔。如何讓控制項繫結到 Global 資源檔呢？Visual Studio 提供了便利的方式，步驟如下：

1. 選取與 Global 資源檔繫結的控制項，然後點選「Properties」視窗，找到「Expressions」並點選後面的...。
2. 在「Expressions」視窗「Bindable properties」： 指定繫結的屬性。
3. 在「Expression type」： 下拉指定「Resources」。
4. 在「Expression properties」： 選取「ClassKey」，輸入 Global 資源檔名。
5. 在「Expression properties」： 選取「ResourceKey」並下拉選取資源的 Key 名稱。

### 使用 Localize 控制項

- 應用在靜態文字的當地語系化功能
- 和 Literal、Label 控制項非常的類似
- 可以直接在 Visual Studio 設計畫面中直接編輯
- Localize 控制項不能套用樣式 (Style)

```
<asp:Localize id="Localize1"
    runat="server"
    meta:resourcekey="Localize1Resource1"
    Text="System Information">
</asp:Localize>
```

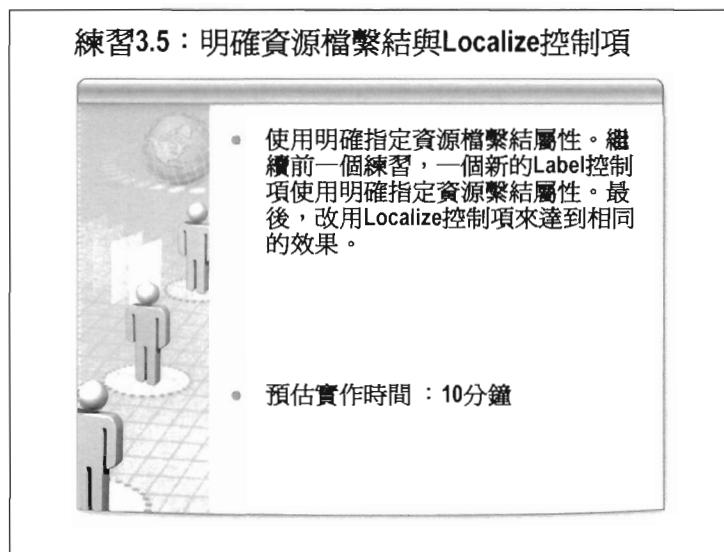
### 使用 Localize 控制項

ASP.NET 提供一個 Localize 伺服器控制項可以應用在靜態文字的當地語系化功能。例如某家公司的簡介或版權宣告若有需支援當地語系化時，就可以使用之。Localize 伺服器控制項主要是應用在網頁中預留當地語系化文字將要顯示的位置。

此控制項和 Literal、Label 控制項非常的類似，但不同的是，它不僅可以在屬性頁設定欲顯示的文字，也可以直接在 Visual Studio 設計畫面中直接編輯。另外 Label 控制項可以套用樣式 (Style)；但 Localize 控制項則不行。

若要設定 Localize 控制項要顯示的資料，可以使用 Text 屬性。以下是使用 Localize 控制項的範例：

```
<asp:Localize id="Localize1"
    runat="server"
    meta:resourcekey="Localize1Resource1"
    Text="System Information">
</asp:Localize>
```



## 練習 3.5：明確資源檔繫結與 Localize 控制項

**目的：**

使用明確指定資源檔繫結屬性。網頁標題具有多國語言的顯示能力。

**功能描述：**

繼續前一個練習，一個新的 Label 控制項使用明確指定資源繫結屬性。最後，改用 Localize 控制項來達到相同的效果。

**預估實作時間：10 分鐘**

**實作步驟：**

- 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。

- 從「File」→「Open」→「Web Site」→選取「磁碟:\U9544\Practices\VB 或 CS\Mod03\_5\Starter\Mod03\_1」或繼續前一個專案。

3. 開啓 Page2.aspx 的「Source」畫面，從「Toolbox」拖拉 Label 控制項到

## 區段之內，如下所示。

```
<h2 style="text-align: center;">
    <asp:Label ID="Label4" runat="server">
    </asp:Label>
</h2>
```

4. 切換到「Design」畫面。(因為「Expressions」屬性只有在「Design」畫面時才會出現)
5. 然後點選「Properties」視窗，找到「Expressions」並點選後面的...。
6. 在「Expressions」視窗「Bindable properties：」選取 Text 屬性。
7. 在「Expression type：」下拉選取「Resources」。
8. 在「Expression properties：」選取「ClassKey」，輸入 MessageText(資源檔名)。
9. 在「Expression properties：」選取「ResourceKey」並下拉選取 page2Title(Key 名稱)。
10. 執行結果如下圖：



11. 結束程式，回到設計畫面。刪除步驟二建立的 Label，改用 Localize 控制項。從「Toolbox」工具箱中拖拉一個 Localize 控制項到網頁中

## 區段之內，如下所示：

```
<h2 style="text-align: center;">
    <asp:Localize ID="Localize1" runat="server"></asp:Localize>
</h2>
```

12. 修改 Text 屬性如下：

```
<asp:Localize ID="Localize1" runat="server"
    Text='<%$ Resources:MessageText,page2Title %>' >
</asp:Localize>
```

13. 執行程式，可以得到和上一次相同的結果。

### 總結

- 了解什麼是本地化
- 設定語言喜好設定
- 取得用戶端選用語系
- 認識資源檔
- 學會建立Local 資源檔與Global 資源檔

### 總結

本章介紹如何利用 Visual Studio 快速設計一個多國語言的網頁。完成本章課程，您必須了解什麼是當地語系化，以及如何在 IE 上設定喜好語言，程式如何判斷使用者的語言。另外，網頁的資源檔(local)、網站的共用資源檔(Global)的使用與存取方式也是本章重點。

# 第四章: 狀態管理機制

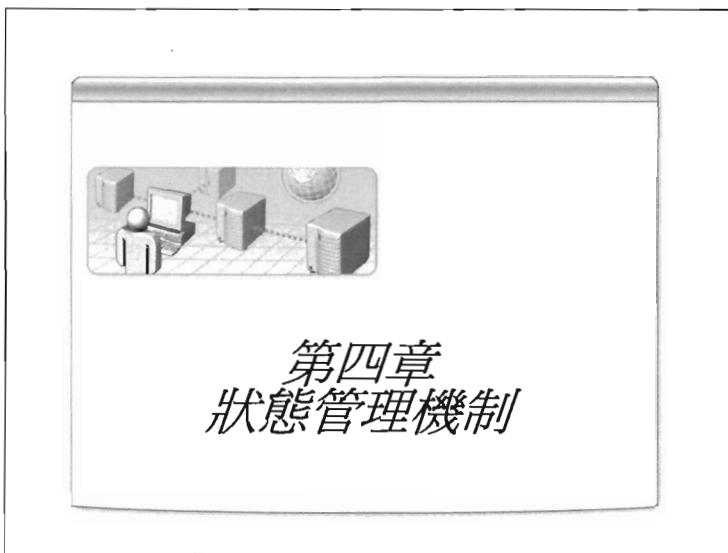
## 本章大綱

什麼是狀態維護.....	4
狀態維護的類型.....	6
狀態維護的運作.....	7
HttpApplicationState 物件.....	8
練習 4.1 : 使用 Application 物件.....	10
Session 物件.....	15
使用 Session 物件.....	17
練習 4.2 : 使用 Session 物件.....	19
Webfarm 架構下的狀態維護.....	24
使用 Out-of-Process Session.....	26
練習 4.3 : 使用 StateServer 儲存 Session .....	27
儲存 Session 到 SQL Server.....	29
練習 4.4 : 使用 SQL Server 儲存 Session .....	30
ViewState 物件.....	36
使用 Cookie .....	37
HttpCookie 物件的使用.....	38
練習 4.5 : 使用 Cookie 紀錄使用者工作階段 .....	40

作者：

趙敏翔





## 大綱

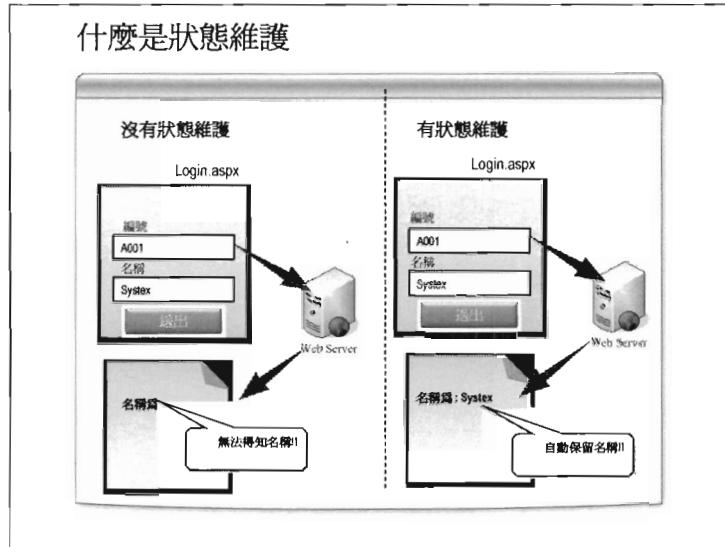
- 狀態管理機制簡介
- Session 物件設計
- Application 物件應用
- WebFarm 架構時狀態的維護
- ViewState 機制
- Cookie 物件設計
- 總結

本章將介紹 ASP.NET 的狀態機制維護管理，包含介紹伺服器端用來保存使用者狀態的 Session 物件、Application 物件等運作方式，以及使用者端的 Cookie 和 ViewState 物件等運用。

此外，也將進一步探討當 ASP.NET 網站系統想要建置 WebFarm 架構時，所需考量的狀態機制管理與設定。

在這一章中將學習到

- Session 物件設計
- Application 物件應用設計
- WebFarm 架構時狀態的維護設定
- ViewState 機制運用
- Cookie 物件設計



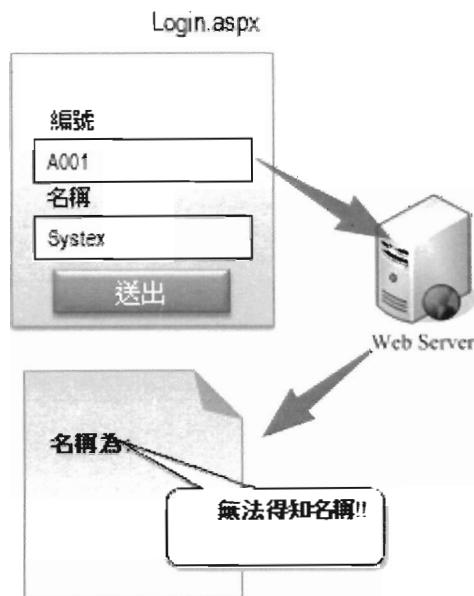
## 什麼是狀態維護

ASP.NET 網站應用程式的運作方式為使用者打開瀏覽器，輸入網址瀏覽網頁後，使用者端的瀏覽器會透過網路存取 ASP.NET 的網頁，網頁執行完畢，再將執行後的結果產生一份 HTML 輸出到使用者端的瀏覽器呈現。

因為 ASP.NET 整體運作是透過 HTTP 來存取，而 HTTP 本身的運作過程當中，並不會自動幫使用者保留當初在瀏覽器中所輸入的相關資料，這樣的運作機制稱為 Stateless，也就是無狀態機制。

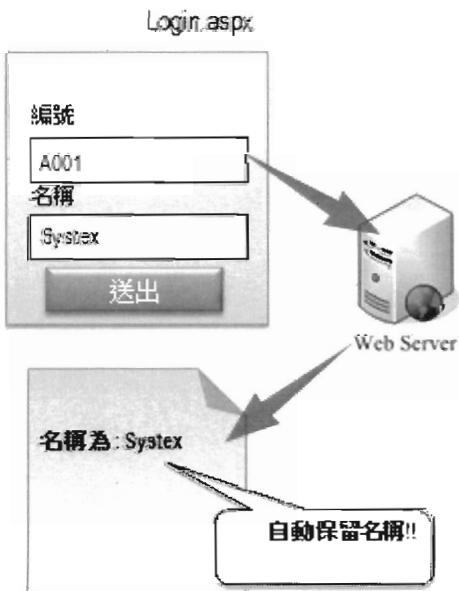
然而對開發者來說，Stateless 機制卻是一大惡夢，根據微軟統計早期的網頁設計者大概一隻網頁應用程式的完成時間，有三分之一是花在將使用者輸入的資料，正確的保留起來，並填回適當位置的瑣碎設計中。

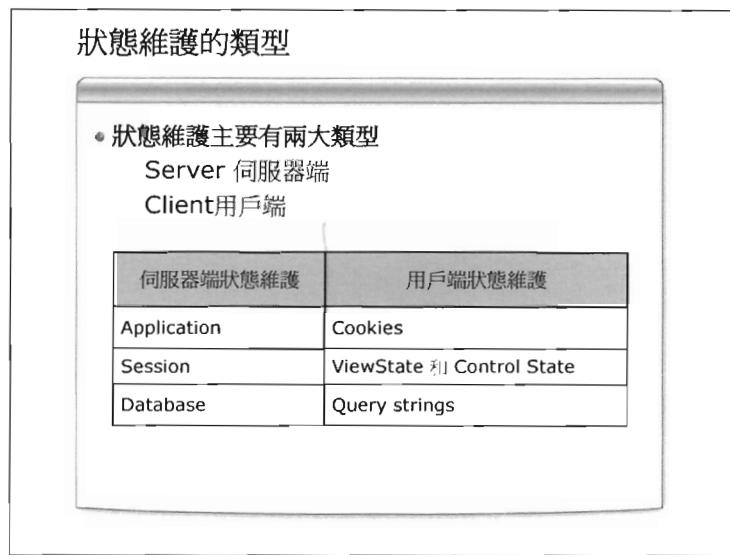
## 沒有狀態維護



因為使用者的資料並無法自動保留，因此開發者就必須利用各種物件或是方式來將這些使用者的資料正確的保留，並且在網頁執行後，放回正確的位置上，將使用者狀態保留這樣的機制稱為 Statefull，也就是有狀態機制。

## 有狀態維護





## 狀態維護的類型

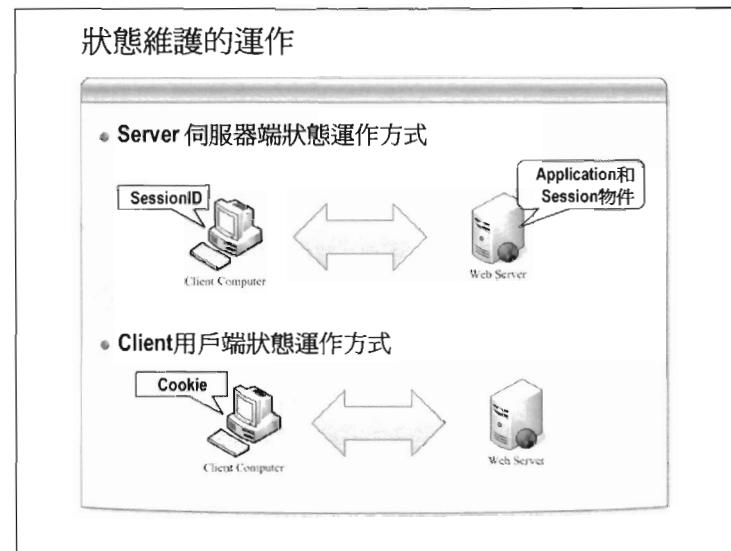
設計 ASP.NET 的狀態維護運作，主要可以分為兩個部分來考量，一個是伺服器端的狀態維護，一個是用戶端的狀態維護。

伺服器端的狀態維護可由以下幾個主要的物件或機制來設計：

- Application 物件
- Session 物件
- Database 資料庫

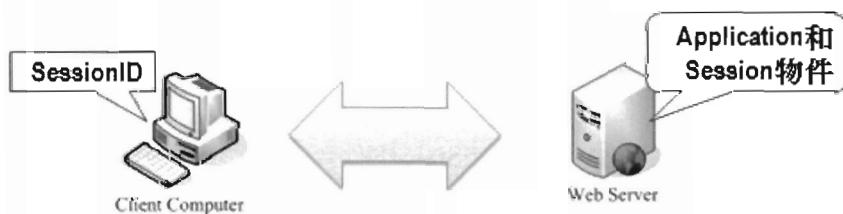
用戶端的狀態維護可由以下幾個主要的物件或機制來設計：

- Cookie 物件
- ViewState 物件
- Query String

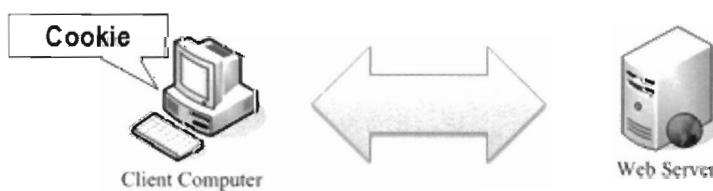


## 狀態維護的運作

狀態維護的運作方式可分為伺服器端和用戶端兩個部分來探討，伺服器端最主要的狀態維護物件為 Application 和 Session 物件，這兩個物件預設都是保留在網站伺服器 IIS 的記憶體中。



用戶端者主要是使用 Cookie 物件或是 HTML 中的隱藏欄位(Hidden Field)，甚至是網址中的查詢字串(Query String)，來保留使用者所需要記錄的資訊，這些資料都是保留在用戶端的瀏覽器或是用戶端的電腦中。



### HttpApplicationState 物件

- ASP.NET 應用程式中，用來暫存應用程式中共享資料的物件
- HttpApplicationState 物件的使用方式
 

Visual Basic	Application("CompanyName") = "NetDB"
C#	Application["CompanyName"] = "NetDB";
- 如何取出HttpApplicationState物件中的資料
 

Visual Basic	Label1.Text = CType(Application("CompanyName"), String)
C#	Label1.Text = (String) Application["CompanyName"];

## HttpApplicationState 物件

HttpApplicationState 物件主要是用來保留整個網站中的共用資料，不管是否為同一個瀏覽器，或是網站中的不同網頁，所存取到的 HttpApplicationState 物件一定是同一個，將來這一些資料就可以在應用程式中的任何地方被使用。

HttpApplicationState 物件實體可直接透過網頁物件 Page 的 Application 屬性直接存取，但是因為 Application 中的資料是共享的資料，所以建議在修改前先鎖定 Application 物件，使用完之後再釋放鎖定，設定值的方式為：

Visual Basic	<pre>Application.Lock() Application("CompanyName") = "NetDB" Application.UnLock()</pre>
--------------	-----------------------------------------------------------------------------------------

C#	<pre>Application.Lock(); Application["CompanyName"] = "NetDB"; Application.UnLock();</pre>
----	--------------------------------------------------------------------------------------------

取出設定值的時候，記得要轉型回原本資料的型別，取出方式為：

Visual Basic

```
Label1.Text = CType(Application("CompanyName") , String)
```

C#

```
Label1.Text = (String) Application["CompanyName"];
```



## 練習 4.1：使用 Application 物件

目的：

這個練習主要是了解如何使用 Application 物件來保留網站上共用的資訊。

功能描述：

在這個練習中，使用 Application 物件搭配 Global.asax 中的共用事件來設計網站的瀏覽人數計數器，讓每一個使用者所存取到的瀏覽人數計數器都是相同的值。

預估實作時間：10 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選

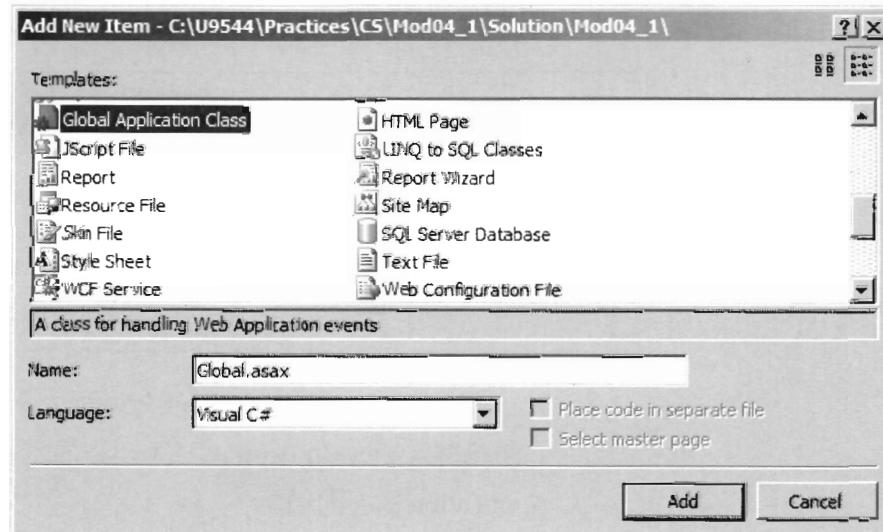
取「\U9544\Practices\VB 或 CS\Mod04\_1\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod04\_1」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 在網頁的 Page\_Load 的事件中，加入將瀏覽人數計數器顯示在畫面的程式碼：

```
Visual Basic
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Response.Write("網頁瀏覽人數：" + CType(Application("counter"), String))
End Sub
```

```
C#
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("網頁瀏覽人數：" + Application["counter"].ToString());
}
```

5. 自主選單「Web Site」下「Add New Item...」，選取「Global Application Class」，使用預設的檔名 Global.asax。



6. 在 Global.asax 中 Application 的 Start 事件程序中，設定起始一個 Application 物件，資料名稱為 counter，資料值預設為 0，程式碼為：

```
Visual Basic
Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
    ' Code that runs on application startup
    Application("counter") = 0
End Sub
```

```
C#
void Application_Start(object sender, EventArgs e)
{
    // Code that runs on application startup
    Application["counter"] = 0;
}
```

7. 假設不同瀏覽器的使用者進入，就會把計數器累加。在 Session 的 Start 事件程序中，將 Application 中的 counter 值累加，累進值為 1，程式碼為：

```
Visual Basic
Sub Session_Start(ByVal sender As Object, ByVal e As EventArgs)
    ' Code that runs when a new session is started
    Application.Lock()
    Application("counter") = CType(Application("counter"), Integer) + 1
    Application.UnLock()
End Sub
```

```
C#
void Session_Start(object sender, EventArgs e)
{
    // Code that runs when a new session is started
    Application.Lock();
    Application["counter"] = (int)Application["counter"] + 1;
    Application.UnLock();
}
```

8. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。可以看到畫面有存取到 Application 物件的值：



9. 複製網址，打開另外兩個不同的瀏覽器，瀏覽同一個網頁，可以看到瀏覽人數會累加。



10. 接著把每一個瀏覽器中的網頁都重新整理，可以看到都會出現相同的值，表示所存取的都是同一份資料。



### Session物件

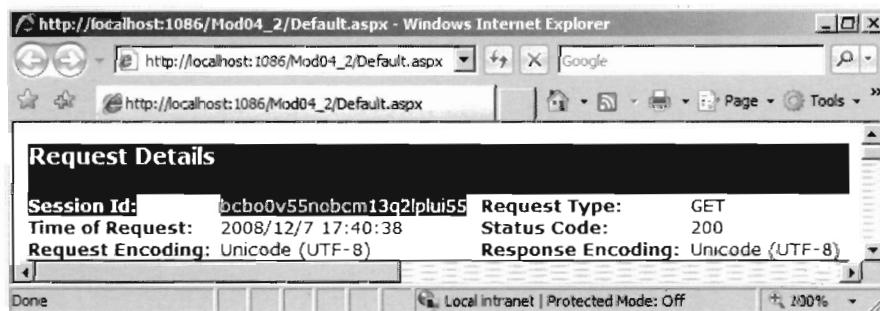
- Session物件的運作方式
- Session的時效性預設為 20 分鐘
- 可透過web.config更改設定

```
<configuration>
  <system.web>
    <sessionState timeout="10" />
  </system.web>
</configuration>
```

- Session使用上效能的考量
  - 記憶體資源
  - 儲存模式的影響

## Session 物件

Session一般稱為使用者工作階段，泛指使用者本次操作網站的過程，其運作方式為當使用者打開瀏覽器存取 ASP.NET 網站時，Web 伺服器會為每一個瀏覽器的連線隨機產生一個 SessionID，並且將所要儲存的資料寫入 Session 物件中，並透過 SessionID 來記錄對應。



接著，當 Web 伺服器要把網頁資料回應至用戶端時，此時會將 SessionID 寫入瀏覽器的記憶體中(存放於一個非持續性 Cookie 中，有關 Cookie 物件將會在本章後續介紹)。

如此，下次使用者連線到 Web 伺服器時，就可以根據 Session ID，找出存放使用者工作階段資料的 Session 物件。

## Session 的 Timeout 異動

Session 的運作過程會因為使用者關閉瀏覽器或是操作逾時而結束工作階段，工作階段的過期時間稱為 Timeout，預設是 20 分鐘，如果想要改變網站的 Timeout 時間，可以設定 web.config 中 sessionState 項目的 timeout 屬性值，下列設定為將工作階段的過期時間設定為 10 分鐘。

```
<configuration>
    <system.web>
        <sessionState timeout="10" />
    </system.web>
</configuration>
```

### 使用 Session 物件

#### HttpSessionState

##### HttpSession 物件

為集合物件，可暫存任何資料

##### 將資料存入 Session

```
Visual Basic  
Session("BookName") = "ASP.NET"
```

```
C#  
Session["BookName"] = "ASP.NET";
```

##### 從 Session 取出資料

```
Visual Basic  
lblFName.Text = CType(Session("BookName"), String)
```

```
C#  
lblFName.Text = (String)Session["BookName"];
```

## 使用 Session 物件

HttpSession 物件的執行個體可以由 Page 物件的 Session 屬性取得，透過 Session 屬性可以直接將任何的資料記錄到 Session 物件中，因為 Session 物件是根據 SessionID 對應，因此只要是同一個瀏覽器，雖然是存取網站中的不同網頁，但是一樣可以在不同的網頁中存取到 Session 物件中的資料，例如：購物車的設計，就可以使用 Session 來保留資料。

HttpSession 物件在網頁中，設定值的方式可以透過 Page 物件的 Session 屬性設定，程式為：

```
Visual Basic  
Session("BookName") = "ASP.NET"
```

```
C#  
Session["BookName"] = "ASP.NET";
```

取出設定值的時候，記得要轉型回原本資料的型別，取出方式為：

Visual Basic

```
Label1.Text = CType(Session("BookName"), String)
```

C#

```
Label1.Text = (String) Session["BookName"];
```

**練習 4.2 : 使用Session物件**



在這個練習中，將了解如何使用 Session 物件保留目前使用者所想儲存之相關資料，並在不同網頁中將儲存的 Session 物件取出並將資料呈現在畫面上。

預估實作時間：20分鐘

## 練習 4.2 : 使用 Session 物件

目的：

在這個練習中，將了解如何使用 Session 物件保留目前使用者所想儲存之相關資料，並在不同網頁中將儲存的 Session 物件取出並將資料呈現在畫面上。

功能描述：

在這個練習中，使用 Session 物件設計一個簡單的書籍購物車，讓使用者可以挑選喜歡的書籍，放到購物車之後，可以在另一個結帳的網頁將購物車中的資料取出。

預估實作時間：20 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod04\_2\Starter」目錄，與使

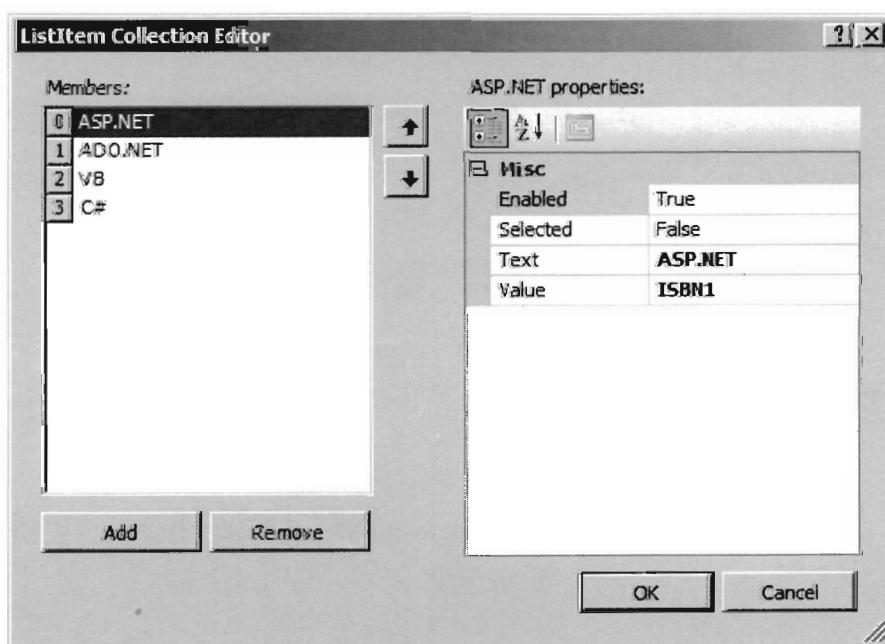
用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod04\_2」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 設計網頁如下圖所示：

書籍清單：  
 ASP.NET  
 ADO.NET  
 VB  
 C#

**加入購物車**

5. 從工具箱拖拉一個 CheckBoxList 以及一個 Button 控制項到網頁中，Button1 的 Text 屬性設定為「加入購物車」，CheckBoxList1 的 Items 屬性，加入幾筆書籍測試資料，如下圖。



6. 雙擊 Button1 控制項，在 Button1 的 Click 事件中，搭配 Dictionary 泛型物件，將所挑選的書籍清單儲存到 Dictionary

物件，再存到 Session，Name 為「BookLists」的物件中，程式碼如下：

```
Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim books As New Dictionary(Of String, String)
    For Each book As ListItem In CheckBoxList1.Items
        If book.Selected = True Then
            books.Add(book.Value, book.Text)
        End If
    Next

    Session("BookLists") = books
    Response.Write("加入購物車成功")
End Sub
```

```
C#
protected void Button1_Click(object sender, EventArgs e)
{
    Dictionary<string, string> books =
        new Dictionary<string, string>();
    foreach (ListItem book in CheckBoxList1.Items)
    {
        if (book.Selected)
        {
            books.Add(book.Value, book.Text);
        }
    }
    Session["BookLists"] = books;
    Response.Write("加入購物車成功");
}
```

7. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，檔名命名為「Payment.aspx」。
8. 在預設的 Default.aspx 網頁中，再加入一個 Button 控制項，Button2 的 Text 屬性設定為「結帳」，並設定 PostBackUrl 屬性為「~/Payment.aspx」。

## 書籍清單：

- ASP.NET
- ADO.NET
- VB
- C#

加入購物車	結帳
-------	----

9. 在 Payment.aspx 的 Page\_Load 事件中，讀取 Session 物件，並轉換為適當的 Dictionary 型別物件，將 Default.aspx 中加入購物車的書籍清單列出，程式碼如下：

## Visual Basic

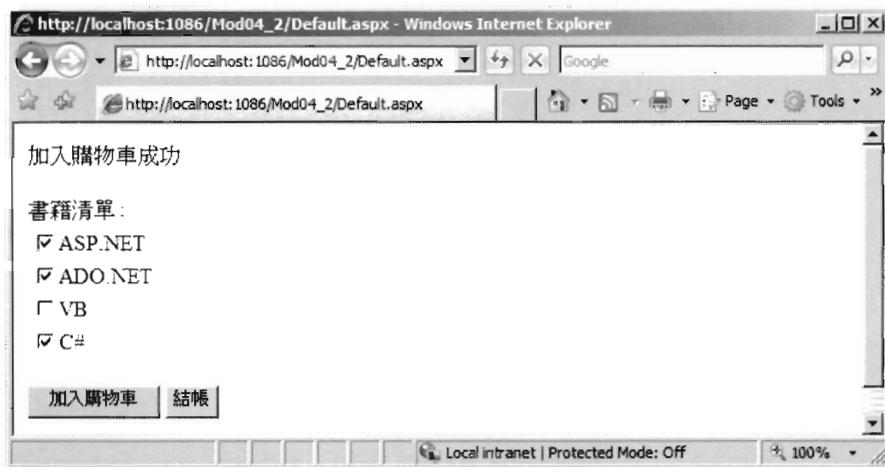
```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    If Session("BookLists") IsNot Nothing Then
        Response.Write("你訂購的書籍有：<br>")
        Dim books As Dictionary(Of String, String) = _
            CType(Session("BookLists"), Dictionary(Of String, String))
        For Each book As KeyValuePair(Of String, String) In books
            Response.Write("書號：" + book.Key + "書名：" + book.Value +
e + "<br>")
        Next
    End If
End Sub
```

## C#

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["BookLists"] != null)
    {
        Response.Write("你訂購的書籍有：<br>");
        var books = (Dictionary<string, string>)Session["BookLists"];
        foreach (var book in books)
        {
            Response.Write("書號：" + book.Key + "書名：" + book.Value +
e + "<br>");
        }
    }
}
```

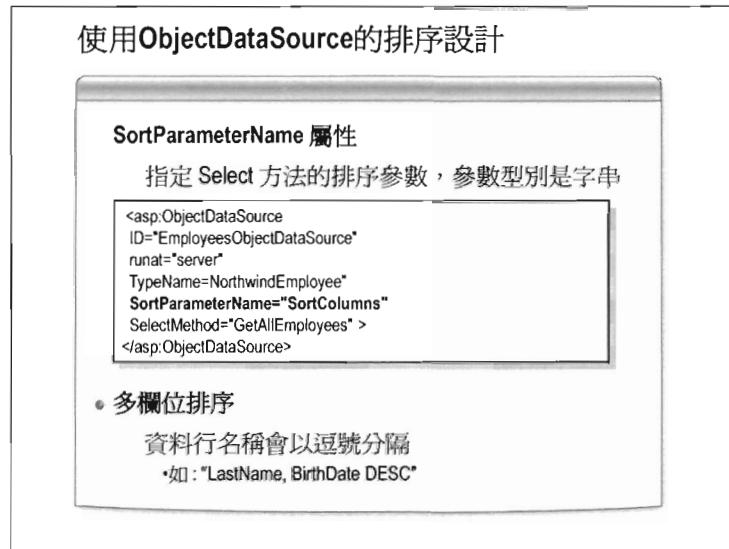
10. 執行網頁測試。在「Solution Explorer」→點選 Default.aspx 網頁→按滑鼠右鍵→選「View In Browser」。

11. 當網頁執行時，勾選幾本書籍資料，並按下「加入購物車」。



12. 接著按下「結帳」，將網頁導向另一個網頁 Payment.aspx，將儲存的 Session 物件中的資料呈現在網頁上。



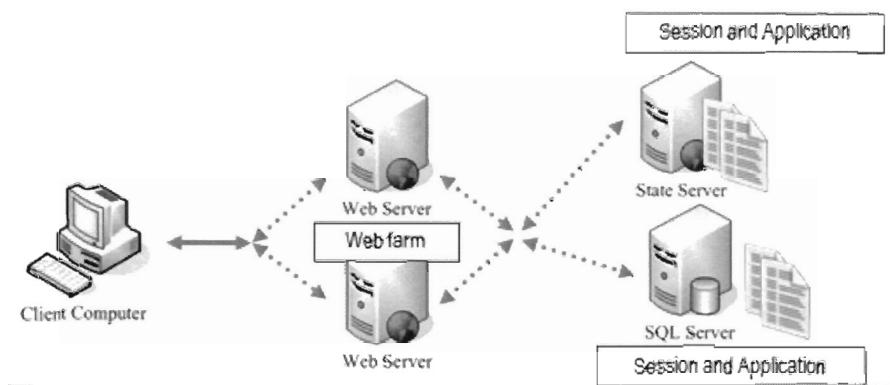


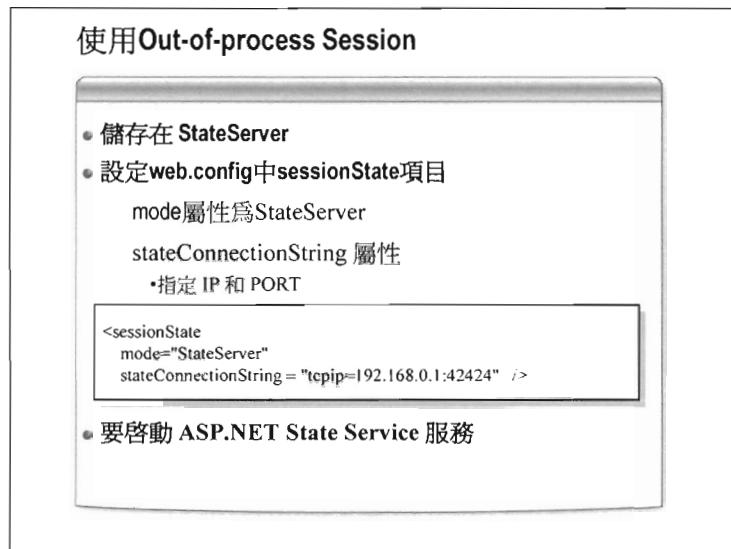
## Webfarm 架構下的狀態維護

有些受歡迎或是有大量使用者存取的網站，在架構設計上，不會只有使用一台 Web 伺服器來執行網站應用程式，通常會使用好幾台以上的 Web 伺服器，建置成 Webfarm 架構搭配網路負載平衡(NLB)機制來建置網站架構。

然而在這樣的 Webfarm 架構下，狀態維護變成是非常重要的一門學問，因為 ASP.NET 的狀態維護預設是設定為 In-Process，也就是說直接將資料記錄儲存的物件存放在網站應用程式所在的 Web 伺服器的記憶體中。

那麼，如果 NLB 機制啓動，將同一個使用者的存取導向到另一台網站伺服器，那麼該使用者的狀態資料將存取錯亂，甚至存取不到。為了解決這方面的問題，ASP.NET 也提供了 out-of-process 的狀態維護，也就是說可以將 Session 或 Application 等物件存放到其它地方，如某台機器的應用程式服務中，或是資料庫。



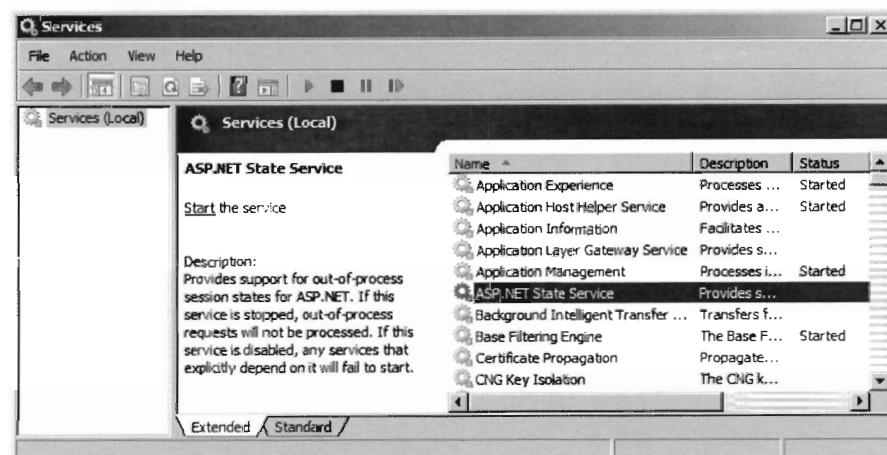


## 使用 Out-of-Process Session

要將 ASP.NET 的 Session 或是 Application 物件儲存到另一台電腦的話，首先，必須設定網站 web.config 中的 sessionState 項目，並且把 mode 屬性設定為 StateServer。再搭配 stateConnectionString 設定要存放的伺服器的 IP 位址以及 Port。

```
<sessionState
  mode="StateServer"
  stateConnectionString = "tcpip=192.168.0.1:42424" />
```

接著啟動指定伺服器的服務：ASP.NET State Service。



### 練習 4.3 : 使用 StateServer 儲存 Session

在這個練習中，將了解如何將 Session 物件保留在指定伺服器的 StateServer 服務中，以達到彈性化設計。

預估實作時間：10分鐘

### 練習 4.3 : 使用 StateServer 儲存 Session

#### 目的：

在這個練習中，將了解如何將 Session 物件保留在指定伺服器的 StateServer 服務中，以達到彈性化設計。

#### 功能描述：

在這個練習中，透過學習設定 web.config 以及啟動 State Server 所需要的啟動的服務，讓 Session 物件可以存放到指定的伺服器中。

預估實作時間：10 分鐘

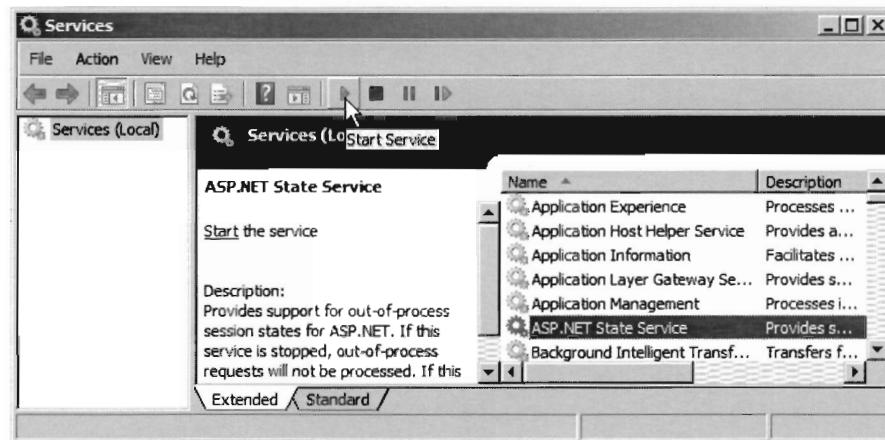
#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啟動 Visual Studio 2008 開發環境。
2. 從「File」→「Open Web Site」→選取「ASP.NET Web Site」→選擇「File System」設定 Folder 選取「\U9544\Practices\VB 或 CS\Mod04\_3\Starter\Mod04\_3」目錄，點選「Open」。

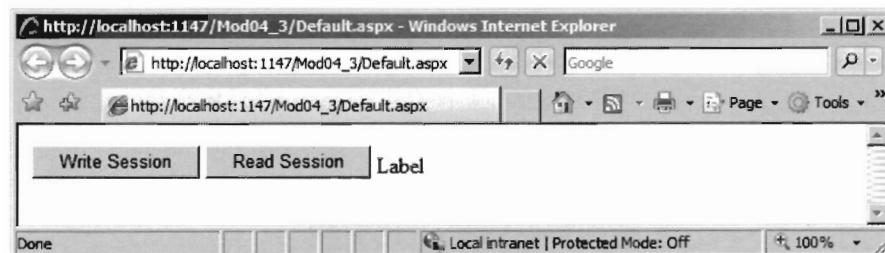
3. 打開 web.config，在 system.web 項目下，加入 sessionState 項目的設定，設定其 mode 屬性為 StateServer，且透過 stateConnectionString 指定伺服器，設定如下：

```
<sessionState mode="StateServer"
stateConnectionString="tcpip=127.0.0.1:42424"/>
```

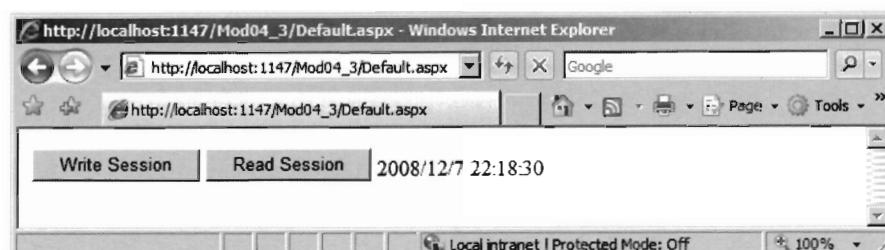
4. 接著啓用 ASP.NET State Service 服務。從「Start」→「Administrative Tools」→「Services」，開啟服務視窗之後，將 ASP.NET State Service 服務啓用。

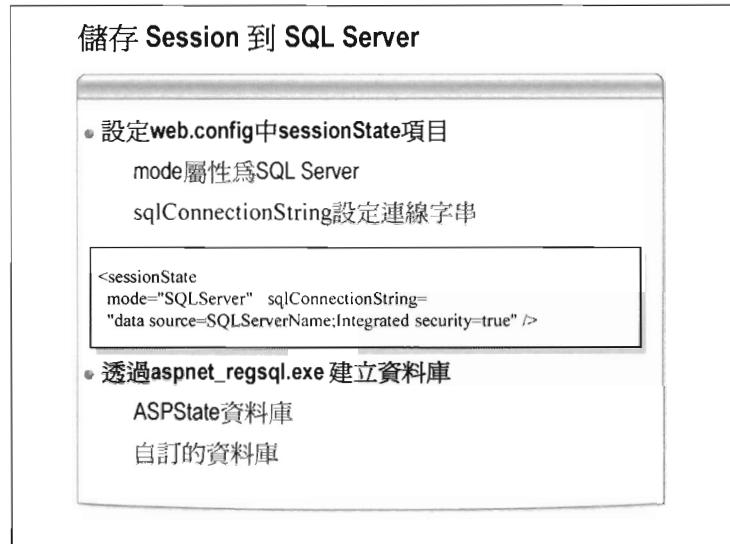


5. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。



6. 當網頁執行時，先點選「Write Session」將現在時間寫入 Session，接著再點選「Read Session」取出 Session，此時 session 已經存放在 ASP.NET State Service 服務中。





## 儲存 Session 到 SQL Server

若是要把 ASP.NET 的 Session 或是 Application 物件儲存 SQL 資料庫中，那麼有兩個主要的步驟要執行，第一個步驟是設定 `web.config` 指定要將狀態物件存放到資料庫中。

```
<sessionState
  mode="SQLServer"  sqlConnectionString=
  "data source=localhost;Integrated security=true" />
```

第二個步驟是安裝用來儲存狀態物件的資料庫。打開「Visual Studio 2008 Command Prompt」直接執行 `Aspnet_Regsql.exe` 工具以安裝 Session State 資料庫。

執行 `aspnet_regsql.exe` 工具需要設定以下幾個主要輸入的參數：

- -S 參數：安裝的 SQL Server 實體。
- -E 參數：使用目前的使用者身份登入 SQL Server。
- -ssadd 參數：指定要安裝 Session State 資料庫。
- -sstype 參數：指定 Session 存放位置，預設是存在 tempdb 資料庫中，p 代表存放在 ASPState 資料庫中，c 代表自訂的資料庫中。

#### 練習 4.4 : 使用 SQL Server 儲存 Session

在這個練習中，將了解如何將 Session 物件保留在 SQL Server 資料庫中，以達到彈性化設計。

預估實作時間：15分鐘

#### 練習 4.4 : 使用 SQL Server 儲存 Session

##### 目的：

在這個練習中，將了解如何將 Session 物件保留在 SQL Server 資料庫中，以達到彈性化設計。

##### 功能描述：

在這個練習中，透過學習設定 web.config 以及透過 aspnet\_regsql.exe 建置用來存放狀態物件的資料庫，讓 Session 物件可以存放到指定的資料庫中。

預估實作時間：15 分鐘

##### 實作步驟：

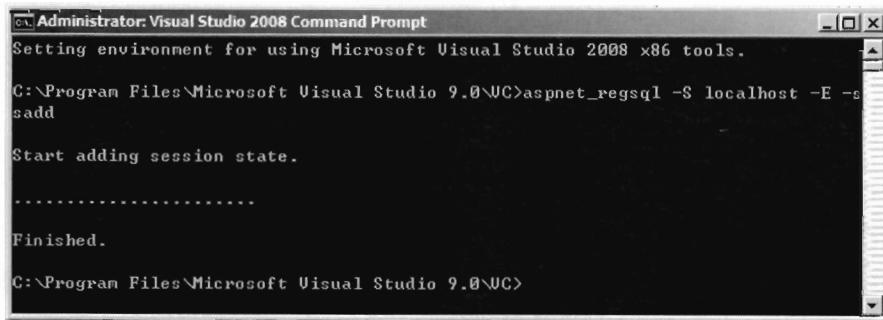
1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「Open Web Site」→選取「ASP.NET Web Site」→選擇「File System」設定 Folder 選取「\U9544\Practices\VB 或 CS\Mod04\_4\Starter\Mod04\_4」目錄，點選「Open」。

3. 打開 web.config，在 system.web 項目下，加入 sessionState 項目的設定，設定其 mode 屬性為 SQLServer，且透過 stateConnectionString 指定伺服器，設定如下：

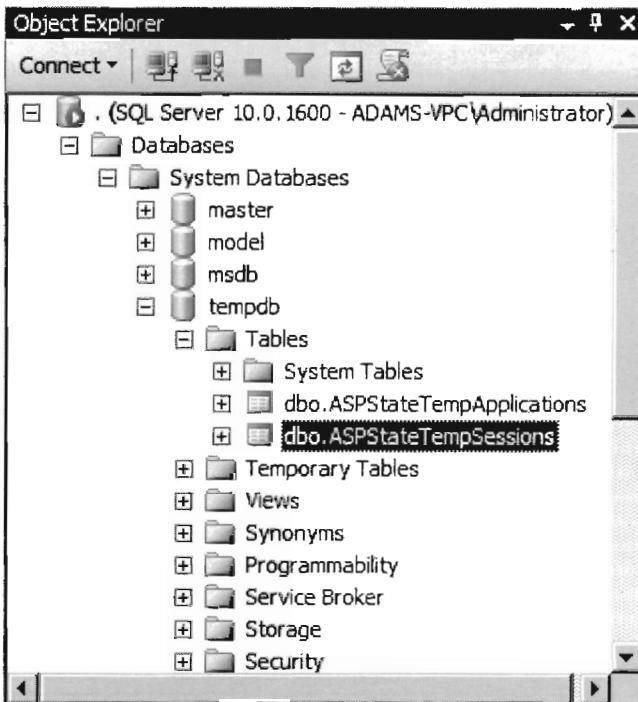
```
<sessionState mode="SQLServer"
    sqlConnectionString="Data Source=localhost;Integrated
    security=true"/>
```

4. 從「Start」→「All Programs」→「Microsoft Visual Studio 2008」→「Visual Studio Tools」開啟 Visual Studio 2008 Command Prompt。
5. 使用 aspnet\_regsql.exe 工具建置資料庫。當存放資料的資料表是存放在 tempdb 資料庫時，在 Visual Studio 2008 Command Prompt 視窗中輸入：

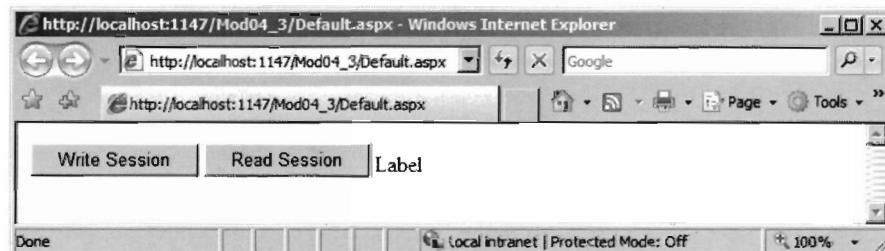
```
aspnet_regsql.exe -S localhost -E -ssadd
```



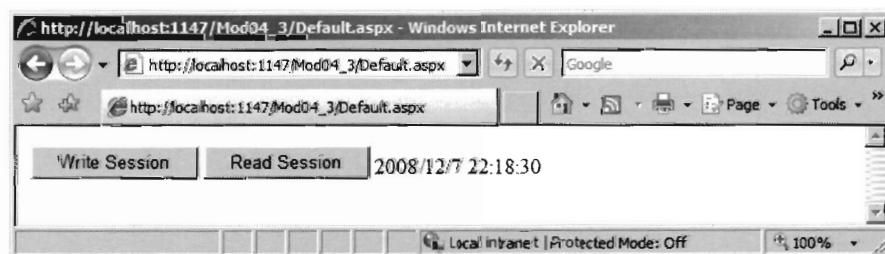
6. 此時會在 SQL Server 的 tempdb 資料庫中建立用來存放資料的資料表。



7. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。



8. 當網頁執行時，先點選「Write Session」將現在時間寫入 Session，接著再點選「Read Session」取出 Session，此時 Session 已經存放在 SQL Server 中。



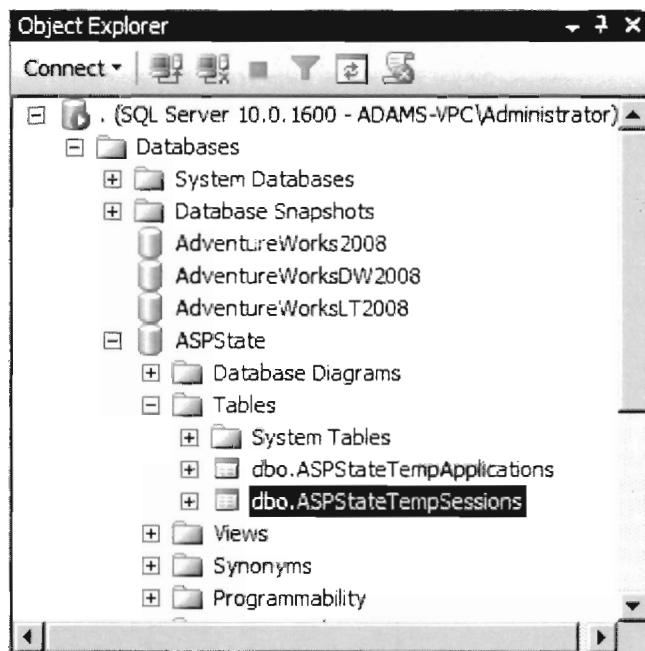
9. 檢視 tempdb 資料庫的 ASPStateTempSessions 資料表：

	SessionId	Created	Expires	LockDate
1	tg00az4530skpy4504pb3ybe3bbf943	2008-12-07 14:54:47.460	2008-12-07 15:14:47.460	2008-12-07 14:54:47.460
2	vd33keyyud5ydecnrjhqc45e3bbf943	2008-12-07 14:54:51.407	2008-12-07 15:14:52.413	2008-12-07 14:54:52.057

10. 當存放資料的資料表是存放在 ASPState 資料庫時，在 Visual Studio 2008 Command Prompt 視窗中輸入：

```
aspnet_regsql.exe -S localhost -E -ssadd -sstype p
```

11. 此時會在 SQL Server 的 ASPState 資料庫中建立用來存放資料的資料表。



12. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。

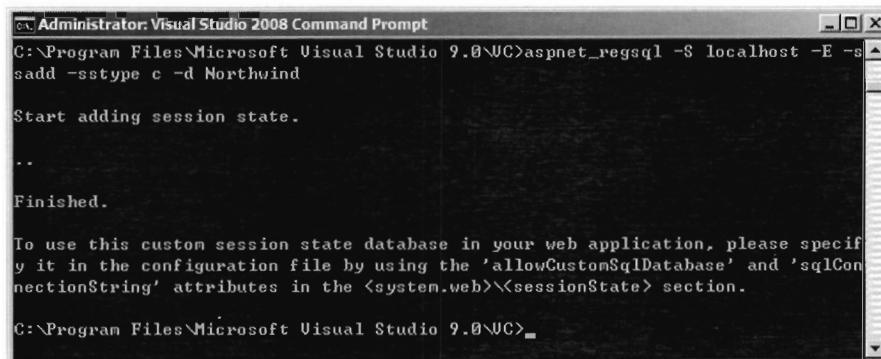
13. 當網頁執行時，先點選「Write Session」將現在時間寫入 Session，接著再點選「Read Session」取出 Session，此時 Session 已經存放在 ASPState 中。

14. 檢視 ASPState 資料庫的 ASPStateTempSessions 資料表：

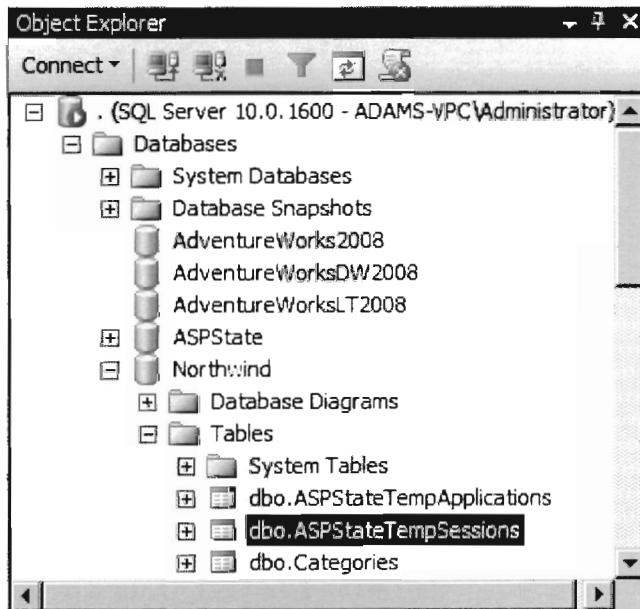
	SessionId	Created	Expires	LockDate	LockDateLocal	LockCookie
>	43kebgue3bbf943	2008-12-07 15:...	2008-12-07 15:...	2008-12-07 15:...	2008-12-07 23:...	2
*	NULL	NULL	NULL	NULL	NULL	NULL

15. 當存放資料的資料表是存放在自訂資料庫時，例如存放在 Northwind 資料庫中，在 Visual Studio 2008 Command Prompt 視窗中輸入：

```
aspnet_regsql.exe -S localhost -E -ssadd -sstype c -d Northwind
```



16. 此時會在 SQL Server 的 Northwind 資料庫中建立用來存放資料的資料表。



17. 此時必須修改 web.config 中，sessionState 的 sqlConnectionString 屬性要加上指定的資料庫，並且要將 allowCustomSqlDatabase 設定為 true。

```
<sessionState mode="SQLServer"
    sqlConnectionString="Data Source=localhost;Integrated security=true;Initial Catalog=Northwind"
    allowCustomSqlDatabase="true"/>
```

18. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。
  19. 當網頁執行時，先點選「Write Session」將現在時間寫入 Session，接著再點選「Read Session」取出 Session，此時 Session 已經存放在 Northwind 中。
  20. 檢視 Northwind 資料庫的 ASPStateTempSessions 資料表：

	SessionId	Created	Expires	LockDate	LockDateLocal	LockCookie
▶	sunqmhl30e3bbf943	2008-12-07 15:...	2008-12-07 15:...	2008-12-07 15:...	2008-12-07 23:...	2
*	NULL	NULL	NULL	NULL	NULL	NULL
◀						

## ViewState物件

- **何謂ViewState**  
將欲儲存的資料存放在ASP.NET表單中，可用來記錄使用者的狀態資料，以一個名叫\_\_ViewState的隱藏欄位存放
- **ViewState對效能上的影響**  
增加資料傳輸量及網頁處理時間
- **使用ViewState物件的基本語法**

Visual Basic ViewState("Sort")= "price"	
C# ViewState["Sort"] = "price";	

## ViewState 物件

ASP.NET 會預設自動維護網頁表單中的資料，將欲儲存的資料存放在一個名叫\_\_ViewState 的隱藏欄位中，在用戶端與伺服器端溝通傳遞時自訂傳送，這個\_\_ViewState 的隱藏欄位可於瀏覽器中檢視原始檔看到。

### 使用 ViewState 物件的基本語法

把需要暫存的資料利用下列語法存入網頁中，將來取出時也可使用相同的語法取得：

Visua l Basic ViewState("Sort")= "price"
---------------------------------------------

C# ViewState["Sort"] = "price";
------------------------------------

### 使用Cookie物件

- 何謂Cookie
- 使用Cookie存放資料需注意
  - 安全性考量
  - 資料型態的考量
  - 使用者端環境的考量
- 使用Cookie基本語法

```
Visual Basic
Response.Cookies("VisitTime").Value=Now
```

```
C#
Response.Cookie["VisitTime"].Value=
DateTime.Now.ToString();
```

## 使用 Cookie

Cookie 是一種相當方便好用的記錄使用者狀態方式，因為使用者狀態資料會記錄在使用者機器上而不會浪費 Server 端的資源，且不會有維護這些資料的負擔。

Cookie 有兩種存放型態，一種是非持續性的 Cookie，資料為放在瀏覽器的記憶體中，另一種是持續性的，資料會存放為一個文字檔，存放在用戶端的電腦中，在程式設計上，兩者的差別最主要在於設定 Cookie 物件的時效性。

### 使用 Cookie 基本語法

直接寫入 Cookie 到使用者端的簡易語法如下：

```
Visua l Basic
Response.Cookies("VisitTime").Value=Now
```

```
C#
Response.Cookies["VisitTime"].Value=DateTime.Now.ToString();
```

## HttpCookie 物件的使用

- 建立HttpCookie物件
 

Visual Basic Dim objCookie As New HttpCookie("User") objCookie.Values.Add("LastVisit", Now.ToString())
C# HttpCookie objCookie = new HttpCookie("User"); objCookie.Values.Add("LastVisit", DateTime.Now.ToString());
- 取得HttpCookie物件
 

Visual Basic Dim objCookie As HttpCookie = Request.Cookies("User")
C# HttpCookie objCookie = Request.Cookies["User"];

## HttpCookie 物件的使用

ASP.NET 提供 `HttpCookie` 類別，讓開發者以物件的設計方式建立及取得 `Cookie` 的資料，再搭配 `Response` 或是 `Request` 物件來寫入或是讀取所設定的 `Cookie` 資料。

### 建立 `HttpCookie` 物件

當建立好 `HttpCookie` 物件之後，可以搭配 `Response` 物件在輸出資料到用戶端時，將 `Cookie` 寫入用戶端。

Visual Basic Dim objCookie As New HttpCookie("user") objCookie.Values.Add("LastVisit", Now.ToString()) objCookie.Values.Add("Username", User.Identity.Name) Response.Cookies.Add(objCookie)
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

C# HttpCookie objCookie = new HttpCookie("user"); objCookie.Values.Add("LastVisit", DateTime.Now.ToString()); objCookie.Values.Add("Username", User.Identity.Name); Response.Cookies.Add(objCookie);
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## 取得 HttpCookie 物件

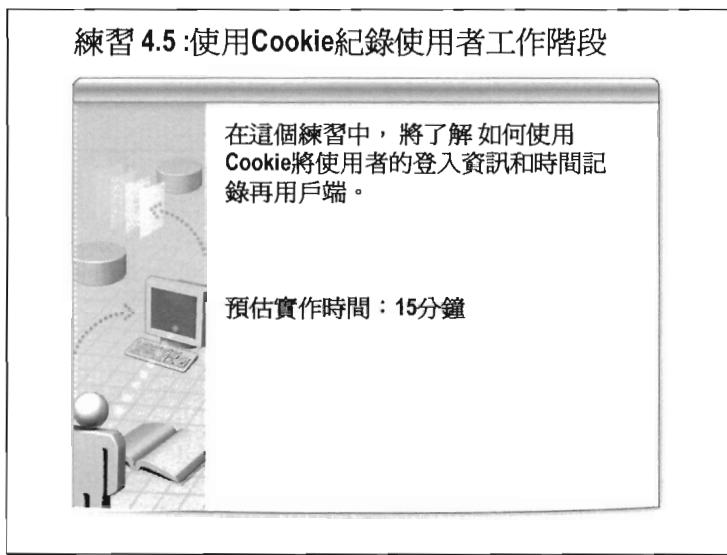
透過 Request 物件接收從 Client 所傳送過來的 Cookie，根據當初所設定的 Cookie 名稱，取出紀錄的 Cookie 值：

Visual Basic

```
Dim objCookie As HttpCookie = Request.Cookies("User")
lblTime.Text = objCookie.Values("LastVisit")
lblUName.Text = objCookie.Values("Username")
```

C#

```
HttpCookie objCookie = Request.Cookies["User"];
lblTime.Text = objCookie.Values["LastVisit"];
lblUName.Text = objCookie.Values["Username"];
```



## 練習 4.5 : 使用 Cookie 紀錄使用者工作階段

### 目的：

在這個練習中，將了解如何使用Cookie將使用者的登入資訊和時間記錄在用戶端，並且設計持續性以及非持續性的Cookie模式。

### 功能描述：

這個練習中會先建立一份HttpCookie物件來記錄使用者的相關資訊，並且使用Response物件寫入到用戶端，接著再使用Request物件將存放在Cookie中的資料取出，呈現在畫面中。

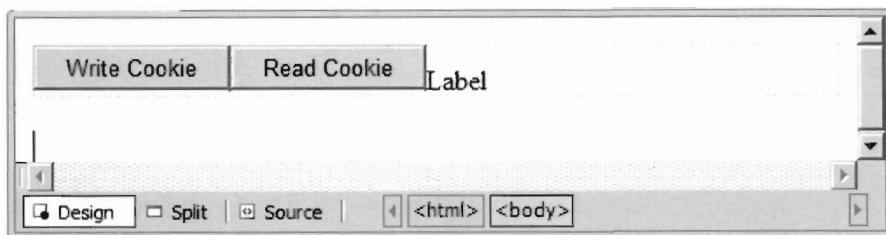
**預估實作時間：15分鐘**

### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動Visual Studio 2008開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod04\_5\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod04\_5」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 從工具箱中，拖拉兩個 Button 控制項，一個 Label 控制項到網頁中，設定 Button1 的 Text 屬性為「Write Cookie」、Button2 的 Text 屬性為「Read Cookie」，設計畫面如下：



5. 使用滑鼠雙擊 Button1，在 Button1 的 Click 事件程序中，將目前使用者的名稱以及存取網頁的時間寫入到 HttpCookie 物件，並且透過 Response 物件寫入到用戶端，程式碼如下：

```
Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim myCookie As New HttpCookie("UserInfo")
    myCookie.Values.Add("Username", User.Identity.Name)
    myCookie.Values.Add("visitTime", Now.ToString())
    Response.Cookies.Add(myCookie)
    Label1.Text = "Cookie 寫入成功"
End Sub
```

```
C#
protected void Button1_Click(object sender, EventArgs e)
{
    HttpCookie myCookie = new HttpCookie("UserInfo");
    myCookie.Values.Add("Username", User.Identity.Name);
    myCookie.Values.Add("visitTime", DateTime.Now.ToString());
    Response.Cookies.Add(myCookie);
    Label1.Text = "Cookie 寫入成功";
}
```

6. 接著，使用滑鼠雙擊 Button2，在 Button2 的 Click 事件程序中，使用 Request 物件將用戶端的 Cookie 資料取出來，程式碼如下：

```

Visual Basic
Protected Sub Button2_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim myCookie As HttpCookie = Request.Cookies("UserInfo")
    If myCookie IsNot Nothing Then
        Label1.Text = "使用者:" + myCookie.Values("Username").ToString()
        Label1.Text += "<br>存取時間:" + myCookie.Values("visitTime").ToString()
    End If
End Sub

```

```

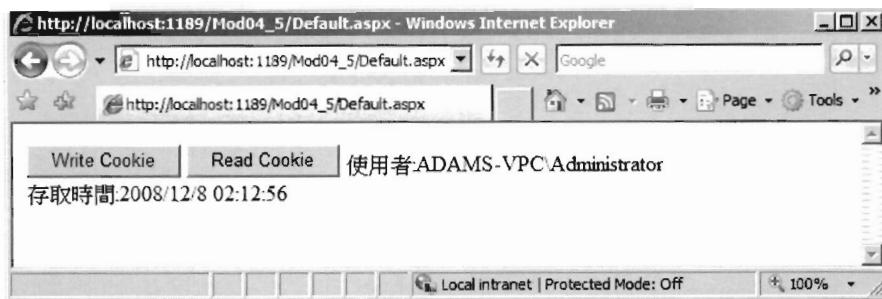
C#
protected void Button2_Click(object sender, EventArgs e)
{
    HttpCookie myCookie = Request.Cookies["UserInfo"];
    if (myCookie!=null)
    {
        Label1.Text = "使用者:" + myCookie.Values["Username"].ToString();
        Label1.Text += "<br>存取時間:" + myCookie.Values["visitTime"].ToString();
    }
}

```

7. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。
8. 先按下畫面的「Write Cookie」按鈕，將使用者的資訊寫入到Cookie中，畫面如下：



9. 接著按下「Read Cookie」按鈕，將剛剛寫入Cookie的資訊讀取出來，顯示在網頁上，畫面如下：



10. 複製上一步驟的瀏覽器視窗網頁網址，並將瀏覽器關閉。
11. 重新啓動一個新的瀏覽器視窗，將上一步驟所複製的網址貼上，瀏覽網頁，並直接按下「Read Cookie」按鈕，會發現畫面並沒有任何的資訊，這是因為目前的 Cookie 為非持續性的 Cookie。
12. 切換到 Microsoft Visual Studio 2008 工具，預設網頁的 Source 視窗，在 Button1 的 Click 事件程序中，加入透過 Cookie 的 Expires 屬性設定時效性，以便產生持續性的 Cookie，例如設定此 Cookie 的有效期限為 10 分鐘，程式碼如下：

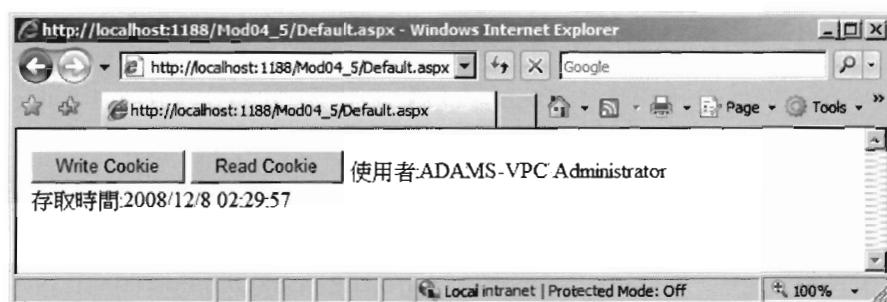
```
Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim myCookie As New HttpCookie("UserInfo")
    myCookie.Values.Add("Username", User.Identity.Name)
    myCookie.Values.Add("visitTime", Now.ToString())
myCookie.Expires = Now.AddMinutes(10)
    Response.Cookies.Add(myCookie)
    Label1.Text = "Cookie 寫入成功"
End Sub
```

```
C#
protected void Button1_Click(object sender, EventArgs e)
{
    HttpCookie myCookie = new HttpCookie("UserInfo");
    myCookie.Values.Add("Username", User.Identity.Name);
    myCookie.Values.Add("visitTime", DateTime.Now.ToString());
myCookie.Expires = DateTime.Now.AddMinutes(10);
    Response.Cookies.Add(myCookie);
    Label1.Text = "Cookie 寫入成功";
}
```

13. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。先按下畫面的「Write Cookie」按鈕，將使用者的資訊寫入到 Cookie 中，接著按下

「Read Cookie」按鈕，將剛剛寫入 Cookie 的資訊讀取出來，顯示在網頁上。

14. 複製上一步驟的瀏覽器視窗網頁網址，並將瀏覽器關閉。
15. 重新啓動一個新的瀏覽器視窗，將上一步驟所複製的網址貼上，瀏覽網頁，並直接按下「Read Cookie」按鈕，會發現畫面將會顯示剛剛所寫入 Cookie 的使用者資料。



## 總結

- 狀態管理機制簡介
- Session物件設計
- Application物件應用
- WebFarm架構時狀態的維護
- ViewState 機制
- Cookie物件設計

# 第五章: AJAX 進階設計

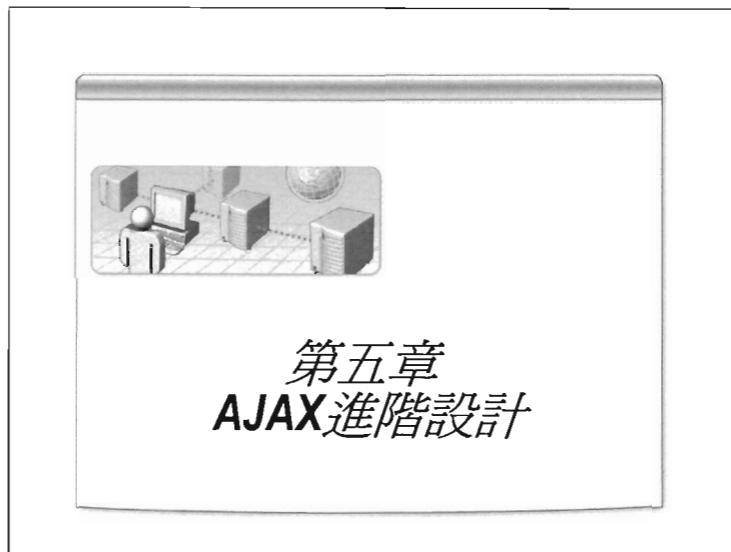
## 本章大綱

ASP.NET AJAX 架構 .....	4
Microsoft Ajax 程式庫.....	6
簡易的捷徑語法.....	8
網頁生命周期與事件.....	9
註冊 Application 物件用戶端事件處理常式.....	11
使用 PageRequestManager 控管部分更新 .....	13
停止或取消非同步 Postback .....	15
練習 5.1 : 防止使用者重複按按鈕.....	16
在部分更新過程更新 UpdatePanel 外的資料 .....	19
練習 5.2 : 使用 RegisterDataItem 傳遞自訂資料 .....	21
AJAX 除錯設定 .....	24
使用 Sys.Debug 類別除錯 .....	26
練習 5.3 : 使用 TraceDump 印出除錯資訊.....	28
使用及管理指令碼 (Script).....	31
將 JavaScript 檔案內嵌在資源組件.....	33
發行版與除錯版指令碼.....	34
練習 5.4 : 部署內嵌在資源組件的 AJAX 指令碼 .....	37
指令碼全球化設計 .....	42
指令碼當地語系化 .....	44
練習 5.5 : 指令碼當地語系化 .....	45
動態建立指令碼.....	49
動態建立指令碼範例.....	50
組合指令碼.....	52

作者：

許薰尹





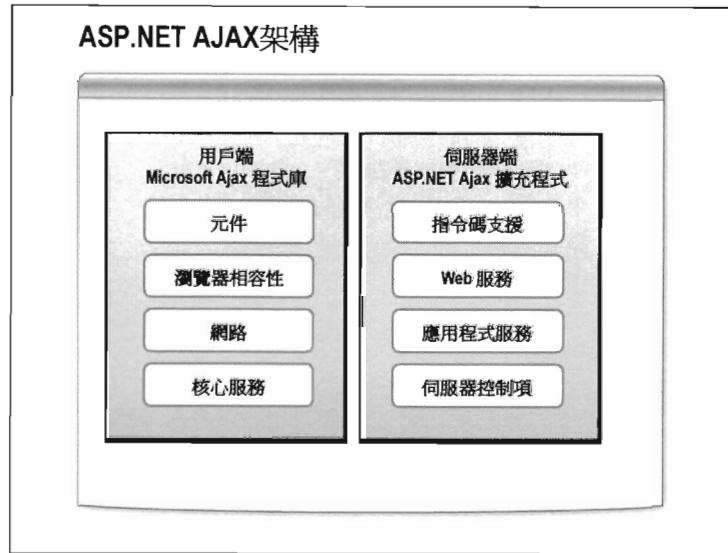
## 課程大綱

- ASP.NET AJAX架構
- JavaScript更多擴充功能
- 用戶端 JavaScript指令碼的偵錯與追蹤功能
- 發行版與除錯版JavaScript指令碼
- 指令碼當地語系化

在這個章節中將介紹如何利用 ASP.NET AJAX Client Library 所提供的擴充功能來設計 AJAX 網頁。您可以透過 JavaScript 來控管網頁非同步更新的動作，攔截相關的事件。

本章介紹以下主題：

- ASP.NET AJAX 架構
- JavaScript 更多擴充功能
- 用戶端 JavaScript 指令碼的偵錯與追蹤功能
- 發行版與除錯版 JavaScript 指令碼
- 指令碼全球化設計
- 指令碼當地語系化



## ASP.NET AJAX 架構

ASP.NET 中的 AJAX 功能架構主要分為兩大個部分，提供完整的開發架構，包含：

- 用戶端指令碼程式庫 (Client-Script Libraries)
- 伺服器元件(server components)

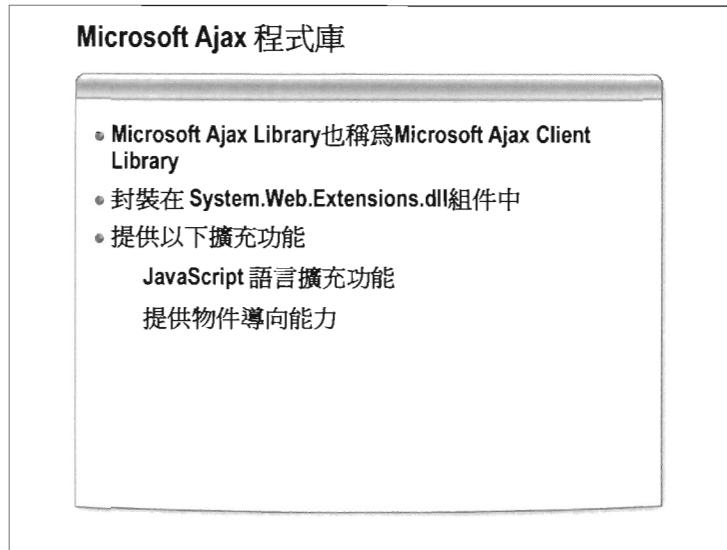


用戶端指令碼程式庫 (Client-Script Libraries) 包含：

- 元件：提供非視覺化元件，擴充 DOM 元素行為的 Behavior 元件、以及自訂控制項。
- 瀏覽器相容性：提供常用的瀏覽器(包括 Microsoft Internet Explorer、Mozilla Firefox 等)與 AJAX 指令碼的相容性。
- 網路：處理瀏覽器中的指令碼與伺服端程式之間的溝通細節，並可進行非同步呼叫。
- 核心服務：由許多 JavaScript (.js) 檔案所組成，提供 JavaScript 具備物件導向的功能。另外也提供 Sys.Debug 類別以支援除錯、追蹤機制。以及全球化能力，以建立豐富的多國語言程式。

伺服器元件 (server components) 包含：

- 指令碼支援：ASP.NET 會由伺服端送出指令碼到瀏覽器，以便建立啓用 AJAX 的網頁，以支援 JavaScript 擴充功能，如物件導向、介面...等，以及網頁局部生成(Partial- Page Rendering) 功能。ASP.NET 指令碼包含發行版和除錯版。
- Web 服務：您可以使用 JavaScript 來呼叫 ASP.NET Web 服務 (.asmx) 或者是 Windows Communication Foundation (WCF) 服務 (.svc)，以設計服務導向應用程式。
- 應用程式服務：整合 ASP.NET 內建的表單驗證 (Forms Authentication)、角色(Role)和使用者設定檔(Profile)服務。
- 伺服器控制項：提供許多伺服器控制項來建立 AJAX 網頁，如 ScriptManager、UpdatePanel、UpdateProgress 與 Timer 控制項等等。



## Microsoft Ajax 程式庫

Microsoft Ajax 程式庫 (Microsoft Ajax Library，也稱為 Microsoft Ajax Client Library) 是以 JavaScript 寫成的，這些 JavaScript 封裝在 System.Web.Extensions.dll 組件中，預設.NET Framework 安裝時會將它安裝到 GAC 之中，請勿將它置於網站的 bin 子目錄。提供以下擴充功能：

### JavaScript 語言擴充功能

增強 JavaScript 型別系統，為陣列、布林、字串、數字提供更多功能。提供的物件包含：

- Array：擴充原生的陣列物件。
- Boolean：擴充原生的布林物件。
- Date：擴充原生的日期物件。
- Error：擴充原生的 Error 物件，以做錯誤處理。
- Function：擴充原生的 Function 物件，以定義類別、命名空間、委派(Delegate)...等。
- Number：擴充原生的 Number 物件。

- Object：擴充原生的 Object 物件，以取得型別資訊。
- RegExp：擴充原生的 RegExp 物件，以支援規則運算式。
- String：擴充原生的 String 物件，提供更多字串處理函式。

### 提供物件導向能力

為 JavaScript 提供命名空間(Namespace)、繼承(Inheritance)、介面(Interface)、列舉型別(Enum)與反映(Reflection)的機制。

### 簡易的捷徑語法

- 提供更簡易的捷徑語法，來加速程式的開發

- **JavaScript 範例：**

```
JavaScript
document.getElementById("TextBox1").value
```

- 對等捷徑語法：

```
JavaScript
$get("TextBox1").value
```

- 常用的捷徑語法

\$create  
\$find

### 簡易的捷徑語法

Microsoft Ajax 程式庫提供更簡易的捷徑語法，來加速程式的開發，例如原來透過 JavaScript 與 DOM 模型存取文字方塊的值會使用以下語法：

```
document.getElementById("TextBox1").value
```

透過 Microsoft Ajax 程式庫，可以使用\$get 改寫為：

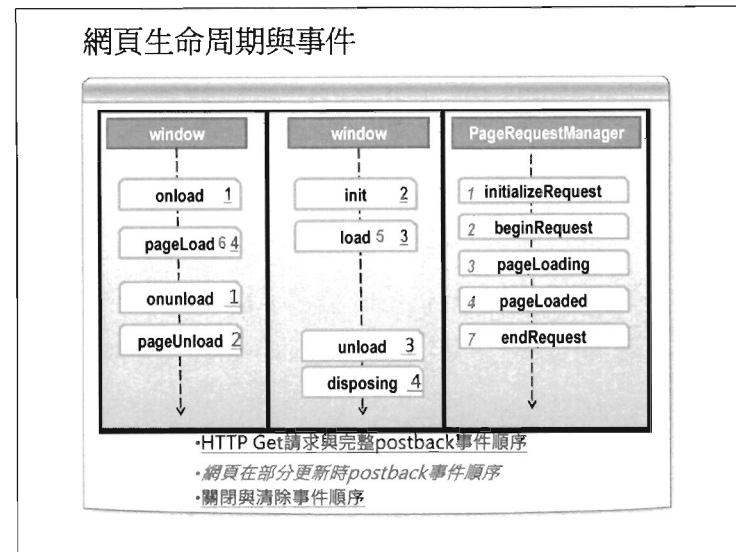
```
$get("TextBox1").value
```

此外，常用的捷徑語法包含：

- \$create：建立 Microsoft AJAX Library 元件。
- \$find：找尋元件

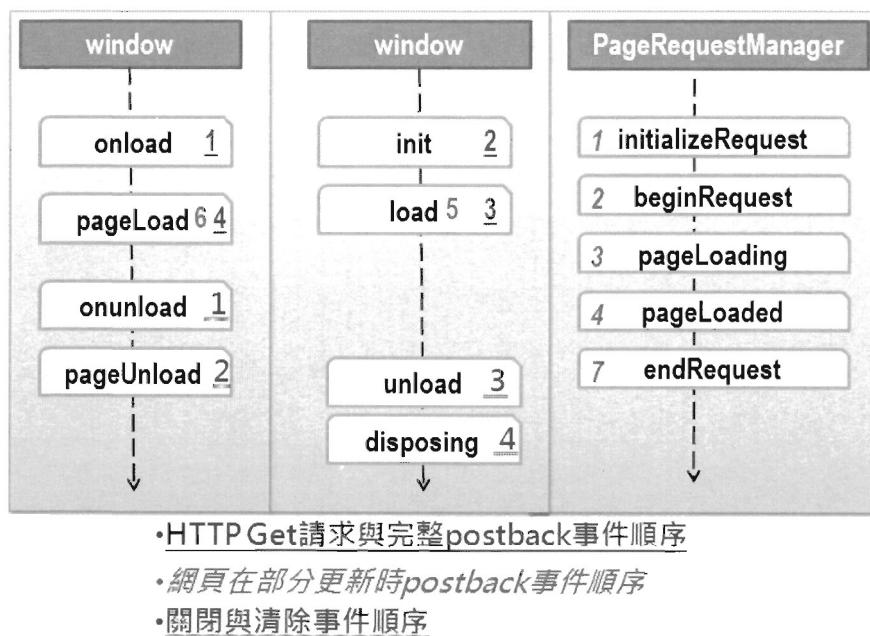
\$get 使用範例如下，利用\$get 取得網頁中 Label1 控制項，並把文字設為「new」字串：

```
<script type="text/JavaScript">
    UpdateData("Label1", "new");
    function UpdateData (l, t) {
        var lbl = $get(l);
        lbl.innerHTML = t;
    }
</script>
```



## 網頁生命周期與事件

ASP.NET AJAX 網頁在瀏覽器上執行時，會觸發類似在伺服器端執行的生命週期事件，以便於讓您操作使用者介面，或管理元件。觸發的事件可能是 HTML DOM Window 物件觸發的；也可能是由 Microsoft AJAX Library 中的 Sys.Application 物件或 Sys.WebForms.PageRequestManager 物件所觸發的。事件發生順序請參考下圖：



Application 類別是繼承自 Sys.Component 類別，當網頁包含 ScriptManager 控制項，網頁就會自動建立 Application 物件。

Application 物件觸發的事件包含：

- Sys.Application.init 事件：在載入 JavaScript 之後，但尚未建立任何物件之前觸發。
- Sys.Application.load 事件：當網頁第一次在瀏覽器上執行，或網頁執行同步 Postback 動作時，將會觸發 Application 的 Load 事件，此事件後，所有的用戶端指令碼(Script)與元件(Component)都已載入，可以直接使用。
- Sys.Application.unload 事件：在用戶端所有物件 dispose 之前觸發。
- Sys.Component.disposing 事件：當元件的 dispose 方法被呼叫時觸發。

### 註冊 Application 物件用戶端事件處理常式

- 註冊語法  
add\_事件名稱

- 移除語法  
remove\_事件名稱

add\_事件名稱 和 remove\_事件名稱

#### JavaScript

```
<script type="text/javascript">
Sys.Application.add_init(init);
function init(sender) {
    alert('init');
}
</script>
```

### 註冊 Application 物件用戶端事件處理常式

若要新增或移除由 Application 物件所觸發的事件之事件處理常式，可以使用此類別的 add\_事件名稱 和 remove\_事件名稱方法。例如以下範例說明如何註冊 Application 物件的 init 事件處理常式。

```
<script type="text/javascript">
Sys.Application.add_init(Init);
function Init(sender) {
    alert('init');
}
</script>
```

### 撰寫 Application Load 與 Unload 事件處理常式

Application 物件的 load 和 unload 事件處理常式，不需要透過程式來進行註冊，只要建立名為 pageLoad 和 pageUnload 的函式就會在事件發生時自動地執行。下列範例程式碼說明如何撰寫 Application 物件的 load 事件與 unload 事件處理常式：

```
<script type="text/javascript">
function pageLoad() {
    alert('pageLoad');
}

function pageUnload() {
```

```
    } alert('pageUnload');
  </script>
```

### 使用 **PageRequestManager** 控管部分更新

- **PageRequestManager** 控管部分更新
- **PageRequestManager** 類別應用在：
  - 管理多個非同步 postback 的執行順序。
  - 提供視覺化的訊息提示。
  - 顯示執行進度，或取消 Postback 動作。
  - 可自訂錯誤訊息。
- 擷取 **PageRequestManager** 事件
  - EnablePartialRendering 屬性設為 true
  - 網頁中有 ScriptManager Web 伺服器控制項

## 使用 **PageRequestManager** 控管部分更新

一般設計 AJAX 網頁時，只要在網頁中使用 ScriptManager 及 UpdatePanel Web 伺服器控制項，不需要撰寫程式碼，就可以用網頁部分更新能力。不過，有時您可能需要手動管理部分更新。Microsoft AJAX Library 中包含一個 PageRequestManager 類別，主要是用來控管網頁部分更新的動作，透過 PageRequestManager 類別提供的屬性、方法與相關的事件，能以便用用戶端指令碼，在必要時手動管理網頁是否進行部分更新。

PageRequestManager 類別可以應用在：

- 管理多個非同步 postback 的執行順序，可以客製化瀏覽器中的網頁部分更新的行為。
- 提供視覺化的訊息提示。
- 顯示執行進度，或取消 Postback 動作。
- 可自訂錯誤訊息。

### PageRequestManager 類別與事件

PageRequestManager 類別可應用在處理用戶端頁面生命週期過程中觸發的事件。但請注意，要使用 PageRequestManager 類別時，網頁中一定要有 ScriptManager 伺服器控制項，且 ScriptManager 控制項的 EnablePartialRendering 屬性得設為 true。然後就可以利用用戶端指令碼來處理由 PageRequestManager 類別觸發的事件。事件包含：

- initializeRequest：在非同步 postback 的請求(Request)初始化之前觸發。
- beginRequest：在非同步 postback 的請求送到伺服器之前觸發。
- pageLoading：在最近一次的非同步 postback 回應收到之後，尚未更新網頁之前觸發。
- pageLoaded：網頁部分更新動作完成後觸發。
- endRequest：請求完成後觸發。

您可以使用 ScriptManager 控制項的 RegisterDataItem 方法，在非同步 postback 動作發生時，額傳送外資料，以便用來更新 UpdatePanel 之外的控制項。。

### 停止或取消非同步 Postback

- 取消方式有兩種：

使用 abortPostBack 方法

設定 cancel 屬性為 true

```
JavaScript
<script type="text/javascript">
Sys.WebForms.PageRequestManager.getInstance().add_initializeRequest(InitializeRequest);
function InitializeRequest(sender, args) {
    var prm = Sys.WebForms.PageRequestManager.getInstance();
    if (prm.get_isInAsyncPostBack() &
        args.get_postBackElement().id == 'Button2') {
        prm.abortPostBack();
    }
}
</script>
```

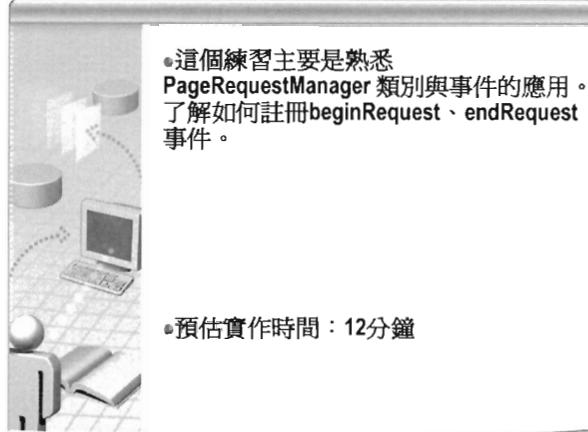
### 停止或取消非同步 Postback

若要執行停或取消非同步 Postback 的執行，可以呼叫 PageRequestManager 類別的 abortPostBack 方法。或在 initializeRequest 事件將 PageRequestManager 類別的 cancel 屬性設定為 true。參考以下呼叫 abortPostBack 方法的範例：

```
<script type="text/javascript">
Sys.WebForms.PageRequestManager.getInstance().add_initializeRequest(InitializeRequest);

function InitializeRequest(sender, args)
{
    var prm = Sys.WebForms.PageRequestManager.getInstance();
    if (prm.get_isInAsyncPostBack() &
        args.get_postBackElement().id == 'Button2') {
        prm.abortPostBack();
    }
}
</script>
```

### 練習 5.1 : 防止使用者重複按按鈕



### 練習 5.1 : 防止使用者重複按按鈕

#### 目的：

這個練習主要是熟悉 PageRequestManager 類別與事件的應用。了解如何註冊 beginRequest、endRequest 事件。

#### 功能描述：

這個練習會在網頁中加入指令碼，在網頁進行非同步更新過程中，使用 PageRequestManager 類別與事件來偵測非同步作業是否已開始，若已開始則將按鈕停用，防止使用者重複按按鈕。

#### 預估實作時間：12 分鐘

#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啟動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod05\_1\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod05\_1」。

3. 自主選單「Web Site」下「Add New Item...」，選取「AJAX Web Form」，清除「Place code in separate file」核取方塊，新增一個 AJAX 網頁，使用預設的檔名命名。
4. 從「Toolbox」工具箱中拖拉一個 UpdatePanel 控制項到網頁中 ScriptManager 控制項下方。
5. 從「Toolbox」工具箱中拖拉一個 Button，一個 Label，到 UpdatePanel 控制項到之中。將 Button 控制項的 Text 屬性設定為「Get Data」。將 Label 控制項的 Text 屬性設定為空白。目前標籤看起來如下：

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
<ContentTemplate>
<asp:Button ID="Button1" runat="server"
Text="GetData" />
<br />
<asp:Label ID="Label1" runat="server"></asp:Label>
</ContentTemplate>
</asp:UpdatePanel>
```

6. 從「Toolbox」工具箱中拖拉一個 UpdateProgress 控制項到 ScriptManager 控制項下方。在 UpdateProgress 控制項 ProgressTemplate 項目中，輸入「作業處理中,請稍待 ...」；目前標籤看起來如下：

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server">
<ProgressTemplate>
    作業處理中,請稍待 ...
</ProgressTemplate>
</asp:UpdateProgress>
```

7. 切換到設計畫面，雙擊網頁中 Button 控制項，產生 Click 事件處理常式，加入以下程式碼：

Visual Basic System.Threading.Thread.Sleep(3000) Label1.Text = System.DateTime.Now.ToString()
-----------------------------------------------------------------------------------------------------

C# System.Threading.Thread.Sleep(3000); Label1.Text = System.DateTime.Now.ToString();
---------------------------------------------------------------------------------------------

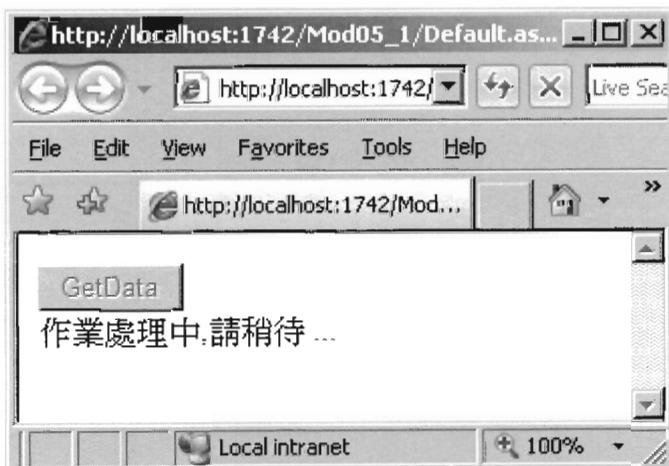
8. 在網頁最下方，加上以下程式碼區塊，利用 PageRequestManager 註冊攔截 beginRequest、endRequest 事件，在 beginRequest 事件發生時，將 Button1 的 disabled 屬性設為 true；在 endRequest 事件發生時，將 Button1 的 disabled 屬性設為 false：

```
JavaScript
<script type="text/javascript">
    var elem
    function pageLoad() {
        var pm =
            Sys.WebForms.PageRequestManager.getInstance();
        pm.add_beginRequest(OnBeginRequest);
        pm.add_endRequest(OnEndRequest);

    }
    function OnBeginRequest(sender, args) {
        elem = args.get_postBackElement();

        if (elem.id == "Button1")
            $get("Button1").disabled = true;
    }
    function OnEndRequest(sender, args) {
        if (elem.id == "Button1")
            $get("Button1").disabled = false;
    }
</script>
```

9. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。當按鈕被按下時，按鈕就無法重覆點選，參考：



### 在部分更新過程更新UpdatePanel外的資料

- 允許自訂資料從伺服器傳送至用戶端
- 使用 **ScriptManager** 的 **RegisterDataItem** 方法  
    非同步 Postback 的過程中才能夠呼叫
- 資料只能在 **PageRequestManager** 物件的以下事件處理常式存取
  - pageLoading
  - pageLoaded
  - endRequest

### 在部分更新過程更新 UpdatePanel 外的資料

在網頁進行非同步 Postback 的部分更新動作時，若想要自訂一些資料從伺服器傳送至用戶端，可以使用 ScriptManager 的 RegisterDataItem 方法。RegisterDataItem 方法只有在非同步 Postback 的過程中才能夠呼叫。

使用 RegisterDataItem 方法來註冊的資料項目只有在 PageRequestManager 物件的 pageLoading、pageLoaded 和 endRequest 等事件處理常式中才可存取。

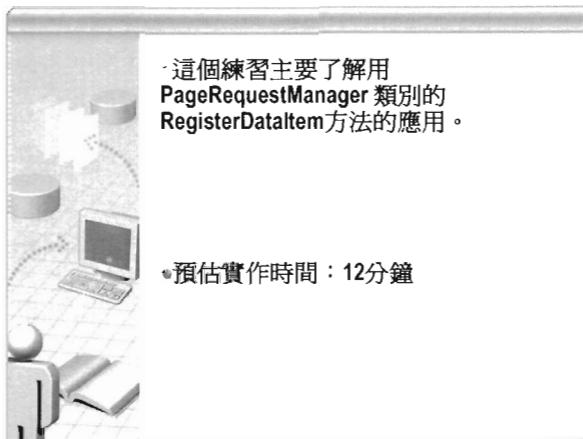
```
Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs)
    Label1.Text = "L1"
    Label2.Text = "L2"
    ScriptManager1.RegisterDataItem(Label3, "L3")
End Sub
```

```
C#
protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "L1";
    Label2.Text = "L2";
    ScriptManager1.RegisterDataItem(Label3, "L3");
```

{

```
JavaScript
<script type="text/javascript" language="javascript">
    Sys.WebForms.PageRequestManager.getInstance().add_pageLoading(PageLoadingHandler);
    function PageLoadingHandler(sender, args) {
        var dataItems = args.get_dataItems();
        if ($get('Label3') !== null)
            $get('Label3').innerHTML = dataItems['Label3'];
    }
</script>
```

### 練習 5.2 : 使用 RegisterDataItem 傳遞自訂資料



### 練習 5.2 : 使用 RegisterDataItem 傳遞自訂資料

目的：

這個練習主要了解用 PageRequestManager 類別的 RegisterDataItem 方法的應用。

功能描述：

這個練習會在網頁進行非同步 Postback 的部分更新動作時，使用 ScriptManager 的 RegisterDataItem 方法傳遞自訂資料，以在非同步更新動作完成後，並在 pageLoading 事件處理常式中，更新 UpdatePanel 之外的控制項資料。

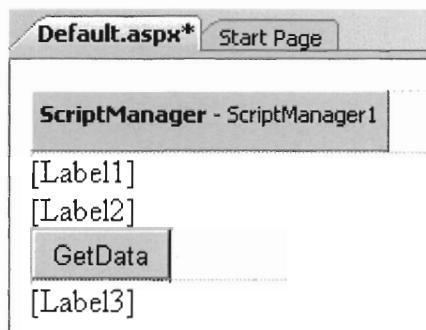
預估實作時間：10 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選

取「\U9544\Practices\VB 或 CS\Mod05\_2\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod05\_2」。

3. 自主選單「Web Site」下「Add New Item...」，選取「AJAX Web Form」，清除「Place code in separate file」核取方塊，新增一個 AJAX 網頁，使用預設的檔名命名。
4. 從「Toolbox」工具箱中拖拉一個 UpdatePanel 控制項到網頁中 ScriptManager 控制項下方。
5. 從「Toolbox」工具箱中拖拉兩個 Label，一個 Button，到 UpdatePanel 控制項到之中。拖拉一個 Label 到 UpdatePanel 控制項下方。
6. 將 Button 控制項的 Text 屬性設定為「Get Data」。將三個 Label 控制項的 Text 屬性設定為空白。畫面看起來如下：



7. 切換到設計畫面，雙擊網頁中 Button 控制項，產生 Click 事件處理常式，加入以下程式碼：

<b>Visual Basic</b> <pre>Label1.Text = System.DateTime.Now.ToShortDateString() Label2.Text = System.DateTime.Now.ToShortTimeString() ScriptManager1.RegisterDataItem(Label3, System.DateTime.Now. ToUniversalTime().DayOfYear.ToString())</pre>
----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

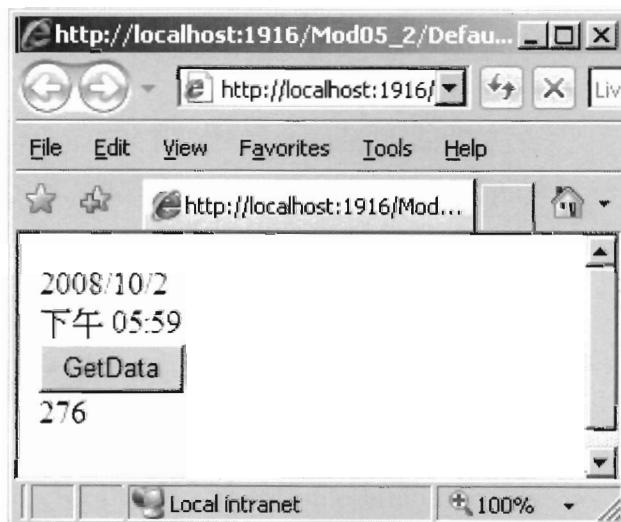
<b>C#</b> <pre>Label1.Text = System.DateTime.Now.ToShortDateString(); Label2.Text = System.DateTime.Now.ToShortTimeString(); ScriptManager1.RegisterDataItem(Label3, System.DateTime.Now. ToUniversalTime().DayOfYear.ToString());</pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

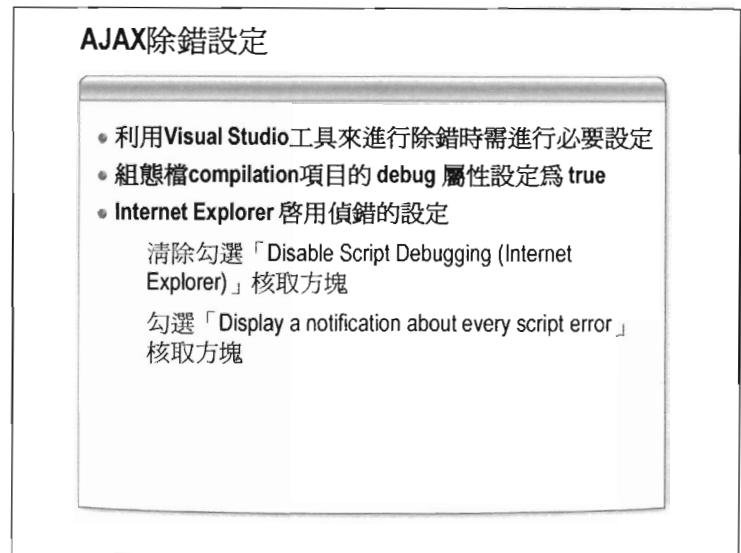
8. 在網頁最下方，加上以下程式碼區塊，利用 PageRequestManager 註冊攔截 pageLoading 事件，在叫用 get\_dataItems 方法取的傳遞的資料，更新 UpdatePanel 外 Label3 控制項的顯示文字：

```
JavaScript
<script type="text/javascript" language="javascript">
    Sys.WebForms.PageRequestManager.getInstance().add_pageLoading(PageLoadingHandler);
    function PageLoadingHandler(sender, args) {
        var dataItems = args.get_dataItems();
        if ($get('Label3') != null)
            $get('Label3').innerHTML = dataItems['Label3'];

    }
</script>
```

9. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。當按鈕被按下時，Label1、Label2、Label3 會分別顯示時間、日期與今天為本年的第幾天：





## AJAX 除錯設定

ASP.NET AJAX 架構提供更易於偵錯的開發模型，您可以透過工具或 Sys.Debug 類別來輔助除錯。你可以將追蹤訊息輸出以便於了解發生錯誤的問題點。為了要利用 Visual Studio 工具來進行除錯，您需要在網站組態檔與瀏覽器中進行適當的設定。

### 設定網站組態檔以進行偵錯

若要啓用偵錯功能，需要在網站組態檔將 compilation 項目的 debug 屬性設定為 true。例如：

```
<configuration>
  <system.web>
    <compilation debug="true">
    </compilation>
  </system.web>
<configuration>
```

### Internet Explorer 啓用偵錯的設定

從 Internet Explorer，點選「Tools」功能表上的「Internet Options」。項目，從「Advanced」頁，清除勾選「Disable Script

Debugging (Internet Explorer)」核取方塊，並勾選「Display a notification about every script error」核取方塊。

**使用 Sys.Debug 類別除錯**

- 顯示追蹤訊息、使用判斷提示以及中斷偵錯工具
- 常用的方法：
  - Sys.Debug.assert
  - Sys.Debug.fail
  - Sys.Debug.traceDump
- 使用 textarea 項目檢視偵錯訊息  
將 ID 設為 TraceConsole

```
JavaScript
<script type = "text/javascript" >
function dump()
{
  var a=[];
  a= new Array(10,20,30);
  Sys.Debug.traceDump(a, "info");
}
function cleardata()
{
  Sys.Debug.clearTrace();
}</script>
```

## 使用 **Sys.Debug** 類別除錯

ASP.NET AJAX 提供 **Sys.Debug** 類別來偵錯用戶端應用程式。藉由呼叫 **Sys.Debug** 類別的方法，在網頁的結尾以可閱讀的格式顯示物件、顯示追蹤訊息、使用判斷提示以及中斷偵錯工具。

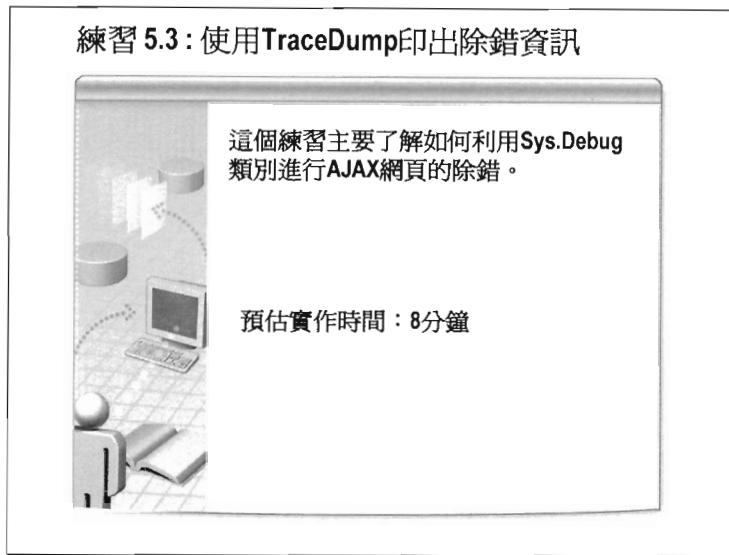
**Sys.Debug** 類別可以用來除錯程式、啓動除錯工具，也可以輸出錯誤訊息。常用的方法：

- **Sys.Debug.assert** 方法：在評估判斷式為 `false` 的情況下，則顯示錯誤訊息。
- **Sys.Debug.clearTrace** 方法：清除所有追蹤訊息。
- **Sys.Debug.fail** 方法：跳出是否除錯的視窗，以選取除錯工具，然後進入 Visual Studio 中斷模式。
- **Sys.Debug.trace** 方法：輸出錯誤訊息。
- **Sys.Debug.traceDump** 方法：將物件的資訊輸出。

以下範例程式碼，展示如何利用 `traceDump` 印出陣列的資訊；以及清除追蹤訊息。

```
JavaScript
<script type = "text/javascript" >
    function dump()
    {
        var a=[];
        a= new Array(10,20,30);
        Sys.Debug.traceDump(a,"info");
    }
    function cleardata()
    {
        Sys.Debug.clearTrace();
    }
</script>
```

如果在 Visual Studio 開發工具進行除錯，錯誤訊息會出現在「Output」視窗中；如果不是使用 Visual Studio，則可以在網頁中加入一個 HTML textarea 項目，將其 ID 設為 TraceConsole，就可以在瀏覽器中 textarea 項目檢視偵錯訊息。



### 練習 5.3 : 使用 TraceDump 印出除錯資訊

目的：

這個練習主要了解如何利用 Sys.Debug 類別進行 AJAX 網頁的除錯。

功能描述：

練習中將建立一個陣列，設定陣列中的元素值，然後利用 Sys.Debug 類別 TraceDump 方法，將陣列物件的相關資訊，輸出到 HTML textarea'項目中，以檢視其中的內容。

預估實作時間：8分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod05\_3\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod05\_3」。

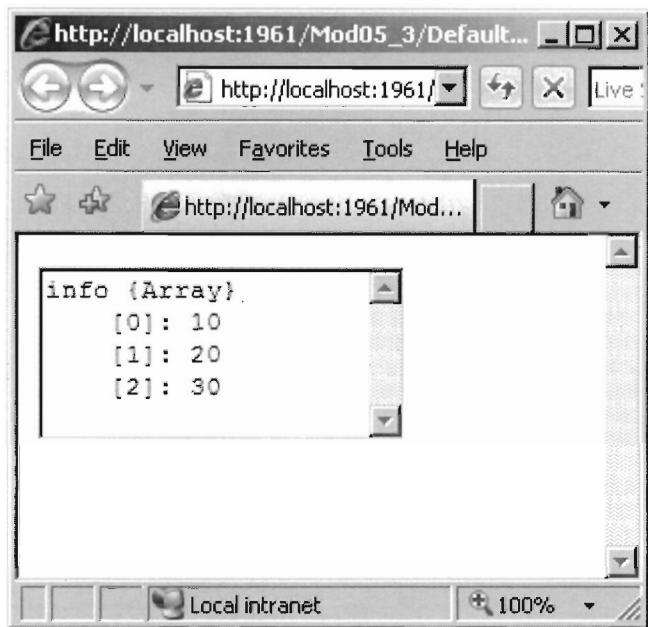
3. 自主選單「Web Site」下「Add New Item...」，選取「AJAX Web Form」，清除「Place code in separate file」核取方塊，新增一個 AJAX 網頁，使用預設的檔名命名。
4. 從「Toolbox」工具箱 HTML 頁中拖拉一個 textarea 控制項到網頁中 ScriptManager 控制項下方。
5. 將 textarea 控制項其 ID 設為 TraceConsole，將 rows 設為 5。目前標籤看起來如下：

```
<textarea id="TraceConsole" cols="20" rows="5" ></textarea>
```

6. 在網頁預設的 Java Script 區塊中 pageLoad 事件處理常式加入以下程式碼：

```
JavaScript
<script type="text/javascript">
    function pageLoad() {
        var a = [];
        a = new Array(10, 20, 30);
        Sys.Debug.traceDump(a, "info");
    }
</script>
```

7. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。當網頁載入時，就會顯示陣列的相關資訊。



## 使用及管理指令碼 (Script)

- ScriptManager 控制項主要是用來管理指令碼
- 利用 ScriptReference 參照到指令碼檔案
- 部署指令碼方式

    靜態檔案

        內嵌在資源組件 (resource assembly)

- 靜態檔案範例

```
JavaScript
<asp:ScriptManager ID="ScriptManager1" runat="server">
    <Scripts>
        <asp:ScriptReference Path="~/test/JScript.js" />
    </Scripts>
</asp:ScriptManager>
```

## 使用及管理指令碼 (Script)

ASP.NET AJAX 應用程式之中，經常會使用到許多用戶端指令碼 (JavaScript)。在 ASP.NET 架構下，可以使用 ScriptManager 控制項來管理這些指令碼，以及您自行撰寫的 JavaScript。

### 在 ASP.NET 網頁使用 JavaScript

在 ASP.NET 網頁中若要使用外部的 JavaScript 檔案，可以在 `<script>` 區塊中宣告，例如：

```
<script type="text/javascript" src="JScript.js"></script>
```

但是這些 JavaScript 將無法參與網頁部分更新動作也無法使用某些 Microsoft AJAX Library 元件。要在 AJAX 網頁中使用到指令碼，需要利用網頁中的 ScriptManager 控制項進行註冊，利用 ScriptReference 參照到想使用的指令碼檔案。通常可選擇兩種方式來部署指令碼，一是在應用程式中，直接撰寫一個 JavaScript 檔案(靜態檔案)，然後在 AJAX 網頁，ScriptManager 的標籤中，利用 Path 來描述參考的 JavaScript 檔名，例如以下的標籤中描述 ScriptManager 參考到 JScript.js 檔案：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
    <Scripts>
        <asp:ScriptReference Path="~/test/JScript.js" />
    </Scripts>
</asp:ScriptManager>
```

為了避免載入 Script 完成之前就使用到 AJAX 物件導致發生錯誤，因此，每個檔案檔尾應加上一行程式呼叫 Sys.Application.notifyScriptLoaded 方法，通知 Script 檔案已經載入完成：

```
JavaScript
if (typeof(Sys) !== 'undefined') Sys.Application.notifyScriptLoaded
();
```

### 將 JavaScript 檔案內嵌在資源組件

- 較容易進行部署
- 不需要叫用 `Sys.Application.notifyScriptLoaded` 方法
- 使用 `Name` 指定 JavaScript 檔案名稱
- 使用 `Assembly` 指定組件名稱

```
JavaScript
<asp:ScriptManager ID="ScriptManager1" runat="server">
    <Scripts>
        <asp:ScriptReference Name="MyScript.embeddedSample.js"
            Assembly="MyScript" />
    </Scripts>
</asp:ScriptManager>
```

### 將 JavaScript 檔案內嵌在資源組件

在撰寫 Ajax 應用程式時，將 JavaScript 檔案內嵌在資源組件 (resource assembly) 之中較容易進行部署。如此使用者就不會不小心任何地開啟這些包含 JavaScript 的 js 檔案，或誤刪了它們。若要將 .js 檔案以資源來嵌入在.NET 的組件(Assembly)之中，就不需要叫用 `Sys.Application.notifyScriptLoaded` 方法。然後改用 `Assembly` 屬性來描述參考的組件，使用 `Name` 指定 JavaScript 檔案名稱：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
    <Scripts>
        <asp:ScriptReference Name="MyScript.embeddedSample.js"
            Assembly="MyScript" />
    </Scripts>
</asp:ScriptManager>
```

### 發行版與除錯版指令碼

- 指令碼分為發行版(Release)與除錯版(Debug)兩種
- 除錯版的名稱規則  
發行版的檔案名稱.debug.js  
Ex:發行版的名稱為JScript.js；除錯版的名稱為JScript.debug.js
- 設定要載入的指令碼版本  
ScriptMode屬性:包含Auto、Debug、Inherits與Release設定

### 發行版與除錯版指令碼

ScriptManager 控制項管理的指令碼分為發行版與偵錯版兩種，若發行版的名稱為 JScript.js；則偵錯版的名稱為 JScript.debug.js。您可以利用 ScriptMode 屬性來設定想使用的版本。

Ajax 網頁之中的 ScriptManager 會按照網站的設定，自動自組件中取出發行版(Release)或是除錯版(Debug)的指令碼檔案。ScriptManager 透過 ScriptResource.axd 取出指令碼內容之後，會自動產生一行 Sys.Application.notifyScriptLoaded 指令，通知 ASP.NET AJAX 指令碼已經載入。相較於使用 Path 設定指令碼的 Ajax 網頁，ScriptManager 則不會產生這一行通知的指令，程式設計師應該自行撰寫以確保程式能正確執行。

#### 了解 **ScriptMode** 的設定

ScriptManager 與 ScriptReference 都有 ScriptMode 屬性，可以用來設定要載入的指令碼版本。ScriptMode 包含四種設定：Auto、Debug、Inherits 與 Release。假設目前 ScriptManager 與 ScriptReference 皆使用預設值：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
<Scripts>
<asp:ScriptReference Name="MyScript.JScript.js"
Assembly="MyScript" />
</Scripts>
</asp:ScriptManager>
```

然後將 web.config 檔案 compilation 區段，將 debug 為 false：

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

執行程式，將執行發行版指令。

修改 web.config，將 compilation 區段，debug 為 true：

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

執行程式，將執行除錯版指令。

如果將 ScriptReference、ScriptManager 的 ScriptMode 都設定為「Debug」，則不管 web.config 檔案中 compilation 區段，將 debug 設為 true 或 false，都會得到除錯版：

```
<asp:ScriptManager ID ="ScriptManager1" runat ="server"
ScriptMode = "Debug">
<Scripts>
<asp:ScriptReference Name ="MyScript.JScript.js"
Assembly = "MyScript" ScriptMode = "Debug" />
</Scripts>
</asp:ScriptManager>
```

若在 ScriptManager 設定 ScriptMode 為「Debug」，在 ScriptReference 設 ScriptMode 為「Inherit」，得到 Debug 版 (ScriptReference 繼承 ScriptManager)：

```
<asp:ScriptManager ID = "ScriptManager1" runat = "server"
ScriptMode = "Debug">
<Scripts>
<asp:ScriptReference Name = "MyScript. JScript.js"
Assembly = "MyScript" ScriptMode = "Inherit"/>
</Scripts>
</asp:ScriptManager>
```

在 ScriptManager 設定 ScriptMode 為「Release」，在 ScriptReference 設 ScriptMode 為「Inherit」，得到 Release 版 (ScriptReference 繼承 ScriptManager)：

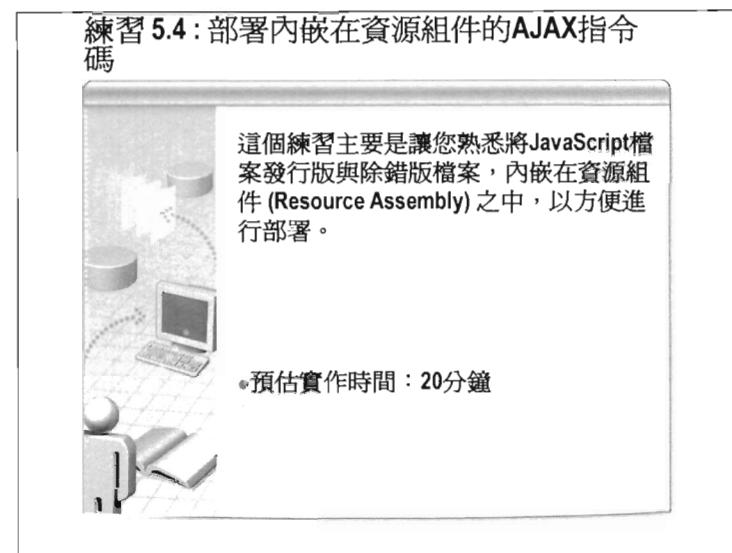
```
<asp:ScriptManager ID="ScriptManager1" runat="server"
    ScriptMode="Release">
    <Scripts>
        <asp:ScriptReference Name="MyScript.JScript.js"
            Assembly="MyScript" ScriptMode="Inherit"/>
    </Scripts>
</asp:ScriptManager>
```

在 ScriptManager 設定 ScriptMode 為「Auto」，在 ScriptReference 設 ScriptMode 為「Inherit」，則當 Web.config 設 debug 為 true 時，得到除錯版指令碼；當 Web.config 設 debug 為 false 時，將得到發行版 (Release 版) 指令碼。

```
<asp:ScriptManager ID = "ScriptManager1" runat = "server"
    ScriptMode = "Auto">
    <Scripts>
        <asp:ScriptReference Name = "MyScript. JScript.js"
            Assembly = "MyScript" ScriptMode = "Inherit"/>
    </Scripts>
</asp:ScriptManager>
```

在 ScriptManager 設定 ScriptMode 為「Auto」，在 ScriptReference 設 ScriptMode 為「Auto」，則當 Web.config 設 debug 為 true 時，得到除錯版指令碼；當 Web.config 設 debug 為 false 時，將得到發行版 (Release 版) 指令碼。

```
<asp:ScriptManager ID="ScriptManager1" runat="server"
    ScriptMode="Auto">
    <Scripts>
        <asp:ScriptReference Name="MyScript. JScript.js"
            Assembly="MyScript" ScriptMode="Auto"/>
    </Scripts>
</asp:ScriptManager>
```



## 練習 5.4 : 部署內嵌在資源組件的 AJAX 指令碼

### 目的：

這個練習主要是讓您熟悉將 JavaScript 檔案發行版與除錯版檔案，內嵌在資源組件 (Resource Assembly) 之中，以方便進行部署。

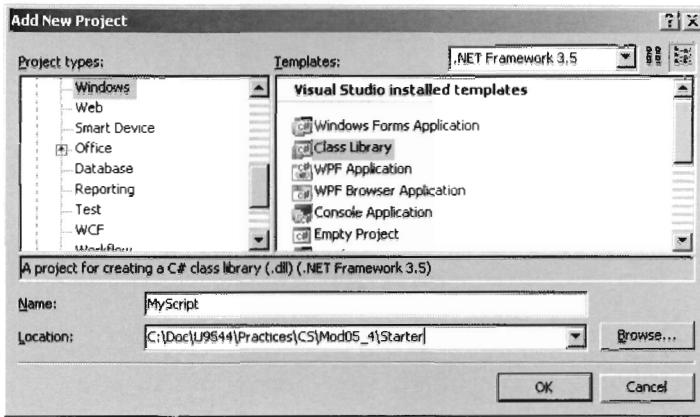
### 功能描述：

這個練習會在網頁中使用以下的步驟將說明如何建立一個內嵌指令碼的組件，以讓 AJAX 網頁，能夠叫用其中的程式，並說明如何利用 ScriptMode 來設定要執行的指令碼版本。

### 預估實作時間：20 分鐘

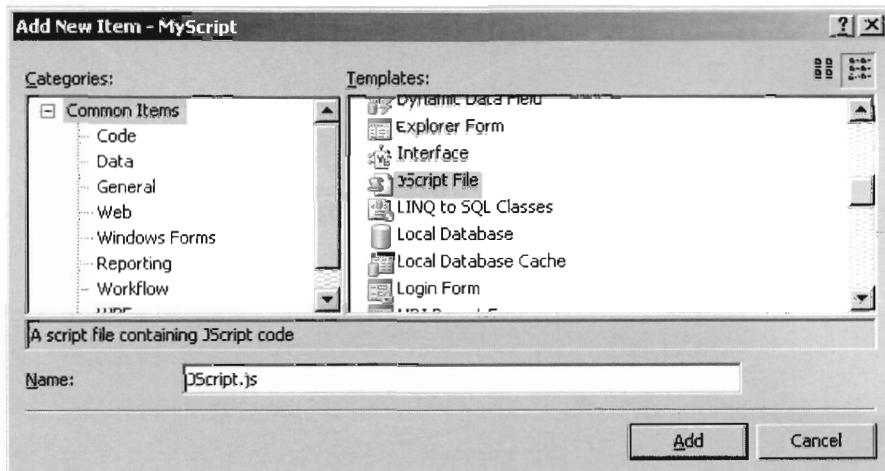
### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 在 Visual Studio 之中，選取「File」→「New Project」，選取 Visual Basic 或 Visual C# 程式語言，建立一個類別庫 (Class Library) 專案，將專案名稱定義為 MyScript。



3. 刪掉預設的 Class1.cs 或 Class1.vb 檔案。

4. 建立一個發行版的 JScript.js 檔：



5. 在 JScript.js 檔，然後加入以下程式碼，建立一個 Add 函式，傳入兩個參數，並把兩個參數的值相加後回傳：

```
JavaScript
function Add(arg1 ,arg2) {
    return arg1 + arg2;
}
```

6. 建立一個 JScript.debug.js 檔，注意除錯版本的檔案命名規則，以.debug.js 結尾。在除錯版的程式中，先呼叫 alert 函式，顯示除錯訊息，並加上除錯判斷，在參數為未定義 ('undefined') 的情況下，觸發 Error.argumentUndefined 例外。若參數為 null 的情況下，觸發 Error.argumentNull 例外：

```
JavaScript
alert('debug..');
function Add(arg1 ,arg2) {
    if ( (typeof(arg1) === 'undefined') || (typeof(arg2) === 'unde
```

```

fined') ) {
    throw Error.argumentUndefined('參數未定義!');
}
if((arg1 === null) || (arg2 === null)) {
    throw Error.argumentNull('參數為 Null');
}
return arg1 + arg2;
}

```

7. 在 Visual Studio 工具屬性視窗，設 JScript.debug.js 與 JScript.js 兩個檔案的「Build Action」屬性設為「Embedded Resource」。
8. 在 MyScript 專案，加入 System.Web.dll 的參考。
9. 在 MyScript 專案中的 AssemblyInfo.cs 或 AssemblyInfo.vb 檔案中，匯入 System.Web.UI 命名空間，並加上以下兩個定義 WebResource 的屬性：

```

Visual Basic
Imports System.Web.UI
<Assembly: WebResource( "MyScript.JScript.js", "text/javascript")>
<Assembly: WebResource( "MyScript.JScript.debug.js", "text/java
script")>

```

```

C#
using System.Web.UI;
[assembly: WebResource("MyScript.JScript.js", "text/javascript")]
[assembly: WebResource("MyScript.JScript.debug.js", "text/javasc
ript")]

```

10. 在目前的方案中，選『File』→『Add』→『New Web Site』加入一個「ASP.NET Web Site」網站專案，名稱為 Mod05\_4。
11. 設定此網站參考 MyScript 專案。在『Solution Explorer』視窗，點選 Mod05\_4 網站中，按滑鼠右鍵，選『Add Reference』，點選『Projects』頁中的 MyScript 專案。
12. 在網站專案中加入一個 Ajax 網頁，使用預設的檔名，找到 ScriptManager 標籤，將它修改如下：

```

<asp:ScriptManager ID="ScriptManager1" runat="server">
```

```
<Scripts>
<asp:ScriptReference Name="MyScript.JScript.js"
Assembly="MyScript" />
</Scripts>
</asp:ScriptManager>
```

利用 ScriptReference 參考到 MyScript 組件中的 MyScript.JScript.js。注意，不管你是要載入除錯版本或是發行版本的 javascript 檔案，Name 的部份一律設定為發行版檔案名稱。稍後您可以利用 ScriptMode 屬性，來設定要載入發行版還是除錯版的檔案。

13. 在網頁最下方建立一段 JavaScript 函式，在 pageLoad 事件之中，呼叫 Add 函式，然後將結果顯示在訊息方塊之中。

```
JavaScript
<script type="text/javascript">
function pageLoad() {
    alert( Add(1,3) );
}
</script>
```

14. 建置專案，從 Visual Studio 「Build」選單，選取「Build Solution」。

15. 測試，執行網頁，執行時會下載發行版(Release 版)的指令碼 (Script)。

修改 ScriptReference 中，ScriptMode 的設定，將 ScriptMode 設為 "Debug"：

```
<asp:ScriptReference Name="MyScript.JScript.js"
Assembly="MyScript" ScriptMode="Debug" />
```

執行時會下載除錯版指令碼，你會先看到除錯的訊息方塊。

故意修改 pageLoad 事件中的 Javascript，呼叫 Add 函數時，不帶任何參數：

```
JavaScript
<script type="text/javascript">
function pageLoad() {
    alert( Add() );
}
</script>
```

則當 ScriptMode 設定為「Debug」的情況下，會有「參數未定義的」除錯訊息。若 ScriptMode 為「Release」時，會顯示 NaN.。

### 指令碼全球化設計

- 以特定的文化，對資料進行格式化或者是剖析時間、貨幣等相關資料
- 啓用指令碼全球化

ScriptManager 的 EnableGlobalization 屬性設定為 True

```
<asp:ScriptManager runat="server" ID="ScriptManager1"
    EnableScriptGlobalization="true" >
</asp:ScriptManager>

<script type="text/javascript">
function pageLoad(sender, args) {
    var culture = Sys.CultureInfo.CurrentCulture.name;
    var d = new Date();
    alert(culture + " :" + d.localeFormat("D"));
}
</script>
```

### 指令碼全球化設計

指令碼全球化設計 (Script globalization) 是指能以特定的文化，對資料進行格式化或者是剖析的動作。例如要表達貨幣的資訊，台灣習慣使用 NT\$；美國使用\$符號。日期對於台灣人而言，可能想看到的是「2007 年 12 月 1 日」；而美國人可能想看到的是「December 1, 2007」。

在 AJAX 網頁中，可以將 ScriptManager 的 EnableGlobalization 屬性設定為 True，如此 ScriptManager 便可以根據特定的文化，自動產生 CultureInfo 物件，來解析及處理這些和時間、貨幣等相關的資料。預設 EnableGlobalization 屬性設定為 false。若要達成指令碼當地化的目標，可以在 ASP.NET AJAX 網頁上，設定 Page 的 Culture 屬性為特訂文化代碼(如 zh-TW 代表台灣)，或者是設定為「auto」自動根據瀏覽器的語言喜好設定切換。

以下為 ScriptManager 啓用指令碼全球化設定範例：

```
<asp:ScriptManager runat="server" ID="ScriptManager1" Enable
    ScriptGlobalization="true" >
</asp:ScriptManager>
```

若要自動根據瀏覽器的語言喜好設定切換全球化設定，可在 AJAX 網頁 Page 宣告中，設定 Culture 為「auto」：

```
<%@ Page Language="VB" Culture="auto" %>
```

以下 JavaScript 範例，在網頁載入時，取得目前的文化特性代碼，並利用 Date 物件的 localeFormat 函式，顯示符合此文化特性的日期資訊。localeFormat 函式傳入格式化字串「D」，代表完整日期：

```
JavaScript
<script type="text/javascript">
function pageLoad(sender, args) {
    var culture = Sys.CultureInfo.CurrentCulture.name;
    var d = new Date();
    alert(culture + " : " + d.localeFormat("D"));
}
</script>
```

### 指令碼當地語系化

- ASP.NET AJAX 的用戶端指令碼支援當地語系化 (Localization)能力
- 需將 ScriptManager 的 EnableScriptGlobalization 設為 true
- 搭配瀏覽器個人語言喜好的設定  
設定 Page 宣告 UICulture Attribute 為 auto

```
<asp:ScriptManager ID="ScriptManager1" runat="server"
EnableScriptLocalization="true" >
<Scripts>
<asp:ScriptReference Assembly="MyCalcScript"
Name="MyCalcScript.JScript.js" />
</Scripts>
</asp:ScriptManager>
```

## 指令碼當地語系化

簡單的說，台灣人看到的網頁，會期待它是使用繁體中文顯示文字；而美國人看到的網頁，就會希望他是使用英文來表達。本文介紹如何使用 ASP.NET AJAX 來設計符合當地語系與全球化的網頁程式。

JavaScript 缺當地語系化(Localization)能力，但 ASP.NET AJAX 的用戶端指令碼支援當地語系化(Localization)能力，您可將 Script 檔與相關資源內嵌在組件之中(Assembly)，以設計支援當地語系化的網頁程式。

當 ScriptManager 的 EnableScriptGlobalization 設為 true，ScriptManager 會建立一個 CultureInfo 物件，以進行全球化。你可以把網頁 Page 宣告 Culture 設定為 Auto，自動根據瀏覽器的設定顯示時間、貨幣。若要設計當地語系化的 AJAX 程式，則可設定 Page 宣告 UICulture 設定為 Auto，然後再搭配各語系專用的資源。

### 練習 5.5 : 指令碼當地語系化

這個練習主要是如何設計能支援當地語系化的指令碼，以便在ASP.NET網頁中使用他們。

•預估實作時間：20分鐘

### 練習 5.5 : 指令碼當地語系化

#### 目的：

這個練習主要是如何設計能支援當地語系化的指令碼，以便在ASP.NET 網頁中使用他們。

#### 功能描述：

這個練習會說明如何在網頁中使用到內嵌資源的組件，若JavaScript 執行計算時發生錯誤，將從資源組件中，根據使用者IE 的語言喜好設定，依情況，自動根據 IE 瀏覽器語言喜好的設定最上方的項目，來讀取英文或中文錯誤訊息做顯示。

預估實作時間：20 分鐘

#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 建立當地語系化資源類別庫。在 Visual Studio 之中，在 Visual Studio 之中，選取「File」→「New Project」，選取 Visual

Basic 或 Visual C# 程式語言，建立一個類別庫 (Class Library) 專案，將專案名稱定義為 MyCalcScript。

3. 刪掉專案中，預設的 Class1.cs 或 Class1.vb 檔案。
4. 在 MyCalcScript 專案，自主選單「Web Site」下「Add New Item...」，選取「Resources File」，命名為 MessageResources.resx 檔案。
5. 在資源檔的編輯畫面中，新增一個字串的資源，分別設定「Name」與「Value」為「CalcError」與「Calculation Error!」。
6. 重複步驟 3，在專案加一個 MessageResources.zh-TW.resx 檔案，注意檔名的命名規則，以「.zh-TW.resx」結尾。接著在資源檔的編輯畫面中，新增一個字串的資源，分別設定「Name」與「Value」為「CalcError」與「計算發生錯誤！」。
7. 自主選單「Web Site」下「Add New Item...」，選取「Jscript File」加一個 JavaScript 檔案，名為 JScript.js，加入以下內容，若計算發生錯誤，則讀取出對應在資源檔案中的錯誤訊息後，利用訊息方塊顯示出來：

```
JavaScript
function Add(arg1, arg2) {
    if ((typeof (arg1) === 'undefined') || (typeof (arg2) === 'undefined')) {
        var errMsg = Messages.CalcError;
        alert(errMsg);
        return null;
    }
    return arg1 + arg2;
}
```

8. 在 Visual Studio 工具屬性視窗，分別設定 JScript.js 檔案與 MessageResources.resx、MessageResources.zh-TW.resx 檔案的「Build Action」為「Embedded Resource」。
9. 在 MyCalcScript 專案中，加入 System.Web.dll 與 System.Web.Extensions.dll 參考。

10. 在 MyCalcScript 專案中的 AssemblyInfo.cs 或 AssemblyInfo.vb 檔案中，匯入 System.Web.UI 命名空間，並加上以下兩個定義 WebResource 的屬性：

```
Visual Basic
Imports System.Web.UI
<Assembly: WebResource("MyCalcScript.JScript.js", "text/javascript")>
<Assembly: ScriptResource("MyCalcScript.JScript.js", "MyCalcScript.MessageResources", "Messages")>
```

```
C#
using System.Web.UI;
[assembly: WebResource("MyCalcScript.JScript.js", "text/javascript")]
[assembly: ScriptResource("MyCalcScript.JScript.js",
    "MyCalcScript.MessageResources",
    "Messages")]
```

在 AssemblyInfo.cs 或 AssemblyInfo.vb 檔案中，定義一個 Messages 類別，使用 ScriptResource Attribute 搭配 WebResource Attribute 指示 ScriptManager 來建立此類別。WebResource Attribute 用來指明資源的名稱 (MyCalcScript.JScript.js) 和類型 (text/javascript)。ScriptResource Attribute 指明指令碼檔名 (MyCalcScript.JScript.js) 資源檔名稱 (MyCalcScript.MessageResources)，以及要在 JavaScript 中使用的類別名稱(Messages)。

11. 建置專案，修正可能的錯誤。
12. 在目前的方案中，選「File」→「Add」→「New Web Site」加入一個「ASP.NET Web Site」網站專案，名稱為，命為 Mod05\_5。
13. 設定此網站參考 MyCalcScript 專案。在「Solution Explorer」視窗，點選 Mod05\_4 網站中，按滑鼠右鍵，選「Add Reference」，點選「Projects」頁中的 MyCalcScript 專案。
14. 在 Mod05\_5 專案中，加入一個 AJAX 網頁，找到 ScriptManager 標籤，將它修改如下：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
```

```

EnableScriptLocalization="true" >
<Scripts>
    <asp:ScriptReference Assembly="MyCalcScript"
        Name="MyCalcScript.JScript.js" />
</Scripts>
</asp:ScriptManager>

```

設定 EnableScriptLocalization 為「true」啓用當地語系化功能。並設定 ScriptReference，參考到 MyCalcScript 組件中的 MyCalcScript. JScript.js 檔案。

15. 在網頁最下方建立一段 JavaScript 函式，在 pageLoad 事件之中，呼叫 Add 函式，故意不傳參數值，然後將結果顯示在訊息方塊之中。

```

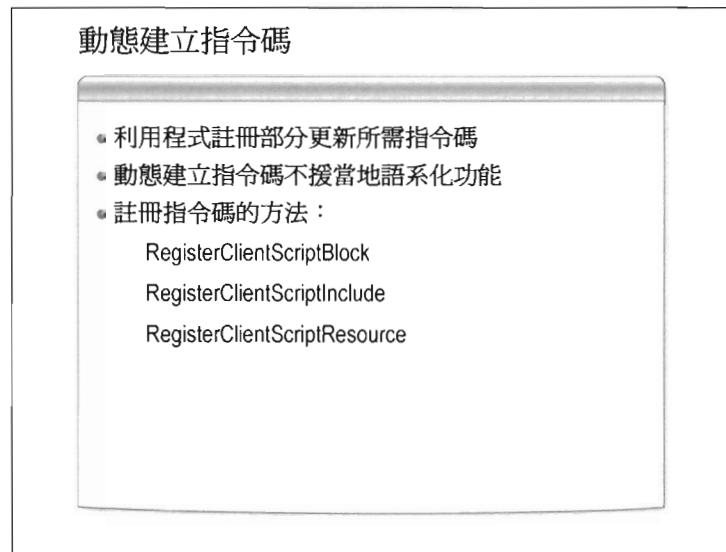
JavaScript
<script type="text/javascript">
function pageLoad() {
    alert(Add());
}
</script>

```

16. 在網頁的 Page 宣告，加上 UICulture=" auto" 設定：

```
<%@ Page Language="CS" UICulture="auto" %>
```

執行網頁測試，按使用者的語言喜好設定，依情況，自動根據 IE 瀏覽器語言喜好的設定最上方的項目，來顯示英文或中文的錯誤訊息。



## 動態建立指令碼

您可以使用 ScriptManager 控制項為 UpdatePanel 註冊指令碼，以進行部分更新作業。但請注意，動態建立指令碼不援當地語系化功能。

註冊指令碼的方法包含：

- RegisterClientScriptBlock 方法：將指令碼先組成字串，再行註冊。指令碼產生在<form>開始標籤之後。
- RegisterClientScriptInclude 方法：將指令碼先儲存在外部檔案，指令碼產生在<form>開始標籤之後。
- RegisterClientScriptResource：指令碼檔案內嵌在組件(Assembly)中，指令碼產生在<form>開始標籤之後，

### 動態建立指令碼範例

```
Visual Basic
Dim script As String = "function SayHi() { alert('hi');}"
ScriptManager.RegisterClientScriptBlock(
    Me,
    GetType(Page),
    "JScript",
    script,
    True)
```

```
C#
string script = @" function SayHi() { alert('hi');}";
ScriptManager.RegisterClientScriptBlock(
    this,
    typeof(Page),
    "JScript",
    script,
    true);
```

### 動態建立指令碼範例

以下是使用 RegisterClientScriptBlock 方法動態建立指令碼範例：

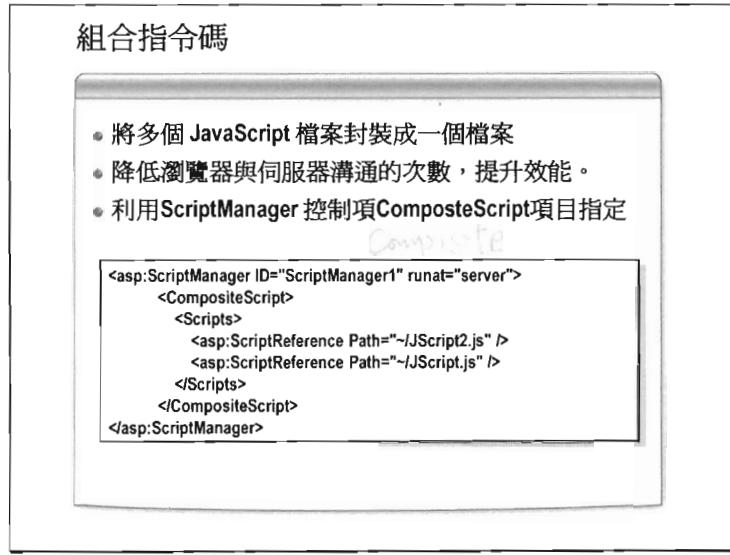
```
Visual Basic
Dim script As String = "function SayHi() { alert('hi');}"
ScriptManager.RegisterClientScriptBlock(
    Me,
    GetType(Page),
    "JScript",
    script,
    True)
```

```
C#
string script = @" function SayHi() { alert('hi');}";
ScriptManager.RegisterClientScriptBlock(
    this,
    typeof(Page),
    "JScript",
    script,
    true);
```

動態建立指令碼之後，在網頁UpdatePanel中的控制項，便可以呼叫此SayHi函式：

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
```

```
<ContentTemplate>
    <asp:Button ID="Button1" runat="server"
        OnClientClick="SayHi();return false;" Text="Button" />
    <asp:Label ID="Label1" runat="server" Text="Label">
    </asp:Label>
</ContentTemplate>
</asp:UpdatePanel>
```



## 組合指令碼

若網站程式使用到許多 JavaScript 檔案，下載單一檔案的速度很快，不會有太大影響；若要下載多個檔案，可能會影響到執行效能。您可以將多個 JavaScript 檔案封裝成一個檔案，將指令碼組合在一起(稱為 Composite Script)，以便降低瀏覽器與伺服器溝通的次數，提升效能。

您可以利用 ScriptManager 控制項或 ScriptManagerProxy 控制項中使用 ComposeScript 項目將多個 JavaScript 檔案結合在一起。若 Script 有相依性問題，就必需由程式設計師適當地調整 ComposeScript 項目中 ScriptReference 項目的順序。

有些瀏覽器支援 Script 壓縮功能，ComposeScript 中的 Script 檔自動經過壓縮，再傳送給瀏覽器，但 Microsoft Internet Explorer 6.0 除外。

在 ScriptManager 控制項設定組合指令碼範例如下，利用 ScriptReference 的 Path Attribute 來參考到 JScript2.js 與 JScript.js 檔案。

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
    <CompositeScript>
        <Scripts>
            <asp:ScriptReference Path("~/JScript2.js" />
            <asp:ScriptReference Path "~/JScript.js" />
        </Scripts>
    </CompositeScript>
</asp:ScriptManager>
```

## 總結

- ASP.NET AJAX架構
- Microsoft Ajax 程式庫提供JavaScript更多擴充功能
- Microsoft Ajax 程式庫提供用戶端 JavaScript指令碼的偵錯與追蹤功能。
- 使用發行版與除錯版JavaScript指令碼
- Microsoft Ajax 程式庫支援指令碼當地語系化

# 第六章: 網站會員機制規 劃

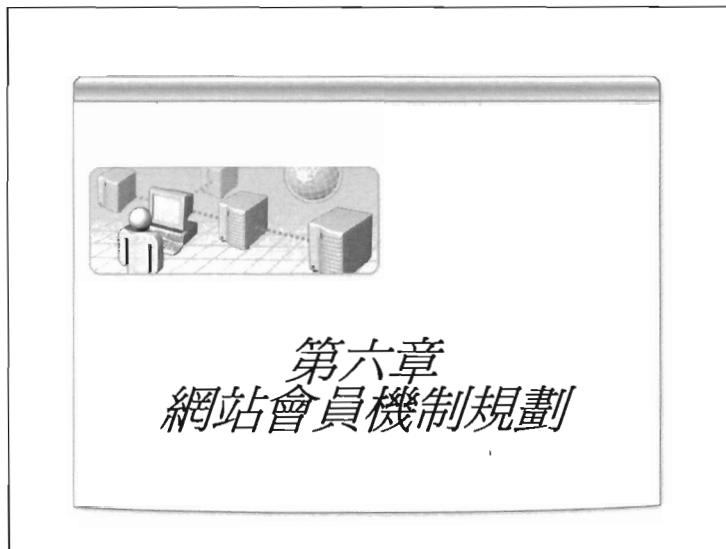
## 本章大綱

ASP.NET 網站會員機制.....	4
認識會員管理服務 .....	5
Membership Provider .....	6
Membership API 之 Membership.....	7
Membership API 之 Membership.....	9
練習 6.1 : 使用 Membership API 建立使用者 .....	10
使用 AJAX 整合 Membership .....	14
練習 6.2 : Ajax Membership API 的整合運用 .....	15
認識角色管理系統 .....	20
Role Provider .....	21
Role API .....	23
練習 6.3 : 使用 Role API 管理角色 .....	24
認識使用者設定檔 .....	29
Profile Provider .....	30
建立 Profile 屬性 .....	31
練習 6.4 : 存取 Profile 的個人資料 .....	33
Profile 整合 Theme .....	37
練習 6.5 : 使用 Profile 搭配 Theme 佈景主題 .....	38



作者：

趙敏翔



---

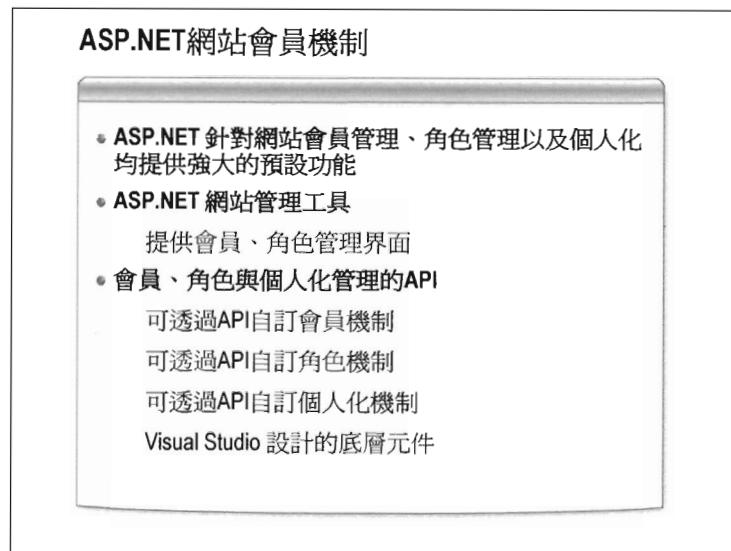
## 大綱

- 認識會員管理服務
- Membership API 介紹
- 認識角色管理服務
- Roles API 介紹
- 何謂使用者設定檔
- Profile 整合 Theme
- 總結

本章將介紹 ASP.NET 的網站會員、角色管理服務以及個人化的網站設定機制，用以快速建立網站的會員登入驗證功能與管理。此外，也將進一步探討使用會員機制搭配 AJAX 的進階設計方式。

在這一章中將學習到

- 認識會員管理服務
- Membership API 介紹
- 認識角色管理服務
- Roles API 介紹
- 何謂使用者設定檔
- Profile 整合 Theme

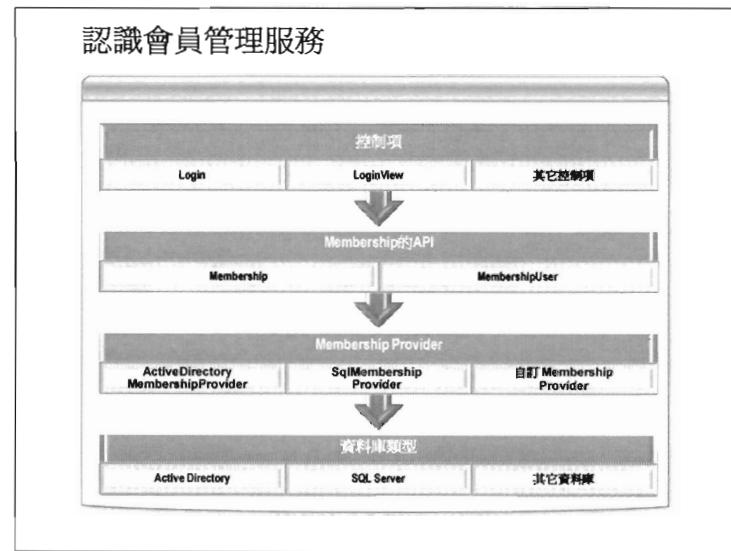


## ASP.NET 網站會員機制

Visual Studio 2008 在網頁的設計上提供一個管理工具它的名字是--ASP.NET 網站管理工具(ASP.NET Web Site Administration Tool)，這個管理工具提供網站 Web.config 的設定並可以建立網站的會員資料及角色管理。

網站的網站管理者或社群版主通常會要求一些特別的線上會員、角色管理網頁，此時網站開發者就得自行開發這個網頁。幸好 ASP.NET 提供這組 API 讓開發人員使用，開發人員就可以省去開發中間層元件的功夫，並且輕易的建立會員、角色管理網頁。

使用者設定檔(User Profile)是一組記錄使用者特性的物件，使用者資訊在 MembershipUser 物件中只提供幾個的欄位，當然對於多變化的網站需求，內建的欄位是不足夠的，User Profile 具有可擴充能力，網站開發者可以依據企業需求自行擴充欄位。



## 認識會員管理服務

ASP.NET 會員管理服務是一組多層式架構的元件，後端可支援各類型資料庫(但必須撰寫自訂 Membership Provider)，中間層包含 Membership API 與 Membership Provider，前端支援各類伺服器控制項。

- **控制項：**支援 Membership API 的控制項包含：Login、LoginStatus、LoginView，CreateUserWizard 等。
- **Membership Provider 與資料庫類型：**屬於資料邏輯層負責處理如何存取後端資料庫，ASP.NET 內建 ActiveDirectoryMembershipProvider 可以連接到 Active Directory，及 SqlMembershipProvider 可以連接到 SQL Server 資料庫。Oracle 及其他類型的資料庫，則必須自行撰寫 Membership Provider。
- **Membership API：**Membership API 主要有 Membership 類別、及 MembershipUser 類別。



## Membership Provider

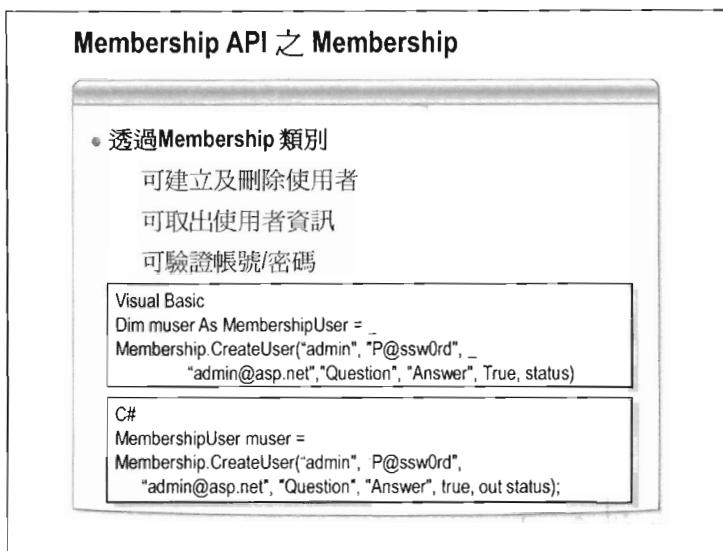
Membership Provider 是 ASP.NET 會員管理的資料提供者，包含連接資料來源及存取會員資訊給上層的 Membership 類別與 MembershipUser 類別，主要的預設設定是儲存在 Machine.config 中。

```
<membership>
  <providers>
    <add name="AspNetSqlMembershipProvider"
         type="System.Web.Security.SqlMembershipProvider"
         connectionStringName="LocalSqlServer" ... ... />
  </providers>
</membership>
```

### 建立 SqlMembershipProvider 的 SQL Server 資料庫

使用 SqlMembershipProvider 可以將資料存放在資料庫中，當然此資料庫的預設結構必須符合 Membership 儲存的方式，因此可以使用 SDK 中所提供的 Aspnet\_regsql.exe 來建置資料庫。另外在建置資料庫時必須加上參數 -Am，代表要建立的資料庫類型是給 SqlMembershipProvider 用的。

aspnet\_regsql.exe -Am



## Membership API 之 Membership

在 ASP.NET 應用程式 Membership 類別是負責驗證使用者憑證及管理使用者基本資料的類別，像是帳號、密碼及 E-mail。

Membership 類別提供的特性包含：

- 建立新的使用者。
- 儲存會員資訊(使用者帳號、密碼、Email 等資訊)在 Microsoft SQL Server 或其他資料來源。
- 驗證瀏覽網站的使用者身份。可以呼叫它的功能以驗證使用者身份，或是使用 Login 控制項建立一個完整的驗證系統。
- 管理密碼，包含建立、變更、查詢、重設密碼等等。

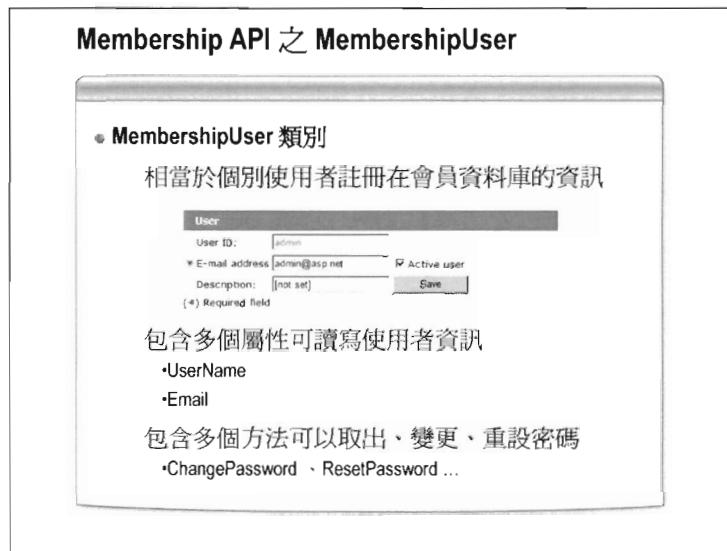
Membership 類別可以獨立使用或者與 FormsAuthentication 搭配使用以建立一個網站驗證使用者的機制。Login 控制項便是封裝 Membership 類別以提供驗證使用者方便的功能。另外，亦可透過常用的靜態方法，如使用 CreateUser 方法來建立使用者帳號，程式碼如下：

Visual Basic

```
Dim muser As MembershipUser  
Membership.CreateUser("admin", "P@ssw0rd", _  
    "admin@asp.net", "Question", "Answer", True, status)
```

C#

```
MembershipUser muser =  
    Membership.CreateUser("admin", "P@ssw0rd",  
        "admin@asp.net", "Question", "Answer", true, out status);
```



## Membership API 之 Membership

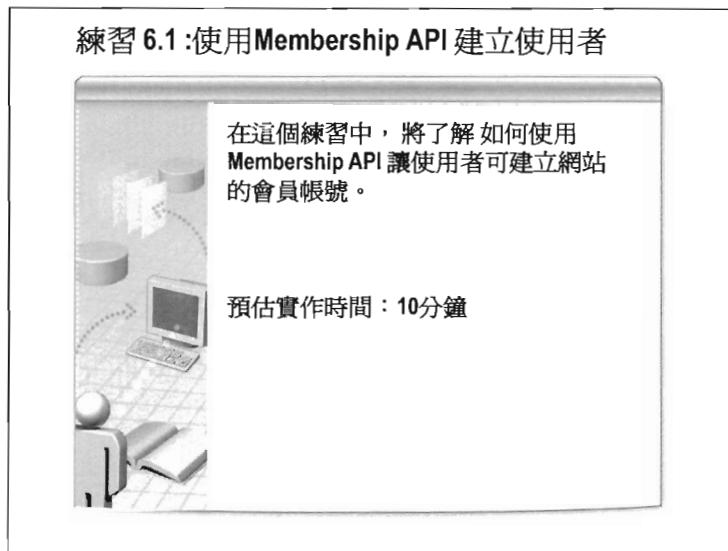
MembershipUser 物件代表一個存在會員資料庫的會員資訊。它展現關於使用者的名字、電子郵件的屬性與會員密碼變更或重設等方法，例如：可以使用 MembershipUser 物件將會員的資料取出到網頁中使用。

User

User ID: admin  
 \* E-mail address: admin@asp.net  Active user  
 Description: [not set]   
 (\*) Required field

MembershipUser 物件有包含多個方法可以取出、變更、重設密碼等，如下列所示：

- ChangePassword：變更使用者存在資料來源的密碼。
- ResetPassword：密碼重設，隨機產生新密碼。



## 練習 6.1：使用 Membership API 建立使用者

### 目的：

在這個練習中，將了解如何使用 Membership API 讓使用者可建立網站的會員帳號。

### 功能描述：

在這個練習中，將使用 Membership 建立使用者帳號使用 CreateUser 方法及 Login 控制項驗證新增帳號是否可以登入。

### 預估實作時間：10 分鐘

### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod06\_1\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod06\_1」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 從工具箱拖拉一個 Button 控制項到畫面中，設定 Button 的 Text 屬性為「建立使用者」。
5. 雙擊 Button 控制項，在 Button 的 Click 事件程序中，加入建立使用者的程式碼，使用 CreateUser 方法建立使用者名稱 Admin 以及其他必要資訊，並判斷建立狀態是否成功：

```

Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim status As MembershipCreateStatus
    Dim muser As MembershipUser
    muser = Membership.CreateUser("Admin", "P@ssw0rd", _
        "Admin@asp.net", "Question", _
        "Answer", True, status)

    Select Case status
        Case MembershipCreateStatus.Success
            Response.Write(muser.UserName + "建立成功.")
        Case MembershipCreateStatus.DuplicateUserName
            Response.Write("帳號名稱重複")
        Case Else
            Response.Write("帳號建立失敗")
    End Select
End Sub

```

```

C#
protected void Button1_Click(object sender, EventArgs e)
{
    MembershipCreateStatus status;
    MembershipUser muser;
    muser = Membership.CreateUser("Admin", "P@ssw0rd",
        "Admin@asp.net", "Question",
        "Answer", true, out status);
    switch (status)
    {
        case MembershipCreateStatus.Success:
            Response.Write(muser.UserName + "建立成功.");
            break;
        case MembershipCreateStatus.DuplicateUserName:
            Response.Write("帳號名稱重複");
            break;
        default:
            Response.Write("帳號建立失敗");
            break;
    }
}

```

```
}
```

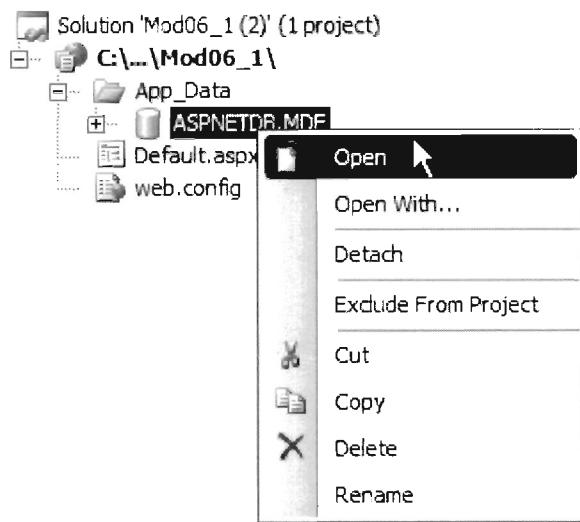
6. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。按下「建立使用者」按鈕建立使用者。



7. 接著再重新按下「建立使用者」按鈕，這時候會出現帳號名稱重複的警示訊息。



8. 點選 Solution Explorer 中的專案，執行重新整理，在專案的 App\_Data 資料夾下將會出現 ASPNETDB.MDF 資料庫。
9. 開啓系統所自動建立的 ASPNETDB.MDF 資料庫。



10. 檢視 aspnet\_Membership 和 aspnet\_Users 資料表，可看到建立的資料。

The screenshot shows the Server Explorer window with the 'aspnet\_Memb...SPNETDB.MDF' connection selected. In the 'Tables' section, the 'aspnet\_Users' table is highlighted. To the right, a detailed view of the 'aspnet\_Users' table is shown in a grid format. The columns are ApplicationId, UserId, and Password. There is one record displayed:

	ApplicationId	UserId	Password
*	36c5-19215c7c4971 8a135e61-485b-...	G1t6mcK2jJqVaL...	



## 使用 AJAX 整合 Membership

以 AJAX 提供整合 ASP.NET 的 Forms 驗證，主要由兩個元件構成，分別為用戶端的 Sys.Services.AuthenticationService 物件以及伺服器端的 AuthenticationWebService，使用 AJAX 整合 ASP.NET 的 Membership 會員機制可以讓使用者無需導向特定的登入頁面，即可在用戶端完成身分驗證功能。

用戶端元件 Sys.Services.AuthenticationService 有提供兩個最主要的方法，分別為：

- login：用來驗證使用者的身份
- logout：用來清除使用者的憑證

伺服器端的元件為 AuthenticationWebService 類別，此類別主要是對應到 Authentication\_JSON\_AppService.axd。另外要特別注意的是，如果要啓用 ASP.NET AJAX 驗證機制，那麼就必須在 web.config 中設定：

```
<authenticationService enabled="true" />
```

### 練習 6.2 : Ajax Membership API 的整合運用

在這個練習中，將了解如何使用 Ajax Membership API，透過 Ajax Membership 機制提供使用者更高的便利性。

預估實作時間：20分鐘

### 練習 6.2 : Ajax Membership API 的整合運用

#### 目的：

在這個練習中，將了解如何使用 Ajax Membership API，透過 Ajax Membership 機制提供使用者更高的便利性。

#### 功能描述：

在這個練習中，使用 Ajax Membership API 的會員服務機制設計，搭配 Forms 驗證機制，讓使用者可以不用切換頁面，就可以更快速的通過網站的驗證。

預估實作時間：20 分鐘

#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod06\_2\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod06\_2」。

3. 自主選單「Web Site」下「ASP.NET Configuration」，點選「Security」→「Select authentication type」→點選「From the internet」，並按下「Done」將驗證方式改為 Forms 驗證。
4. 回到「Security」功能頁面，點選「Create user」，建立一個網站會員，例如建立一個名為 admin 且密碼為 P@ssw0rd 的使用者。

**Create User**

Sign Up for Your New Account

User Name:

Password:

Confirm Password:

E-mail:

Security Question:

Security Answer:

Active User

5. 修改 web.config，設定授權拒絕匿名存取者。

```
<authorization>
  <deny users="?" />
</authorization>
<authentication mode="Forms" />
```

6. 設定 web.config 中 webServices 的 authenticationService 項目啓用 authenticationService 機制。

```
<system.web.extensions>
  <scripting>
    <webServices>
      <authenticationService enabled="true" requireSSL="false"/>
    </webServices>
  </scripting>
</system.web.extensions>
```

7. 自主選單「Web Site」下「Add New Item...」，選取「AJAX Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，檔名命名為 login.aspx。
8. 從工具箱中拖拉四個 HTML 控制項，兩個 Text, 與兩個 Button 控制項到網頁中，設定如下：

```
<div>Username: <input type="text" name="username" /></div>
<div>Password: <input type="password" name="password" /></div>
<div>
    <input type="button" id="loginButton" value="Login"/>
    <input type="button" id="logoutButton" value="Logout" />
</div>
```

9. 在 login 網頁中加入 ASP.NET AJAX 驗證機制用來驗證的用戶端 JavaScript 程式碼(此範例程式碼亦可直接由「\U9544\Practices\VB 或 CS\Mod06\_2\Starter」目錄的 code.txt 中取得)：

```
<script type="text/javascript">
function pageLoad() {
    document.getElementById('loginButton').attachEvent('onclick', login);
    $addHandler($get('logoutButton'), 'click', logout);
}
function login() {
    var u = $get('username').value;
    var p = $get('password').value;
    var isPersistent = false;
    var customInfo = null;
    var redirectUrl = null;
    Sys.Services.AuthenticationService.login(u, p, isPersistent, customInfo, redirectUrl, loginCompleted, loginFailed);
}
function loginCompleted(result, context, methodName) {
    if(result) {
        alert("Successfully logged in.");
    }
    else {
        alert("Failed to login.");
    }
}
function loginFailed(error, context, methodName) {
    alert(error.get_message());
}
function logout() {
    Sys.Services.AuthenticationService.logout('Default.aspx');
}
</script>
```

10. 接著在專案中，加一個 sec 資料夾，並在此資料夾下加入一個新網頁 sec.aspx

11. 回到 login.aspx 加入一個 HTML 的超連結，點下去之後會叫用一個 JavaScript 函式 checkFirst，必須是驗證過後才能執行。

```
<a href="sec\sec.aspx" onclick="return checkFirst()"><b>看秘密網頁</b> (要先登入)</a><br />
```



12. 在 login.aspx 中加入 JavaScript 的 checkFirst 函式設計，程式碼如下：

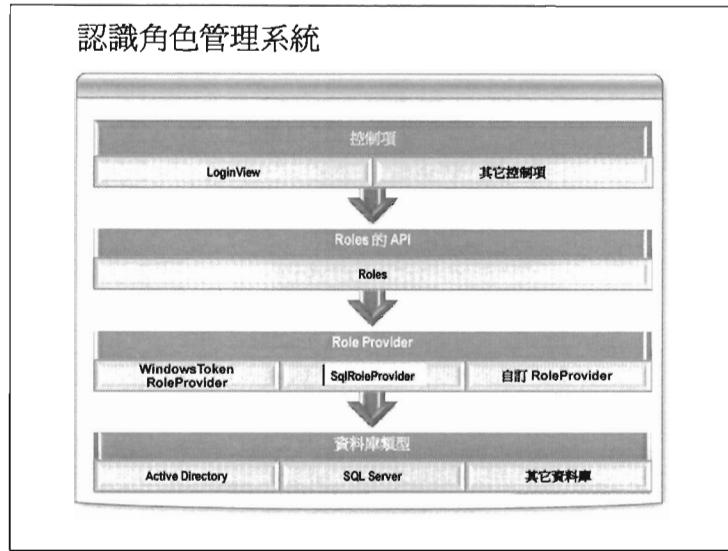
```
function checkFirst()
{
    var loggedIn = Sys.Services.AuthenticationService.get_isLoggedIn();
    if (!loggedIn)
    {
        alert("請先login");
        return false;
    }
    return true;
}
```

13. 執行網頁測試。在「Solution Explorer」→點選 login.aspx 網頁 →按滑鼠右鍵→選「View In Browser」。按下「看秘密網頁」超連結。



14. 輸入帳號 admin 和密碼 P@ssword 後，再次點選「看秘密網頁」超連結。

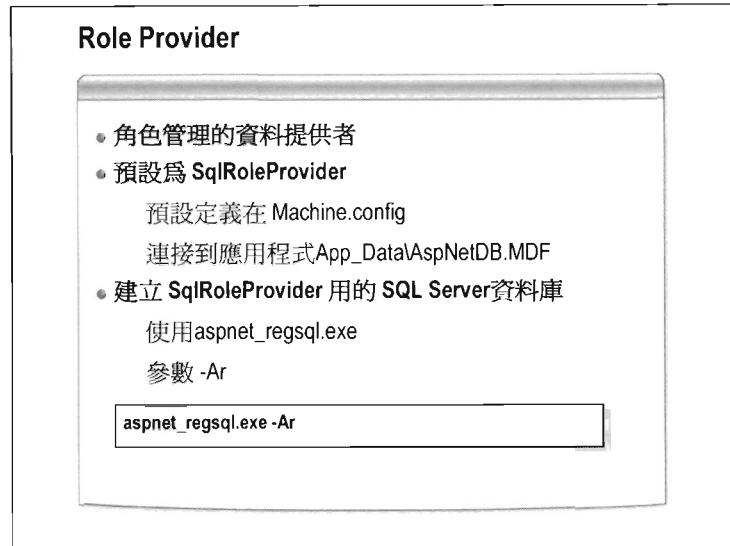




## 認識角色管理系統

角色管理是將會員依資源享用權利分類，以便進行角色安全的授權運作。ASP.NET 的角色管理與會員管理架構相似，後端資料來源為 Active Directory 或 SQL Server，並使用 WindowsTokenRoleProvider、SqlRoleProvider 進行資料存取。

- 控制項：支援 Roles API 的控制項主要是 LoginView。
- Role Provider 與資料庫類型：屬於資料邏輯層負責處理如何存取後端資料庫，ASP.NET 內建 WindowsTokenRoleProvider 可以存取 Windows 系統的使用者群組資料，及 SqlRoleProvider 可以連接到 SQL Server 資料庫。Oracle 及其他類型的資料庫，則必須自行撰寫 Role Provider。
- Roles API：Roles API 主要有 Roles 類別。。



## Role Provider

Role Provider 是 ASP.NET 角色管理的資料提供者。在 Machine.config 設定了預設的 Provider 名稱為 AspNetSqlRoleProvider，類別是 SqlRoleProvider，它是連接到應用程式 App\_Data\AspNetDB.MDF 資料庫。

```
<roleManager>
<providers>
<add
    connectionStringName="LocalSqlServer"
    applicationName="/"
    name="AspNetSqlRoleProvider"
    type="System.Web.Security.SqlRoleProvider" />
<add
    applicationName="/"
    name="AspNetWindowsTokenRoleProvider"
    type="System.Web.Security.WindowsTokenRoleProvider"/>
</providers>
</roleManager>
```

## 建立 SqlRoleProvider 的 SQL Server 資料庫

使用 SqlRoleProvider 可以將資料存放在資料庫中，當然此資料庫的預設結構必須符合 Roles 儲存的方式，因此可以使用 SDK 中所提供之

的 Aspnet\_Regsql.exe 來建置資料庫。另外在建置資料庫時必須加上參數 -Ar，代表要建立的資料庫類型是給 SqlRoleProvider 用的。

```
aspnet_Regsql.exe -Ar
```

## Role API

- 透過 Roles 類別提供的靜態方法

可建立及刪除角色

可加入某使用者到某角色

```
Visual Basic
If Not Roles.RoleExists("Admins") Then
    Roles.CreateRole("Admins")
End If
```

```
C#
if ( ! Roles.RoleExists("Admins"))
{
    Roles.CreateRole("Admins");
}
```

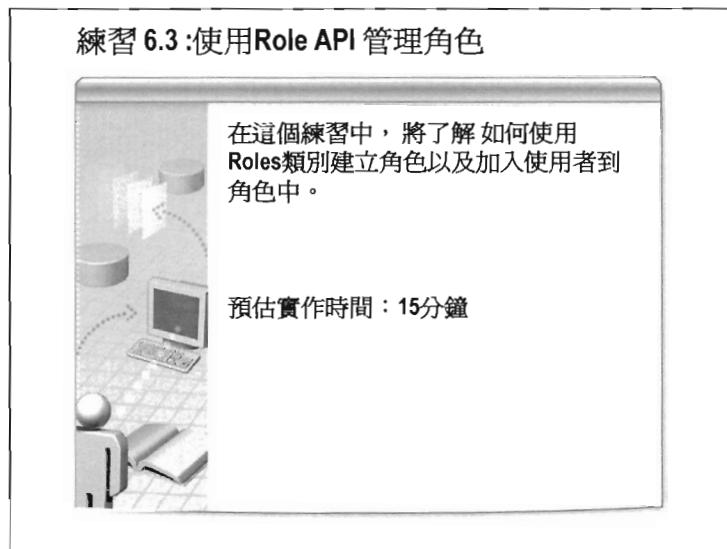
## Role API

Role 角色機制的最主要類別為 Roles 類別，Roles 類別提供一些靜態方法讓開發人員建立或刪除角色，以及會員與角色關係的維護。同時也包含一些唯讀的靜態屬性，以取得 Provider 設定值，例如查詢網站目前使用的角色管理 Provider，或是否啓用快取角色在 Cookie 的功能等。

Roles 類別提供許多角色管理的方法，例如假設想要將某個使用者加入到 Admins 角色，那麼可以叫用 CreateRole 方法來設計，以下範例是使用 RoleExists 方法確定 Admins 角色是否存在，如果不存在使用 CreateRole 方法建立 Admins 角色。

```
Visual Basic
If Not Roles.RoleExists("Admins") Then
    Roles.CreateRole("Admins ")
End If
```

```
C#
if ( ! Roles.RoleExists("Admins " ) )
{
    Roles.CreateRole("Admins ");
}
```



## 練習 6.3 : 使用 Role API 管理角色

目的：

在這個練習中，將了解如何使用 Roles 類別建立角色以及加入使用者到角色中。

功能描述：

在這個練習中，使用 Roles 類別的 CreateRole 方法建立角色，並且使用 AddUserToRole 方法將使用者帳號加到新建立的角色中。

預估實作時間：15 分鐘

實作步驟：

1. 請延續練習專案 Mod06\_1，或是開啓「\U9544\Practices\VB 或 CS\Mod06\_3\Starter\Mod06\_3」網站。
2. 從「File」→「Open Web Site」→選取「ASP.NET Web Site」→選擇「File System」設定 Folder 選取「\U9544\Practices\VB 或 CS\Mod06\_3\Starter\Mod06\_3」目錄，點選「Open」。

3. 打開 web.config 在<system.web>區段中加入啓動角色管理的設定。

```
<roleManager enabled="true" />
```

4. 從工具箱拖拉一個 TextBox 和 Button 控制項到網頁設計畫面中，設定 Button 的屬性為「建立角色」。

5. 在 Button2\_Click 事件中撰寫建立角色的程式碼：

Visual Basic

```
Protected Sub Button2_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    If Roles.RoleExists(TextBox1.Text) = False Then
        Roles.CreateRole(TextBox1.Text)
        Response.Write("角色:" + TextBox1.Text + " 建立成功")
    Else
        Response.Write("角色:" + TextBox1.Text + " 已存在")
    End If
End Sub
```

C#

```
protected void Button2_Click(object sender, EventArgs e)
{
    if (!Roles.RoleExists(TextBox1.Text))
    {
        Roles.CreateRole(TextBox1.Text);
        Response.Write("角色:" + TextBox1.Text + " 建立成功");

    }
    else
    {
        Response.Write("角色:" + TextBox1.Text + " 已存在");
    }
}
```

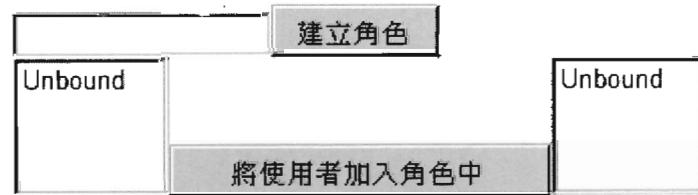
6. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。按下「建立角色」按鈕建立角色。



7. 接著再重新按下「建立角色」按鈕，這時候會出現角色已存在的警示訊息。



8. 按下「建立角色」按鈕建立另一個角色「Users」。
9. 接著，從工具箱拖拉兩個 ListBox 和一個 Button 按鈕，設定 Button 的屬性為「將使用者加入角色中」。



10. 在網頁的 Page\_Load 事件中，加入列出目前網站所有的使用者和角色的程式碼：

```
Visual Basic
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    If Not IsPostBack Then
```

```

For Each muser As MembershipUser In Membership.GetAllUsers()
    ListBox1.Items.Add(muser.UserName)
Next

For Each role As String In Roles.GetAllRoles()
    ListBox2.Items.Add(role)
Next
End If
End Sub

```

```

C#
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        foreach (MembershipUser muser in Membership.GetAllUsers())
        {
            ListBox1.Items.Add(muser.UserName);
        }
        foreach (string role in Roles.GetAllRoles())
        {
            ListBox2.Items.Add(role);
        }
    }
}

```

11. 雙擊「將使用者加入角色中」按鈕，在 Button3 的 Click 事件中，加入程式碼，將所選取的使用者加入指定的角色，程式碼為：

```

Visual Basic
Protected Sub Button3_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    If Not Roles.IsUserInRole(ListBox1.SelectedItem.Text, ListBox2.SelectedItem.Text) Then
        Roles.AddUserToRole(ListBox1.SelectedItem.Text, ListBox2.SelectedItem.Text)
        Response.Write("已成功將使用者加入角色")
    Else
        Response.Write("此使用者已存在指定角色中")
    End If
End Sub

```

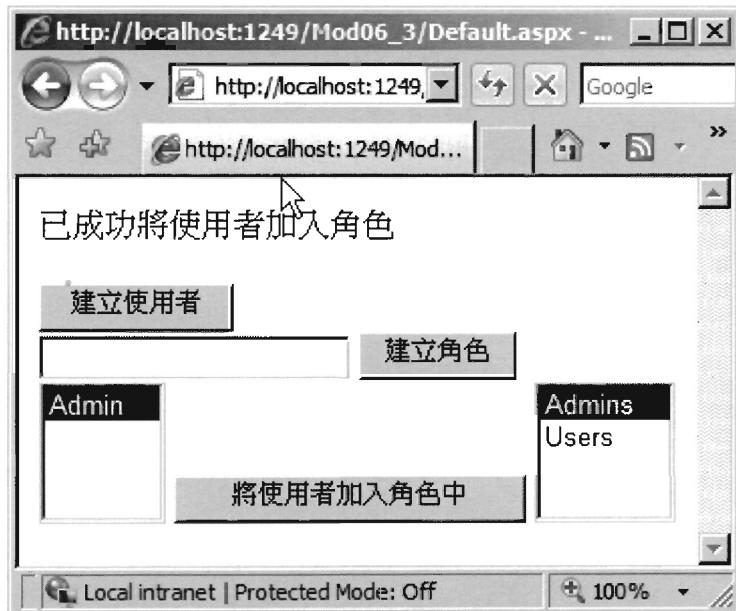
```

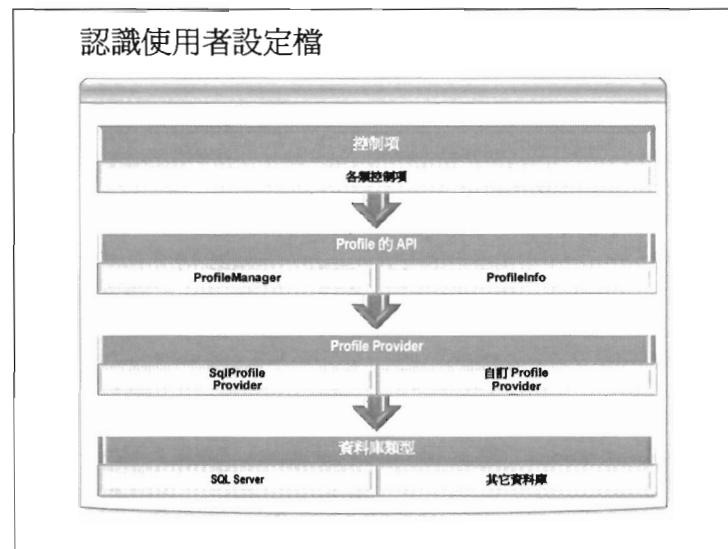
C#
protected void Button3_Click(object sender, EventArgs e)
{
}

```

```
if (!Roles.IsUserInRole(ListBox1.SelectedItem.Text, ListBox2.SelectedItem.Text))
{
    Roles.AddUserToRole(ListBox1.SelectedItem.Text, ListBox2.SelectedItem.Text);
    Response.Write("已成功將使用者加入角色");
}
else
{
    Response.Write("此使用者已存在指定角色中");
}
```

12. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。按下「將使用者加入角色中」按鈕測試執行結果。



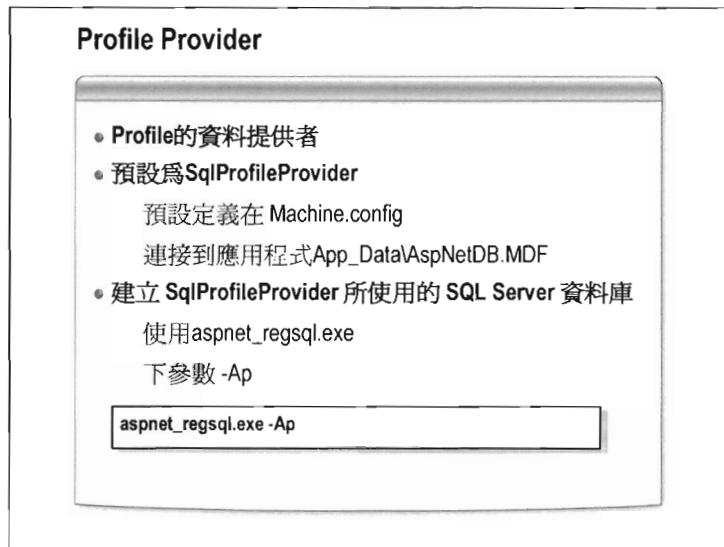


## 認識使用者設定檔

使用者設定檔提供 ASP.NET 設計網頁客製化的能力，包含使用者對於網頁外觀、個人資訊，線上交易系統的購物車...等個人化的功能。它也是一個多層式架構，使用者設定檔可以存在各類型資料庫。

開發人員可以利用 Profile API 存取網站使用者的個人資訊，並將個人化資訊套用於網頁各處適用的地方。

- **控制項**：Profile API 並沒有直接支援某個特定控制項。但可以利用控制項呈現使用者設定檔的資訊。
- **資料庫類型與 Provider**：屬於資料邏輯層負責處理如何存取後端資料庫，ASP.NET 內建 SqlProfileProvider 可以連接到 SQL Server 資料庫。Oracle 及其他類型的資料庫，則必須自行撰寫 Profile Provider。
- **Profile API**：Profile API 主要有 ProfileManager 與 ProfileInfo 等類別。



## Profile Provider

Profile Provider 是 ASP.NET Profile 管理的資料提供者。主要的預設設定是儲存在 Machine.config 中，其 Provider 名稱為 AspNetSqlProfileProvider，類別是 SqlProfileProvider。

```
<profile>
  <providers>
    <add name="AspNetSqlProfileProvider"
        connectionStringName="LocalSqlServer"
        applicationName="/"
        type="System.Web.Profile.SqlProfileProvider" />
  </providers>
</profile>
```

## 建立 SqlProfileProvider 所使用的 SQL Server 資料庫

使用 profile 可以將資料存放在資料庫中，當然此資料庫的預設結構必須符合 Profile 儲存的方式，因此可以使用 SDK 中所提供的 Aspnet\_regsql.exe 來建置資料庫。另外在建置資料庫時必須加上參數 -Ap，代表要建立的資料庫類型是給 profile 用的。

```
aspnet_regsql.exe -Ap
```

### 建立Profile屬性

- Web.config 定義個人化屬性

使用<profile>

可指定型別

可提供預設值

```
<system.web>
<profile>
```

```
<properties>
```

```
<add name="FirstName" />
```

```
<add name="Birthday" />
```

```
</properties>
```

```
</profile>
```

```
</system.web>
```

- 使用HttpContext 的Profile屬性存取

```
TextBox1.Text = Profile[
```

- FirstName
- LastName
- City
- State
- ZipCode

## 建立 Profile 屬性

個人化資訊非常多元化，可能是使用者個人資料或者網頁外觀，隨著網站建構者的規劃，它可以有各種屬性。因此，建立使用者設定檔的第一步就是定義個人化屬性。因為使用者設定檔的多元化特性，它的定義方式要簡單而且要易於改變，因此 ASP.NET 定義使用者設定檔的方式是寫在 Web.config 上，以便於網站開發者容易修改。

Web.config 的設定如下：

```
<configuration>
<system.web>
  <profile>
    <properties>
      <add name="FirstName" />
      <add name="Birthday" />
    </properties>
  </profile>
</system.web>
</configuration>
```

## 指定 Profile 資料型別

當 Profile 屬性定義時並未指定型別時，預設為 String 型別。如果 Profile 屬性需要儲存為特定型別，也可以使用<add>的 type Attribute 指定型別。例如下列 Web.config 的設定：

```
<profile>
  <properties>
    <add name="FirstName" />
    <add name="BirthDay" type="System.DateTime" />
    <add name="Married" type="System.Boolean" />
  </properties>
</profile>
```

## 提供預設值

也可以為 Profile 屬性指定預設值，當屬性值未指定值時，該屬性便是預設值。

```
<add name="Married" type="System.Boolean"
      defaultValue="false" />
```

### 練習 6.4: 存取Profile 的個人資料

在這個練習中，將了解如何設置網站的Profile設定，並且在網頁中存取Profile的個人資料。

預估實作時間：15分鐘



### 練習 6.4 : 存取 Profile 的個人資料

#### 目的：

在這個練習中，將了解如何設置網站的 Profile 設定，並且在網頁中存取 Profile 的個人資料。

#### 功能描述：

在這個練習將針對已經既有存在兩個帳號：admin 和 user 的網站系統，加入在 Web.config 設定 Profile 屬性，並且在網頁上實作 Profile 屬性的存取。

預估實作時間：15 分鐘

#### 實作步驟：

1. 從『Start』→『Program』→『Microsoft Visual Studio 2008』→『Microsoft Visual Studio 2008』，啓動 Visual Studio 2008 開發環境。
2. 從『File』→『Open Web Site』→選取『ASP.NET Web Site』→選擇『File System』設定 Folder 選取『U9544\Practices\VB 或 CS\Mod06\_4\Starter\Mod06\_4』目錄，點選『Open』。

3. 開啓 web.config，定義 Profile 屬性，加入 UserName、Birthday、Married 等屬性名稱。

```
<profile>
<properties>
<add name="UserName" />
<add name="Birthday" type="System.DateTime" />
<add name="Married" type="System.Boolean" />
<add name="ThemeName" type="System.String" />
</properties>
</profile>
```

4. 從「Solution Explorer」開啓 Default.aspx，並切換到「Source」編輯器。
5. 在 Page\_Load 事件中，將 Profile 資訊顯示在 TextBox 及 CheckBox 控制項。

Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
If Not IsPostBack Then
    Label1.Text = User.Identity.Name
    TextBox1.Text = Profile.UserName
    TextBox2.Text = Profile.Birthday.ToShortDateString()
    CheckBox1.Checked = Profile.Married
    DropDownList1.SelectedValue = Profile.ThemeName
End If
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack )
    {
        Label1.Text = User.Identity.Name;
        TextBox1.Text = Profile.UserName;
        TextBox2.Text = Profile.Birthday.ToShortDateString();
        CheckBox1.Checked = Profile.Married;
        DropDownList1.SelectedValue = Profile.ThemeName;
    }
}
```

6. 當按下「存檔」按鈕時，將輸入的「個人資料」存到 Profile 物件。在「存檔」按鈕的 Click 事件中，加入以下程式碼：

Visual Basic

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Profile.UserName = TextBox1.Text
```

```

Profile.Birthday = DateTime.Parse(TextBox2.Text)
Profile.Married = CheckBox1.Checked
Profile.ThemeName = DropDownList1.SelectedValue
Response.Write("個人資料設定成功")
End Sub

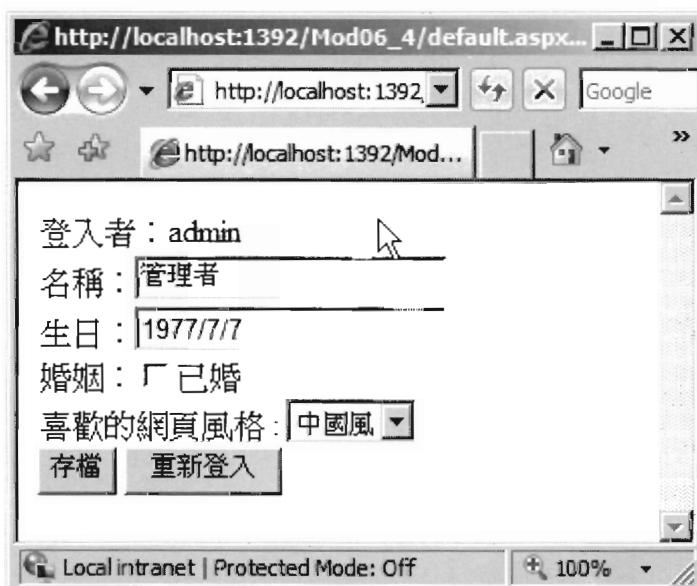
```

```

C#
protected void Button1_Click(object sender, EventArgs e)
{
    Profile.UserName = TextBox1.Text;
    Profile.Birthday = DateTime.Parse(TextBox2.Text);
    Profile.Married = CheckBox1.Checked;
    Profile.ThemeName = DropDownList1.SelectedValue;
    Response.Write("個人資料設定成功");
}

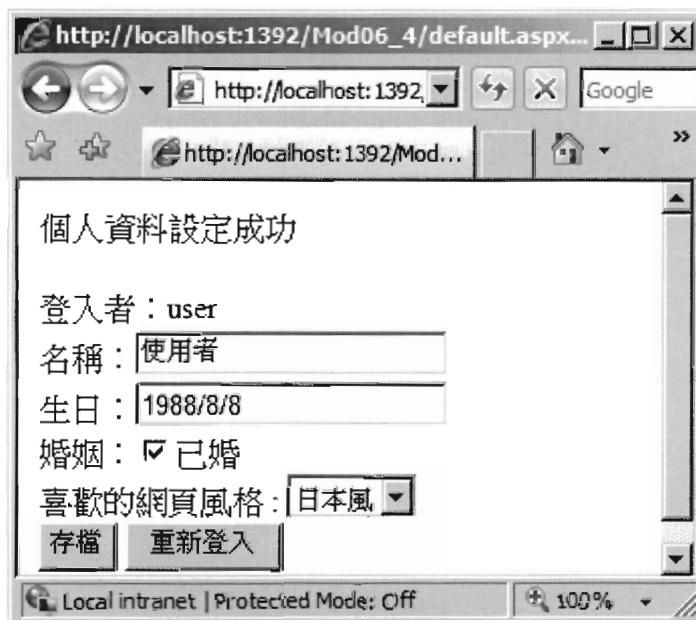
```

7. 執行網頁測試。在「Solution Explorer」→點選「Login.aspx」網頁→按滑鼠右鍵→選「View In Browser」。
8. 輸入帳號：admin，密碼：P@ssw0rd，按下「Log in」，登入後會導向 Default.aspx 網頁。
9. 填入 admin 這個使用者的資訊，如下圖所示，並按下存檔。



10. 按下「重新登入」按鈕，回到「Login.aspx」網頁，重新登入讓網頁導向 Default.aspx 網頁，確認是否已經保存 admin 使用者個人資訊。

11. 針對使用者 user 執行步驟 7 至步驟 10 相同的測試步驟。在「Solution Explorer」→點選「Login.aspx」網頁→按滑鼠右鍵→選「View In Browser」。
12. 輸入帳號：user，密碼：P@ssw0rd，按下「Log in」，登入後會導向 Default.aspx 網頁。填入 user 這個使用者的資訊，如下圖所示，並按下存檔。



13. 按下「重新登入」按鈕，回到「Login.aspx」網頁，重新登入讓網頁導向 Default.aspx 網頁，確認是否已經保存 user 使用者個人資訊。

## Profile整合Theme

### • Profile整合佈景主題使用

```
Visual Basic
<script runat="server">
    Protected Sub Page_PreInit(ByVal sender As Object, ByVal e
    As System.EventArgs)
        Page.Theme = Profile.ThemeID
    End Sub
</script>
```

```
C#
<script runat="server">
    protected void Page_PreInit(object sender, EventArgs e)
    {
        Page.Theme = Profile.ThemeID;
    }
</script>
```

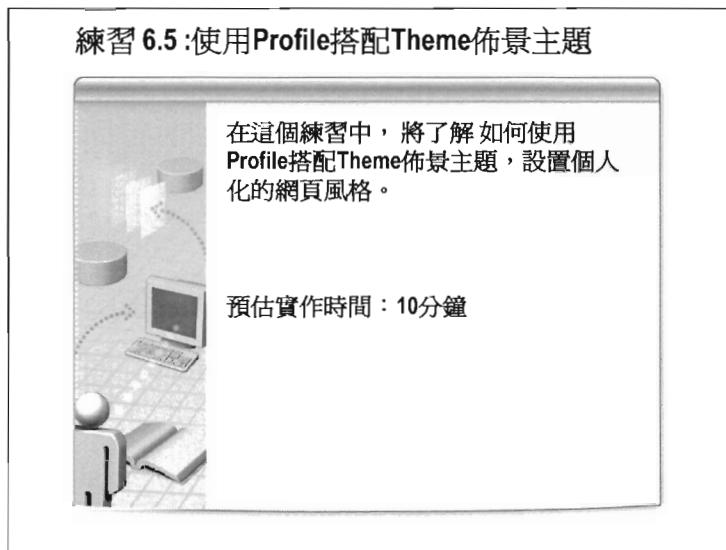
## Profile 整合 Theme

利用撰寫程式來動態控制佈景主題，可以加上整合個人使用設定檔讓每一個使用者都能擁有自己的網站特色。要做到這樣的功能只需要將佈景主題的名稱記錄到個人的 profile 檔即可。

只需要在 Page 物件的 PreInit 事件中，變更 Page 物件的 Theme 屬性，如下：

```
Visual Basic
<script runat="server">
    Protected Sub Page_PreInit(ByVal sender As Object, ByVal e
    As System.EventArgs)
        Page.Theme = Profile.ThemeID
    End Sub
</script>
```

```
C#
<script runat="server">
    protected void Page_PreInit(object sender, EventArgs e)
    {
        Page.Theme = Profile.ThemeID;
    }
</script>
```



## 練習 6.5 : 使用 Profile 搭配 Theme 佈景主題

目的：

在這個練習中，將了解如何使用 Profile 搭配 Theme 佈景主題，設置個人化的網頁風格。

功能描述：

在這個練習中，將了解如何使用 Profile 搭配 Theme 佈景主題，設置個人化的網頁風格。

預估實作時間：10分鐘

實作步驟：

1. 請延續練習專案 Mod06\_4，或是開啓「\U9544\Practices\VB 或 CS\Mod06\_5\Starter\Mod06\_5」網站。
2. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。

3. 從「File」→「Open Web Site」→選取「ASP.NET Web Site」→選擇「File System」設定 Folder 選取「\U9544\Practices\VB 或 CS\Mod06\_5\Starter\Mod06\_5」目錄，點選「Open」。
4. 在專案中，加入一個佈景主題 Theme，命名為「CH」。選取專案→右鍵→「Add ASP.NET Folder」→「Theme」，命名為：「CH」。
5. 在「CH」目錄下加入 Skin 檔。並在 Skin 中針對 Button 控制項和 TextBox 控制項設定背景顏色，如紅色系。
6. SkinFile.skin 檔設定為：

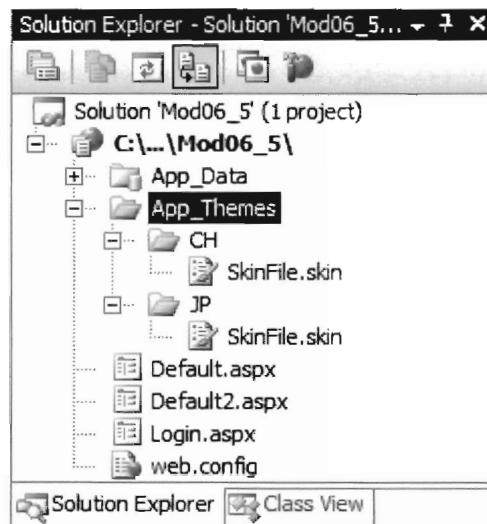
CH:

```
<asp:Button runat="server" BackColor="Red" />
<asp:TextBox runat="server" BackColor="Red" />
```

7. 設定好之後，接著在 Theme 下新增一個「JP」目錄，將「CH」中的 Skin 檔複製過去，改為藍色系。

JP:

```
<asp:Button runat="server" BackColor="Blue" />
<asp:TextBox runat="server" BackColor="Blue" />
```



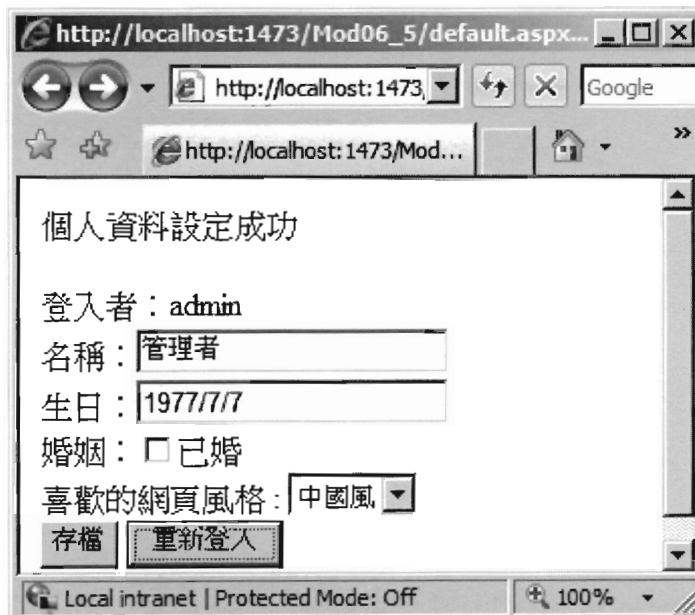
8. 從「Solution Explorer」開啓 Default.aspx，並切換到「Source」編輯器。
9. 在 Page\_PreInit 事件中，動態設定將 Profile 中的 ThemeName 指定給 Page 的 Theme 屬性。

Visual Basic

```
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As System.EventArgs)
    Page.Theme = Profile.ThemeName
End Sub
```

```
C#
protected void Page_PreInit(object sender, EventArgs e)
{
    Page.Theme = Profile.ThemeName;
}
```

10. 執行網頁測試。在「Solution Explorer」→點選「Login.aspx」網頁→按滑鼠右鍵→選「View In Browser」。
11. 輸入帳號：admin 或是 user，密碼：P@ssw0rd，按下「Log in」，登入後會導向 Default.aspx 網頁。
12. 填入 admin 或是 user 這個使用者的資訊，並挑選喜歡的網頁風格，如下圖所示，並按下存檔。



13. 按下「重新登入」按鈕，回到「Login.aspx」網頁，重新登入讓網頁導向 Default.aspx 網頁，將可看到已套用佈景主題。

http://localhost:1473/Mod06\_5/default.aspx... [ ] [ ] [ ]

http://localhost: 1473 [ ] Google

http://localhost:1473/Mod... [ ] [ ] [ ]

登入者 : admin  
名稱 : [REDACTED]  
生日 : [REDACTED]  
婚姻 :  已婚  
喜歡的網頁風格 : [中國風] [ ]  
[ ] 重新登入

Local intranet | Protected Mode: Off 100% [ ]

http://localhost:1473/Mod06\_5/default.aspx... [ ] [ ] [ ]

http://localhost: 1473 [ ] Google

http://localhost:1473/Mod... [ ] [ ] [ ]

登入者 : user  
名稱 : [REDACTED]  
生日 : [REDACTED]  
婚姻 :  已婚  
喜歡的網頁風格 : [日本風] [ ]  
[ ] [ ]

Local intranet | Protected Mode: Off 100% [ ]

## 總結

- 狀態管理機制簡介
- Session物件設計
- Application物件應用
- WebFarm架構時狀態的維護
- ViewState 機制
- Cookie物件設計

---

# 第七章:使用者控制項設計

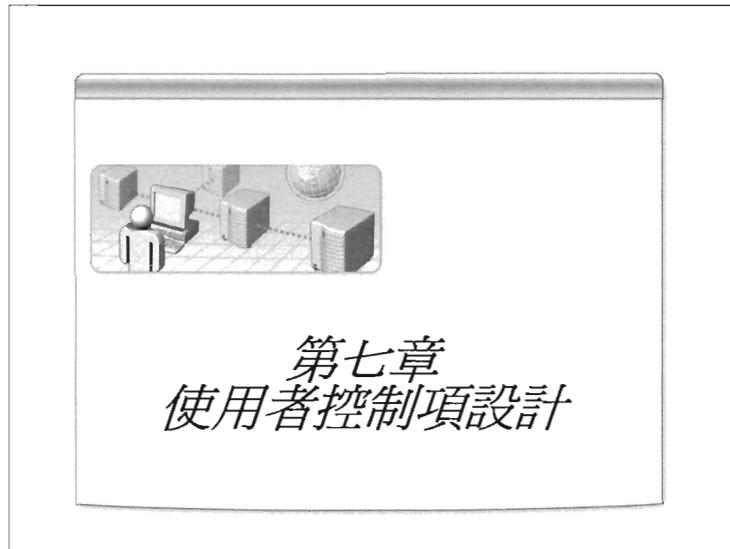
## 本章大綱

何謂 Web User Control .....	4
如何設計 Web User Control .....	5
使用 User Control .....	7
練習 7.1: 設計與使用 User Control .....	8
設計 User Control 自訂屬性 .....	12
使用 User Control 自訂屬性 .....	14
練習 7.2: 存取 User Control 的自訂屬性 .....	15
動態載入 User Control .....	18
練習 7.3: 網頁中動態載入 User Control .....	19
User Control 的狀態維護 .....	21
練習 7.4: 設計維護 User Control 狀態 .....	23



作者：

趙敏翔



---

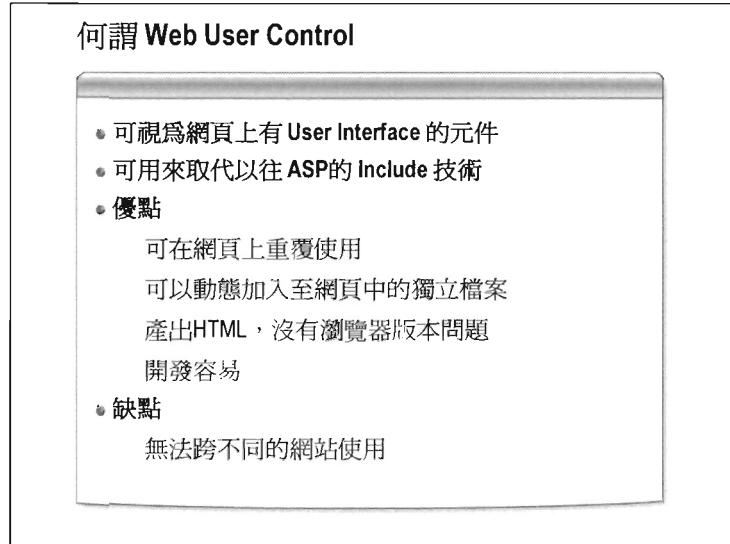
## 大綱

- 何謂 User Control
- 如何設計與使用 User Control
- User Control 自訂屬性
- 動態載入 User Control
- 保存 User Control 的狀態
- 總結

本章將介紹 Web User Control 的設計和開發，Web User Control 是重複使用程式碼與界面的一種技術。在 Web 應用程式中如果有界面或程式碼需要重複使用時，Web User Control 將會相當地有用，同時也可以達到模組化的目的。

在這一章中將學習到

- 何謂 User Control
- 如何設計與使用 User Control
- User Control 自訂屬性
- 動態載入 User Control
- 保存 User Control 的狀態



## 何謂 Web User Control

如果以前就熟悉網頁開發的技術，那麼可以把 Web User Control 當作是 SSI (Server Side Include) 的加強版。Web User Control 的開發方式就如同開發 aspx 網頁一樣，只不過 Web User Control 的指示詞宣告為 Control 且副檔名為 ascx，並且 ascx 的界面與功能會被許多的 aspx 網頁重複使用。

### Web User Control 的優點

Web User Control 的優點除了重複使用的好處外，它們也可以利用程式動態加入至網頁中，讓 Web Form 網頁應用程式的設計更具有彈性。而這些使用者界面基本上全是利用呈現(Render) HTML 產生的，也不會有瀏覽器版本的問題(因為只要是瀏覽器就得認識 HTML)。

### Web User Control 的缺點

Web User Control 的缺點是只能在設計的那個網站上使用，無法跨網站共用，如果要跨網站共用，就必須設計成 Web Custom Control。

## 如何設計 Web User Control

- Web User Control 的特色
  - .ascx 副檔名
  - 可重複使用的畫面以及程式碼
  - User Control 不包含 <html>、<body>、<form>
  - User Control 繼承自 System.Web.UI.UserControl
- 建立 User Control
 

Visual Basic <%@ Control Language="VB" ClassName="myUserControl"%>
C# <%@ Control Language="C#" ClassName=" myUserControl "%>

## 如何設計 Web User Control

Web User Control 有幾個主要的特色：

- Web User Control 的副檔名為 ascx，與 Web Form 網頁應用程式的副檔名 aspx 不同。
- Web User Control 可以在 ascx 檔中撰寫程式。
- Web User Control 不包含 <html>、<body>、或<form>標籤。
- Web User Control 繼承自 System.Web.UI.UserControl 類別。
- Web User Control 擁有使用者界面，可以包含網頁伺服器控制項以及 HTML 控制項。
- Web User Control 可以獨立的設定快取機制以增進整體效能。

## 建立 User Control

在新增建立 Web User Control 時，其 ascx 中的指示詞則設定為 Control，ClassName 則為此控制項的類別名稱：

```
Visual Basic:  
<%@ Control Language="VB" ClassName="myUserControl"%>
```

C#

```
<%@ Control Language="C#" ClassName="myUserControl"%>
```

## 使用User Control

- 將User Control 新增到Page網頁中  
在網頁Page中的最上方使用Register指示詞宣告

```
<% Register ...%>
```

使用 TagPrefix和 TagName屬性

```
<%@ Register Src="myUserControl.ascx"
TagName="myUserControl" TagPrefix="Ucom" %>
```

- User Control 在Page網頁中的顯示

```
<Ucom:myUserControl ID="myUserControl1" runat="server">
```

## 使用 User Control

要在網頁中使用 Web User Control，最快的設計方式是在開發專案中，直接將 ascx 副檔名的 User Control 直接使用拖曳的方式，拉到 aspx 網頁的設計頁面中。

此時，Visual Studio 2008 會自動在 aspx 的網頁中，自動加上使用者控制項的註冊標籤。代表可以在此網頁中參考並使用此使用者控制項，註冊標籤如下：

```
<%@ Register Src="myUserControl.ascx" TagName=" myUserCon
trol " TagPrefix="Ucom" %>
```

並且，此使用者控制項將會依據註冊標籤中的設定，來產生此使用者控制項在這個網頁中的設計標籤：

```
<Ucom:myUserControl ID="myUserControl1" runat="server" />
```

**練習 7.1: 設計與使用 User Control**



設計一個簡便的日曆使用者控制項，了解如何設計 User Control 並且在網頁應用程式中使用 User Control。

預估實作時間：10分鐘

## 練習 7.1：設計與使用 User Control

目的：

設計一個簡便的日曆使用者控制項，了解如何設計 User Control 並且在網頁應用程式中使用 User Control。

功能描述：

在這個練習中，將建立一個常用的日曆使用者控制項，此 User Control 會包含一個 TextBox、一個 Button 和一個 Calendar 控制項，其簡單的使用流程包裝在 User Control 中，當點下 Button 按鈕會將 Calendar 控制項顯示出來，當點選 Calendar 控制項的某一個日期時，則會將所點選的日期顯示在 TextBox 控制項中。

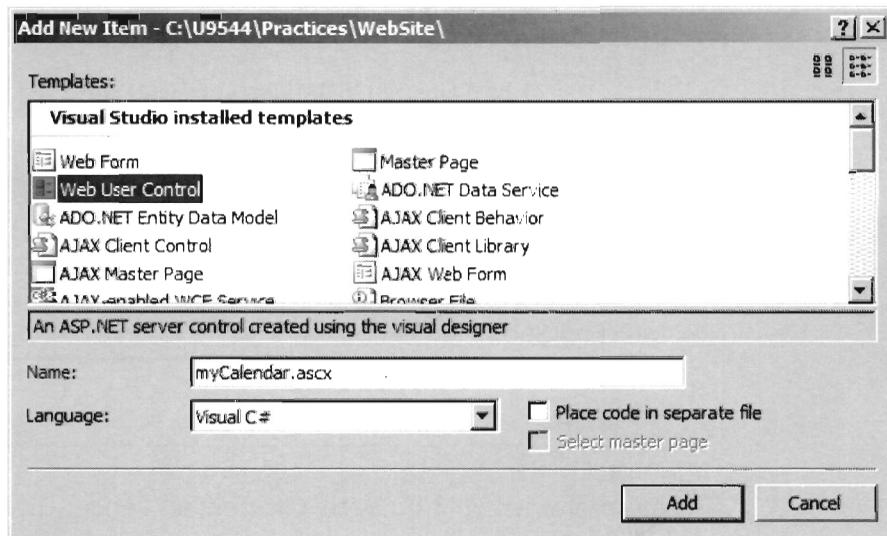
預估實作時間：10 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選

取「\U9544\Practices\VB 或 CS\Mod07\_1\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod07\_1」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 在 Web Site 中建立一個新使用者控制項。從 Visual Studio 2008 開發工具「Web Site」選單 → 選取「Add New Item」，選取 Web User Control，清除「Place code in separate file」核取方塊，將檔案命名為 myCalendar.ascx。



5. 從「Toolbox」工具箱中拖拉一個 TextBox 控制項到 myCalendar.ascx 中並將其命名為「txtDate」。
6. 拖拉一個 Button 控制項到 myCalendar.ascx 中，並將其命名為「btnDate」。
7. 拖拉一個 Calendar 控制項到 myCalendar.ascx 中並將其命名為「CalDate」，並將 CalDate 的 Visible 屬性設定為「false」。
8. 點選畫面上的 Calendar 控制項之智慧型標籤 → 選「Auto Format」選單進行格式化。



9. 在「Solution Explorer」→點選 myCalendar.ascx →按滑鼠右鍵 →選「View Code」。

10. 在 btnDate 的 Click 事件中將 CalDate 日曆控制項的 Visible 設為 true。

```
Visual Basic
Protected Sub btnDate_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    CalDate.Visible = True
End Sub
```

```
C#
protected void btnDate_Click(object sender, EventArgs e)
{
    CalDate.Visible = true;
}
```

11. 在 CalDate 的 SelectionChanged 事件中將選到的日期值取出在 txtDate 的 Text 屬性中，並將 CalDate 日曆控制項隱藏。

```
Visual Basic :
Protected Sub CalDate_SelectionChanged(ByVal sender As Object,
    ByVal e As System.EventArgs)
    txtDate.Text = CalDate.SelectedDate.ToShortDateString()
    CalDate.Visible = False
End Sub
```

```
C# :
protected void CalDate_SelectionChanged(object sender, EventArgs e)
{
```

```

txtDate.Text = CalDate.SelectedDate.ToShortDateString();
CalDate.Visible = false;
}

```

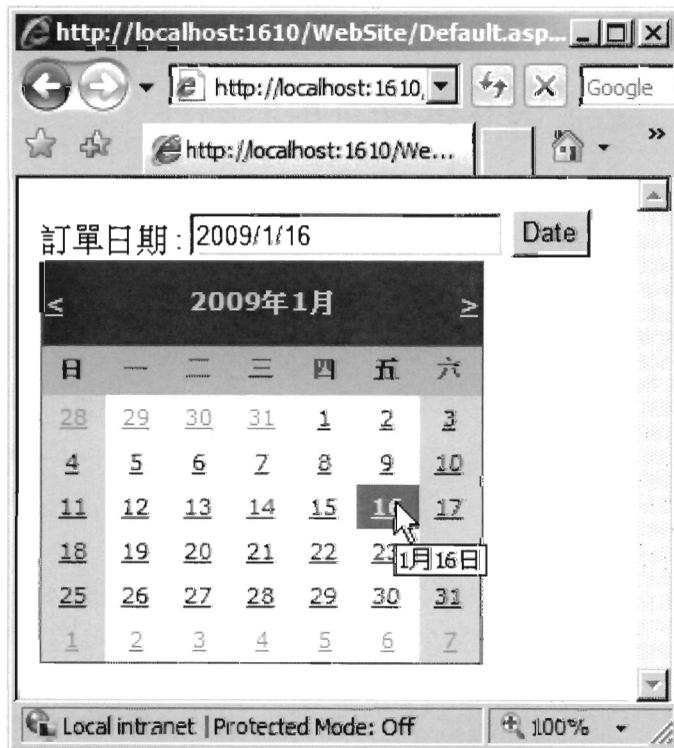
12. 直接將 myCalendar.ascx 使用拖拉的方式拉到 Default.aspx 中，修改 Register 指示詞宣告，如下：

```
<%@ Register src="myCalendar.ascx" tagname="myCalendar"
tagprefix="Ucom" %>
```

13. form 表單中會自動加入控制項。

```
<Ucom:myCalendar ID="MyCalendar1" runat="server" />
```

14. 執行網頁測試。在「Solution Explorer」→點選 myCalendar.aspx 網頁→按滑鼠右鍵→選「View In Browser」。試著將選到的日期顯示在 txtDate 控制項上。



**設計User Control自訂屬性**

- 設計控制項封裝屬性
- 新增User Control 的自訂屬性

```
Visual Basic
Public ReadOnly Property myData() As String
    Get
        ...
    End Get
End Property
```

```
C#
public string myData
{
    get { ... }
}
```

## 設計 User Control 自訂屬性

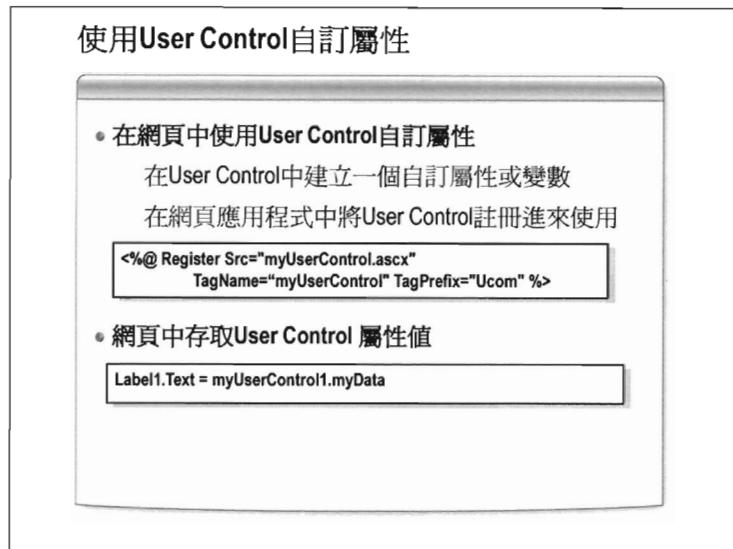
如果只是這樣子去使用 User Control，這就像買了一部飛機，卻只讓它在跑道上跑一樣浪費，這樣就太委屈 User Control 了。

User Control 真正強的地方是它爲了提供彈性，可以自行定義屬性與方法，比如說希望能夠將 User Control 中其中一個子控制項的資料取出來使用。

假設想要將 User Control 其中一個 TextBox 控制項的值，可以在其他的網頁應用程式中可以順利的讀取，並且設計爲只能讀取，則可以使用屬性封裝的方式來設計，User Control 自訂屬性程式碼參考如下：

```
Visual Basic
Public ReadOnly Property myData() As String
    Get
        Return txtDate.Text
    End Get
End Property
```

```
C#
public string myData
{
    get
    {
        return txtDate.Text ;
    }
}
```



## 使用 User Control 自訂屬性

有時候開發者可能會覺得將使用者控制項中的子控制項，直接開放出來存取不是更簡易嗎？但是，如果直接開放存取子控制項，那麼：

1. 可能無法周密的規劃安全性的考量。
2. 無法加入額外的條件判斷。
3. 設計彈性變小。

因此使用屬性封裝的方式就很適合。

假設 User Control 已經開放一個屬性值叫做「myText」，則在網頁應用程式中要去存取此屬性值的程式碼參考如下：

```
Label1.Text = MyCalendar1.myText
```

### 練習 7.2 :存取User Control的自訂屬性

了解如何設計 User Control 的自訂屬性和方法，並且在網頁應用程式中使用其自訂的屬性存取 User Control 中某一控制項的值。

預估實作時間：10分鐘



### 練習 7.2 : 存取 User Control 的自訂屬性

#### 目的：

了解如何設計 User Control 的自訂屬性和方法，並且在網頁應用程式中使用其自訂的屬性存取 User Control 中某一控制項的值。

#### 功能描述：

在這個練習中，將學習如何將日曆控制項 User Control 的自訂屬性和方法，在網頁應用程式中取出來使用。先在 User Control 中自訂一個屬性 myDate 用來回傳 User Control 中 TextBox 的值，以方便讓網頁應用程式去存取。

預估實作時間：10 分鐘

#### 實作步驟：

1. 請延續練習專案 Mod07\_1，或是開啓『U9544\Practices\VB 或 CS\Mod07\_2\Starter\Mod07\_2』網站。
2. 從『File』→『Open Web Site』選取『ASP.NET Web Site』選擇『File System』設定 Folder 選取『U9544\Practices\VB 或 CS\Mod07\_2\Starter\Mod07\_2』目錄，點選『Open』。

3. 在「Solution Explorer」→點選 myCalendar.ascx →按滑鼠右鍵 →選「View Code」。

4. 在 myCalendar.ascx 控制項自訂屬性 myDate。加上自訂屬性程式碼，程式碼撰寫完畢之後，存檔。

```
Visual Basic
Public ReadOnly Property myDate() As String
    Get
        Return txtDate.Text
    End Get
End Property
```

```
C#
public string myDate
{
    get
    {
        return txtDate.Text;
    }
}
```

5. 在「Solution Explorer」→點選 Default.aspx →按滑鼠右鍵 →選「View Designer」。

6. 從「Toolbox」工具箱中拖拉一個 Button 控制項到 Default.aspx 中並將其命名為「btnGetDate」，Text 屬性設定為「取得訂單日期」。

7. 接著從「Toolbox」工具箱中拖拉一個 Label 控制項網頁中並將其命名為「lblDate」。

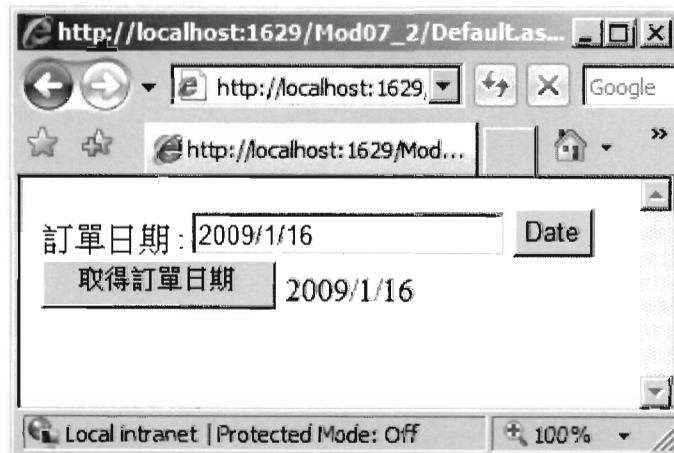
8. 在 btnGetDate 的 Click 事件中存取 myCalendar 所開放的屬性值，將其值顯示在 lblDate 上。

```
Visual Basic
Protected Sub btnGetDate_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    lblDate.Text = myCalendar1.myDate
End Sub
```

```
C#
protected void btnGetDate_Click(object sender, EventArgs e)
{
```

```
    }   lblDate.Text = myCalendar1.myDate;
```

9. 執行網頁測試。在「Solution Explorer」→點選 Default.aspx 網頁→按滑鼠右鍵→選「View In Browser」。先點選到日期，再按下「取得訂單日期」將日期顯示在 Label 控制項上。



### 動態載入 User Control

- 網頁中動態載入 User Control  
彈性的在執行時期載入 Web User Control  
使用Page.LoadControl  
User Control 必須置放在Form表單或容器控制項中

**Visual Basic**

```
Dim control as Control = Page.LoadControl("myUserControl.ascx")
PlaceHolder1.Controls.Add(control)
```

**C#**

```
Control control = Page.LoadControl("myUserControl.ascx");
PlaceHolder1.Controls.Add(control);
```

## 動態載入 User Control

在 User Control 的設計使用上，有時後會根據某些商業邏輯，而決定在應用程式執行時期才動態去載入這個 User Control，這時必須使用到 Page 物件的 LoadControl 方法來載入運作。

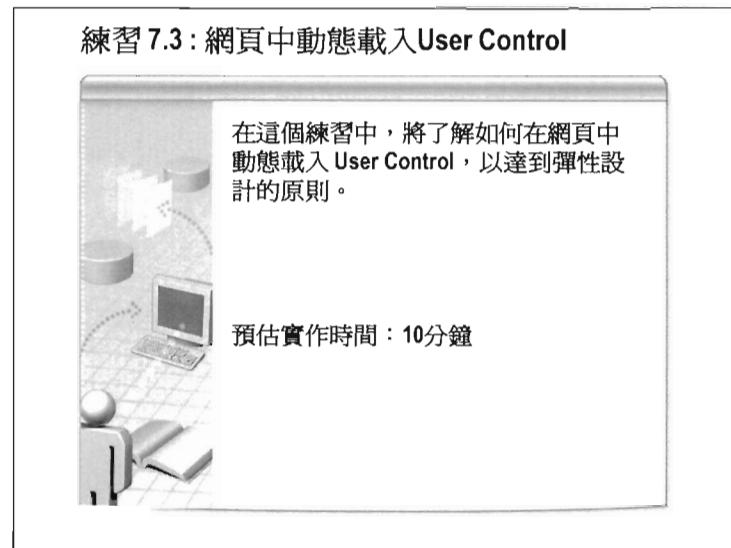
建議使用動態載入使用者控制項時，可以先在 Page 網頁上先加入一個 PlaceHolder 控制項當作控制項的容器，好處是可以隨時調整他的所在位置。在 PlaceHolder 控制項中載入 User Control，程式寫法參考如下：

**Visual Basic**

```
Dim obj As Control = Page.LoadControl("myUserControl.ascx")
PlaceHolder1.Controls.Add(obj)
```

**C#**

```
Control obj = Page.LoadControl("myUserControl.ascx");
PlaceHolder1.Controls.Add(obj);
```



## 練習 7.3 : 網頁中動態載入 User Control

目的：

在這個練習中，將了解如何在網頁中動態載入 User Control，以達到彈性設計的原則。

功能描述：

在這個練習中，將在網頁應用程式執行時期，動態載入所設計好的日曆使用者控制項。

預估實作時間：10 分鐘

實作步驟：

1. 請延續練習專案 Mod07\_2，或是開啟「\U9544\Practices\VB 或 CS\Mod07\_3\Starter\Mod07\_3」網站。
2. 從「File」→「Open Web Site」選取「ASP.NET Web Site」選擇「File System」設定 Folder 選取「\U9544\Practices\VB 或 CS\Mod07\_3\Starter\Mod07\_3」目錄，點選「Open」。

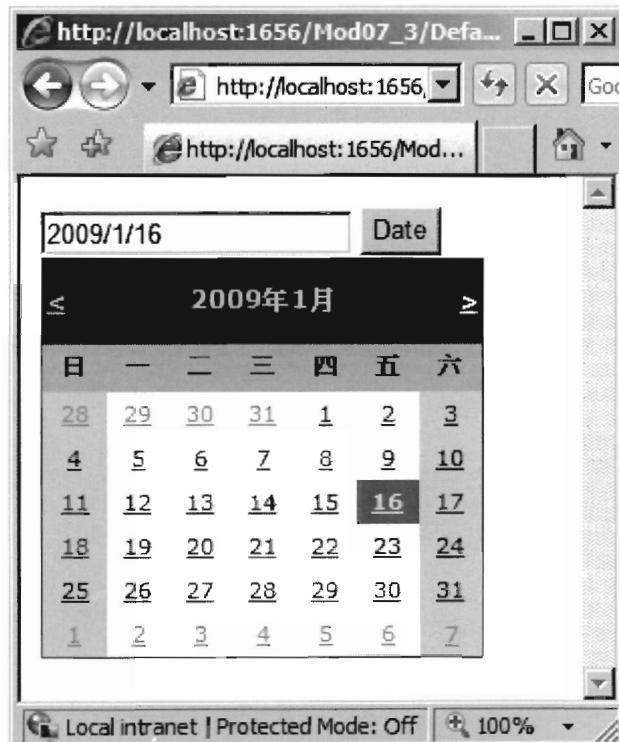
3. 在「Solution Explorer」→點選 Default.aspx →按滑鼠右鍵 →選「View Designer」新增一個 PlaceHolder 控制項到網頁中。

4. 在網頁的 Page\_Load 事件中動態載入 myCalendar 控制項到網頁中，並且動態註冊到 PlaceHolder 控制項。

```
Visual Basic
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim obj As Control = Page.LoadControl("myCalendar.ascx")
    PlaceHolder1.Controls.Add(obj)
End Sub
```

```
C#
protected void Page_Load(object sender, EventArgs e)
{
    Control obj = Page.LoadControl("myCalendar.ascx");
    PlaceHolder1.Controls.Add(obj);
}
```

5. 執行網頁測試。在「Solution Explorer」→點選網頁 →按滑鼠右鍵 →選「View In Browser」。可看到動態載入的日曆控制項。



### User Control 的狀態維護

- 搭配動態載入 User Control 頁面設計
- 必須改寫 LoadViewState 以維護狀態

讓資料在伺服器往返間保存的有效方式

```
Visual Basic
Protected Overrides Sub LoadViewState(ByVal savedState As Object)
    MyBase.LoadViewState(savedState)
    ...
End Sub
```

```
C#
protected override void LoadViewState(object savedState)
{
    base.LoadViewState(savedState);
    ...
}
```

### User Control 的狀態維護

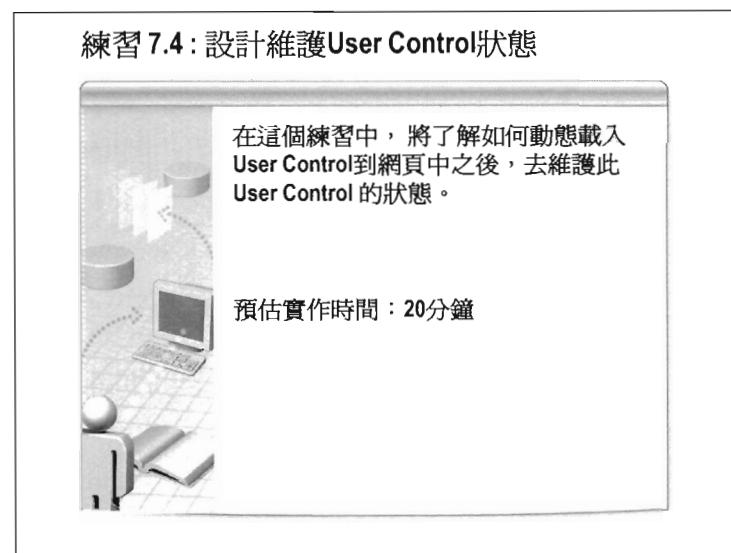
另一個使用者控制項的設計考量是它的狀態維護，例如在網頁上使用者透過按鈕動態將使用者控制項載入到網頁中，那個按下頁面上任何一個控制項，讓 Page 重新 PostBack 回伺服器後，因為 Page 會重新生成，此時使用者所動態加入的控制項就會消失不見。

為了解決這種問題，必須讓控制項的資料在伺服器間往返時，確保其狀態維護的有效方式，改寫 LoadViewState 方法可以達到這樣的效果，範例程式碼如下：

```
Visual Basic
Protected Overrides Sub LoadViewState(ByVal savedState As Object)
    MyBase.LoadViewState(savedState)
    ...
End Sub
```

```
C#
protected override void LoadViewState(object savedState)
{
    base.LoadViewState(savedState);
```





## 練習 7.4 : 設計維護 User Control 狀態

目的：

在這個練習中，將了解如何動態載入 User Control 到網頁中之後，去維護此 User Control 的狀態。

功能描述：

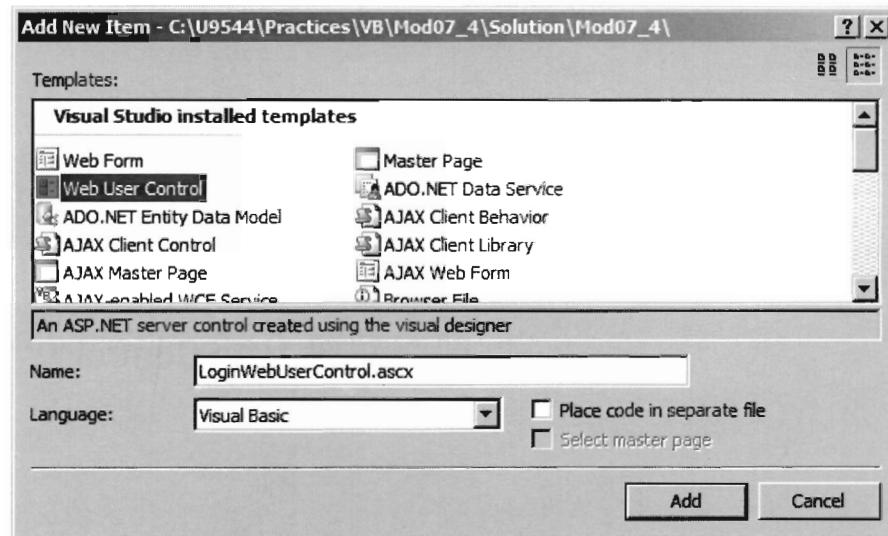
在這個練習中將設計一組可輸入帳號和密碼的使用者控制項，當使用者點選網頁中的啓用功能，將會動態將此使用者控制項載入網頁中，並且自動維護其狀態。

預估實作時間：20 分鐘

實作步驟：

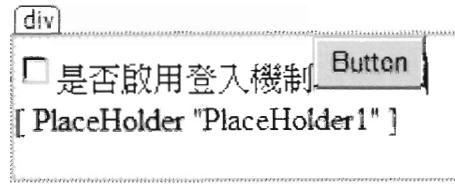
1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「Open Web Site」→選取「ASP.NET Web Site」→選擇「File System」設定 Folder 選取「\U9544\Practices\VB 或 CS\Mod07\_4\Starter\Mod07\_4」目錄，點選「Open」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 在 Web Site 中建立一個新使用者控制項。從 Visual Studio 2008 開發工具「Web Site」選單 → 選取「Add New Item」，選取 Web User Control，清除「Place code in separate file」核取方塊，將檔案命名為 LoginWebUserControl.ascx。



5. 從工具箱中拖拉兩個 TextBox 和一個按鈕到 LoginWebUserControl.ascx 設計畫面中，設計好如下畫面：

6. 從「Solution Explorer」開啓 Default.aspx。
7. 從工具箱中加入一個 CheckBox 控制項到設計頁面中，設定其 Text 屬性為「是否啓用登入機制」，AutoPostBack 屬性為 True。
8. 從工具箱中加入一個 Button 控制項和 PlaceHolder 控制項到設計頁面中。



9. 先加入動態載入使用者控制項的程式碼，使用一個 ViewState 物件判斷是否紀錄要動態載入的控制項資料，程式碼如下：

```
Visual Basic
Public Sub LoadControls()
    If ViewState("Login") IsNot Nothing Then
        If ViewState("Login") = True Then
            Dim control As Control = LoadControl("LoginWebUserCon
trol.ascx")
            control.ID = "CtrlLogin"
            PlaceHolder1.Controls.Add(control)
        End If
    End If
End Sub
```

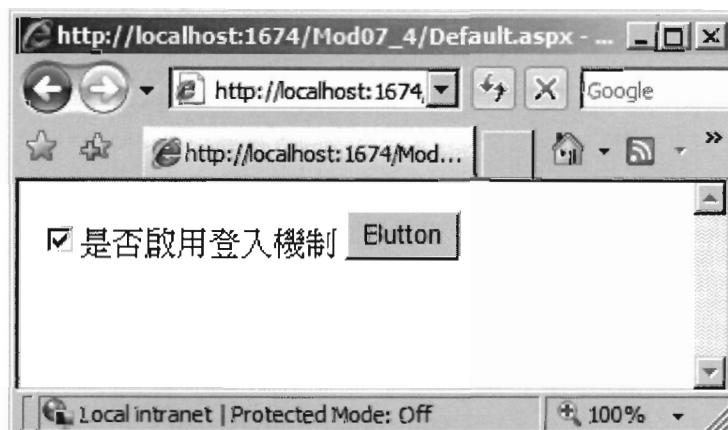
```
C#
public void LoadControls()
{
    if (ViewState["Login"] != null)
    {
        if ((bool)ViewState["Login"] == true)
        {
            Control control = LoadControl("LoginWebUserControl.asc
x");
            control.ID = "CtrlLogin";
            PlaceHolder1.Controls.Add(control);
        }
    }
}
```

10. 在 CheckBox1 的 CheckedChanged 事件中，設定 ViewState 值為 CheckBox1 的 Checked 屬性值，代表是否紀錄想要保留的使用者控制項狀態，並叫用 LoadControls 方法，重新載入控制項，程式碼如下：

```
Visual Basic
Protected Sub CheckBox1_CheckedChanged(ByVal sender As Obj
et, ByVal e As System.EventArgs)
    ViewState("Login") = CheckBox1.Checked
    PlaceHolder1.Controls.Clear()
    LoadControls()
End Sub
```

```
C#
protected void CheckBox1_CheckedChanged(object sender, EventArgs e)
{
    ViewState["Login"] = CheckBox1.Checked;
    PlaceHolder1.Controls.Clear();
    LoadControls();
}
```

11. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。
12. 勾選 CheckBox 可將 LoginWebUserControl.ascx 動態載入，但是再次按下 Button 按鈕時，LoginWebUserControl.ascx 會消失，這是因為尚未控管使用者控制項的狀態。



13. 切換到 Default.aspx，加入維護使用者控制項狀態，改寫 LoadViewState 方法，程式碼如下：

```
Visual Basic
Protected Overrides Sub LoadViewState(ByVal savedState As Object)
    MyBase.LoadViewState(savedState)
    LoadControls()
End Sub
```

```
C#
protected override void LoadViewState(object savedState)
{
    base.LoadViewState(savedState);
    LoadControls();
}
```

14. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」，勾選 CheckBox 並按下按鈕，可發現控制項仍在網頁中。



## 總結

- 了解何謂 User Control
- 了解如何設計與使用 User Control
- 認識User Control 自訂屬性
- 學會動態載入 User Control
- 保存 User Control 的狀態

---

# 第八章: 自訂控制項進階 設計

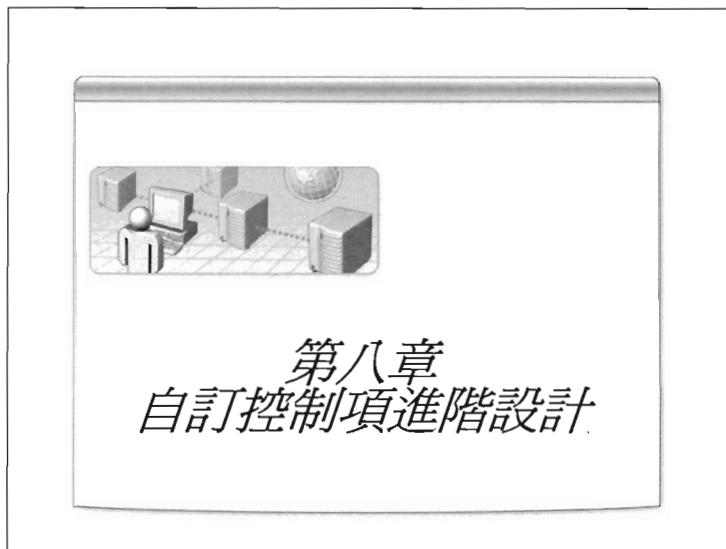
## 本章大綱

什麼是 Web Custom Control.....	4
Web Custom Control 的特色.....	5
設計 Web Custom Control 的步驟.....	7
練習 8.1: 設計一個 Web Custom Control.....	9
在開發工具加入 Web Custom Control .....	12
網頁使用 Web Custom Control .....	14
練習 8.2: 在網站上使用 Web Custom Control.....	16
自訂控制項的狀態維護.....	20
狀態維護設計.....	22
練習 8.3: 設計維護自訂控制項狀態 .....	23
設計 Custom Composite Control .....	29
練習 8.4: 設計 Custom Composite Control .....	31
支援樣版的 Custom Control .....	36
如何設計支援樣版的 Custom Control.....	37
網頁中繫結控制項樣板的資料 .....	39
練習 8.5: 設計支援樣板的自訂控制項.....	40



作者：

趙敏翔



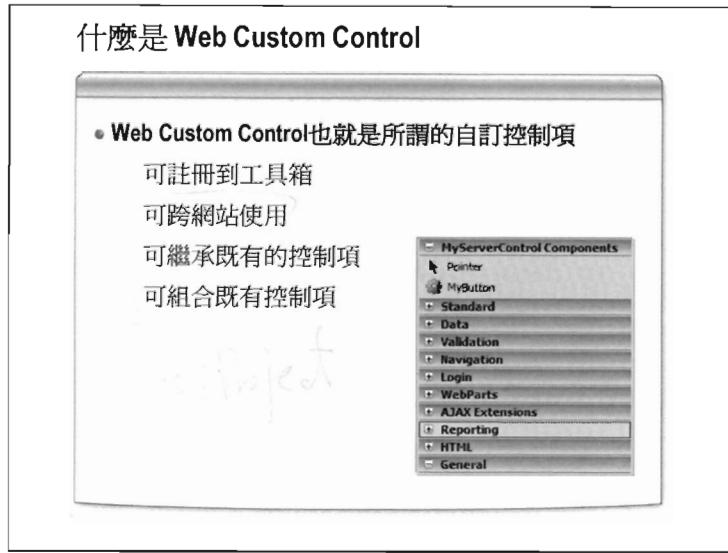
## 大綱

- 什麼是 Web Custom Control
- 設計 ASP.NET Server Control
- 使用 ASP.NET Server Control
- 自訂控制項的狀態維護
- 設計 Composite Web Server Control
- 自訂控制項的樣版設計
- 總結

本章將介紹 Web Custom Control 自訂控制項的開發方式，透過自訂控制項可以將好用並且具備強大彈性化功能的控制項包裝起來，讓其他的網站應用程式使用，已達到真正共用的效果。

在這一章中將學習到

- 什麼是 Web Custom Control
- 設計 ASP.NET Server Control
- 使用 ASP.NET Server Control
- 自訂控制項的狀態維護
- 設計 Composite Web Server Control
- 自訂控制項的樣版設計



## 什麼是 Web Custom Control

Web Custom Control 沒有辦法像 Web User Control 一樣地簡易開發，它沒有設計畫面，全部的界面與屬性都要靠寫程式來產生，當然也因為程式都是可以“自訂”的，開發新功能的彈性相對的也比 Web User Control 好的很多，而且當在 Visual Studio 2008 中要使用它的時候，它提供了良好的設計時期的屬性設定方式。

Web Custom Control 開發時可以選擇直接編寫一個控制項或是把既有的控制項繼承下來改寫，當開發好一個自訂控制項之後，將專屬專案編譯成.dll 組件，接著可以把此組件中的控制項註冊到 Visual Studio 開發工具的工具箱中，讓其它網站參考引用即可。



### Web Custom Control的特色

- Web Custom Control的主要特色

可繼承自 System.Web.UI.Control

    當想設計自訂功能的控制項

可繼承自 System.Web.UI.WebControls.WebControl

    僅想將目前既有的控制項功能改寫

可繼承 System.Web.UI.WebControls.CompositeControl

    使用既有的 ASP.NET 控制項組合

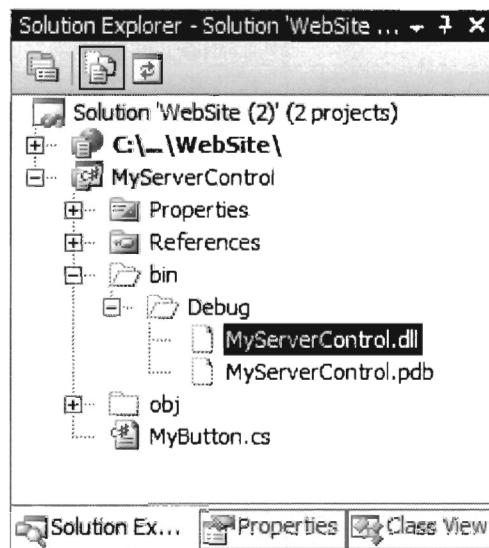
- 自訂控制項使用前，將專案編譯為 dll 組件

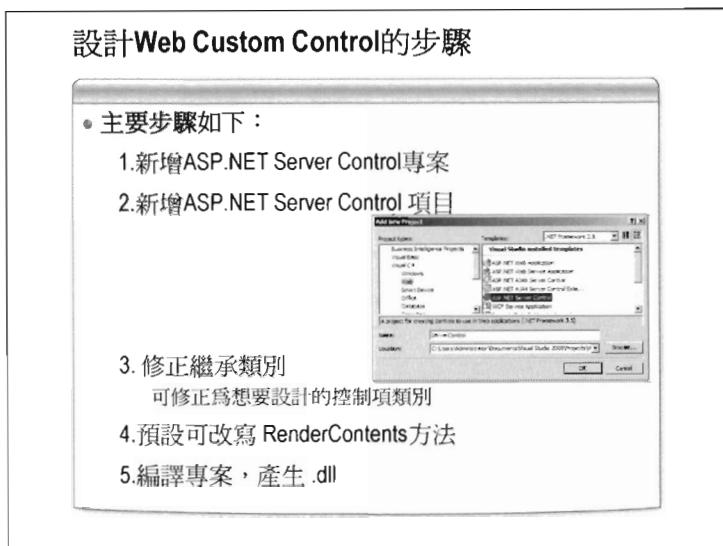
## Web Custom Control 的特色

Web Custom Control 擁有以下的幾個特色：

- 可繼承自 System.Web.UI.Control，當想設計自訂功能的控制項。
- 可繼承自 System.Web.UI.WebControls.WebControl，僅想將目前既有的控制項功能改寫。
- 可繼承 System.Web.UI.WebControls.CompositeControl：使用既有的 ASP.NET 控制項組合。
- 完全使用程式碼來控管 Custom Control 的畫面以及功能。

Web Custom Control 在使用之前必須先將 Custom Control 編譯成組件 (Assembly)。

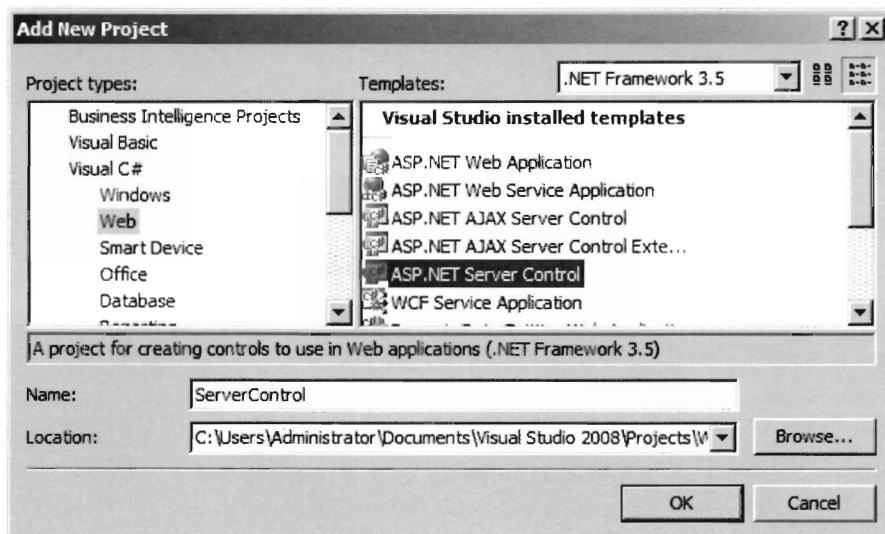




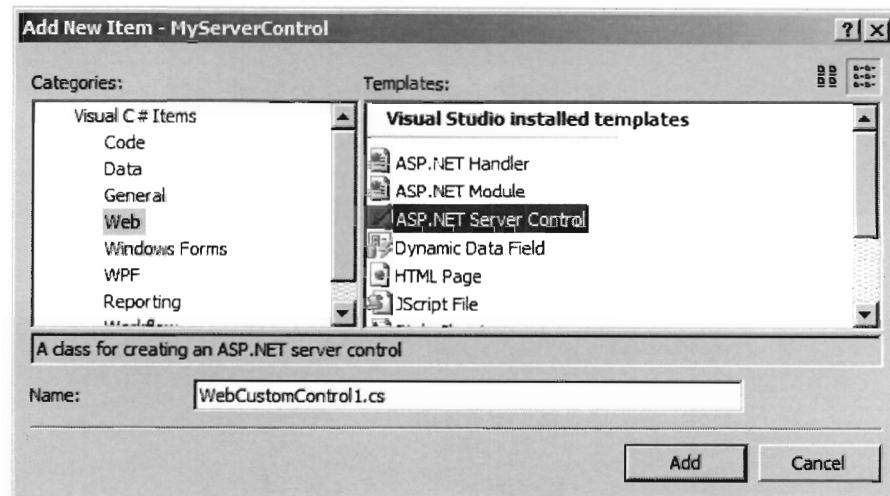
## 設計 Web Custom Control 的步驟

設計 Web Custom Control 的開發的主要步驟如下：

1. 新增 ASP.NET Server Control 專案。



2. 新增 ASP.NET Server Control 項目。



3. 修正繼承類別：因為撰寫 Custom Control 主要有兩大類寫法：一種是繼承既有的控制項來改寫既有的功能或是擴充新功能，這種方式的繼承將要改寫成繼承想要改寫的控制項類別，如 TextBox 控制項；另一種方式則是全部使用程式去產生 Custom Control，且不改寫既有的控制項，這種方式的繼承類別必須改寫為繼承 System.Web.UI.Control。
4. 覆寫 RenderContents 方法：將 RenderContents 方法產生的內容稍作修改，讓顯示出 Text 屬性內容外，再多加上一些說明文字：

```
Visual Basic
Protected Overrides Sub RenderContents(ByVal output As HtmlTextWriter)
    '編輯HTML輸出部分
    output.Write("Custom Control Value : " + Text)
End Sub
```

```
C#
protected override void RenderContents(HtmlTextWriter output)
{
    //編輯HTML輸出部分
    output.Write("Custom Control Value : " + Text);
}
```

5. 編譯 ASP.NET Server Control 專案。

### 練習 8.1: 設計一個 Web Custom Control

在這個練習中，你將學習瞭解如何使用 ASP.NET Server Control 專案樣板，設計一個 Web Custom Control 專屬專案，並且設計一個簡易的 Web Custom Control。

預估實作時間：10分鐘

## 練習 8.1：設計一個 Web Custom Control

### 目的：

在這個練習中，你將學習瞭解如何使用 ASP.NET Server Control 專案樣板，設計一個 Web Custom Control 專屬專案，並且設計一個簡易的 Web Custom Control。

### 功能描述：

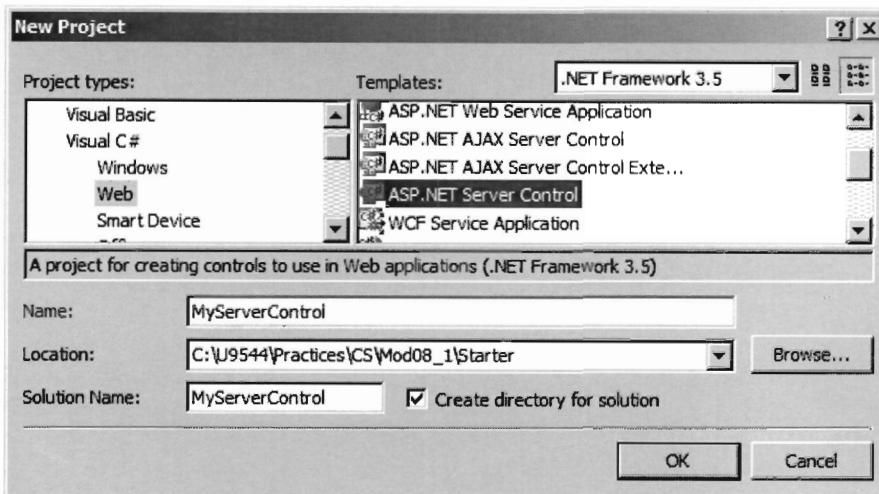
在這個練習中，將使用 ASP.NET Server Control 專案建立一個自訂控制項 MyLabel，當設定的資料有包含 ASP.NET 字串時將會把這個字改變顏色。

**預估實作時間：10 分鐘**

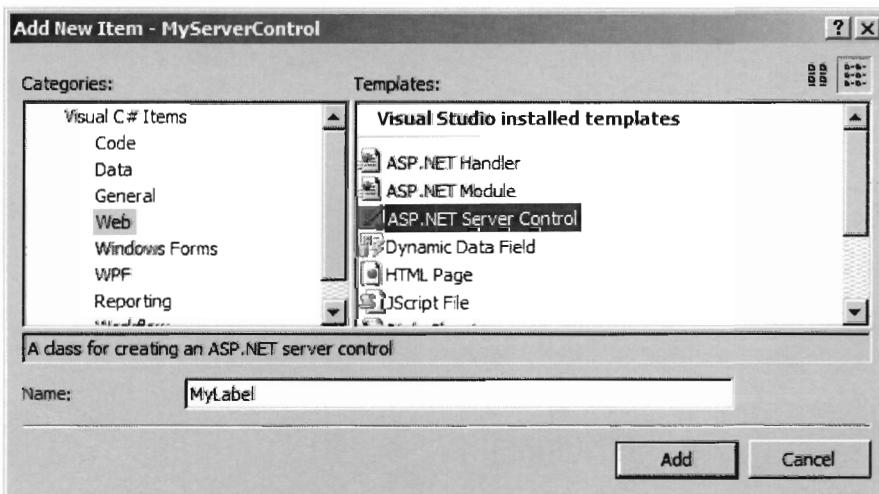
### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Project」→「Project types」選取「Web」→「templates」選取「ASP.NET Server Control」→點選「Browse...」按鈕將「Location」設為「\U9544\Practices\VB」

或 CS\Mod08\_1\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將 Name 取名為「MyServerControl」。



3. 刪除預設的 ServerControl1.cs 或 ServerControl1.vb。
4. 自主選單「Project」下「Add New Item...」，Categories 選取「Web」→，Templates 選取「ASP.NET Server Control」，並將名稱改為 MyLabel→按下「Add」。



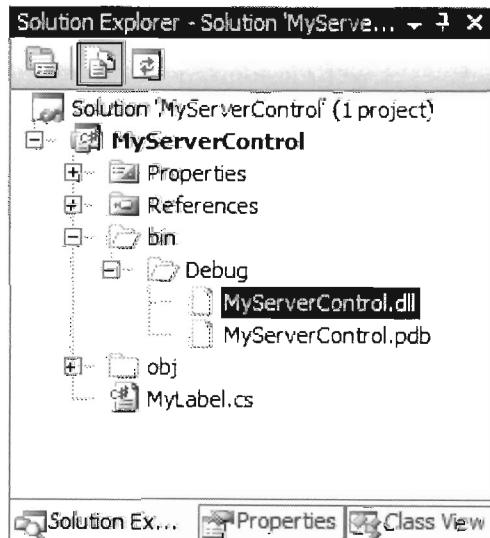
5. 修改 MyLabel 的 RenderContents 方法，加上此控制項想要自訂的行爲模式，例如：當此控制項的 Text 值有出現 ASP.NET 時，就把字變顏色，程式碼如下：

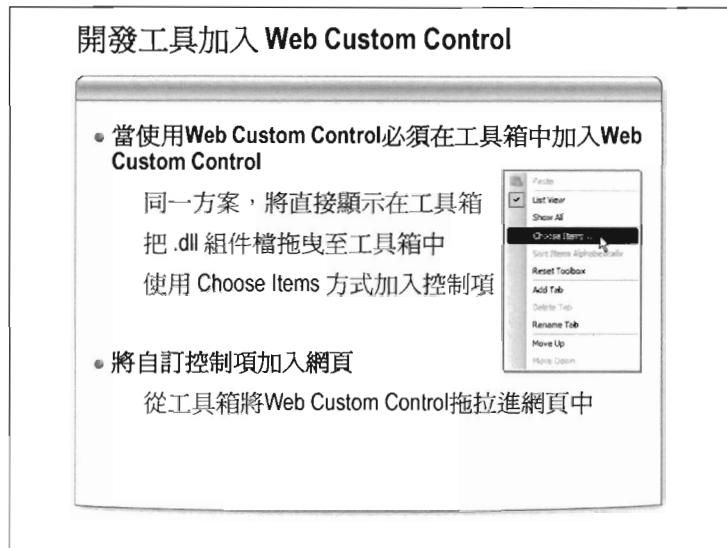
```
Visual Basic
Protected Overrides Sub RenderContents(ByVal writer As HtmlTextWriter)
    ' Your custom rendering logic here
End Sub
```

```
If Text.Contains("ASP.NET") Then  
    Text.Replace("ASP.NET", "<Font Color='Red'>ASP.NET</  
Font>")  
End If  
writer.Write("CustomControl:" + Text)  
End Sub
```

```
C#  
protected override void RenderContents(HtmlTextWriter output)  
{  
    if (Text.Contains("ASP.NET"))  
    {  
        Text.Replace("ASP.NET", "<Font Color='Red'>ASP.NET</Fo  
nt>");  
    }  
    output.Write("CustomControl:" + Text);  
}
```

6. 自主選單「Build」下，選取「Build Solution」，將專案建置產生組件。



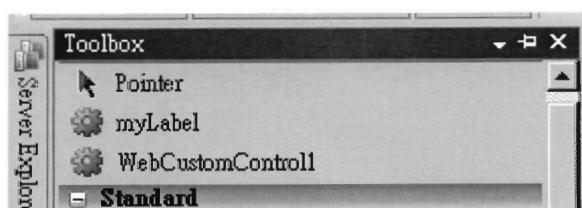


## 在開發工具加入 Web Custom Control

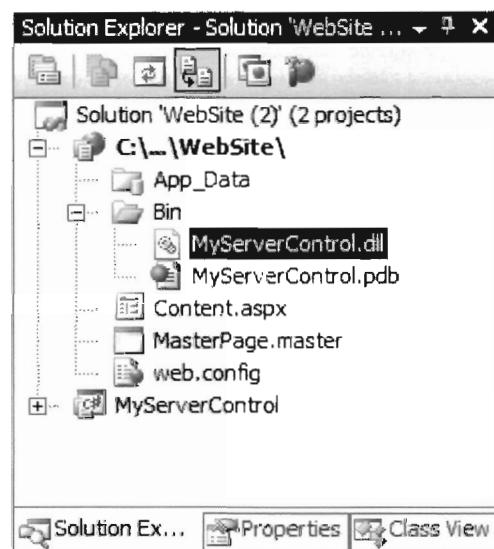
要使用 Web Custom Control 必須在專案中有其參考，因此第一種方式是手動加入控制項的組件參考，加入方式為：

1. 在「Solution Explorer」視窗中找到 Web Application 專案。
2. 按滑鼠右鍵，點選「Add Reference」，在「Add Reference」對話方塊中點選「Browse」，將 ASP.NET Server Control Project 所編譯的組件(\*.dll)加入，例如：  
「myServerControl.dll」
3. 按下「OK」按鈕。

另外，在 ASP.NET 中如果自訂控制項的 ASP.NET Server Control 專案以及 ASP.NET 專案是在同一個 Solution 中，則自訂控制項將會被「自動」的新增到「Toolbox」工具箱中，如下圖所示。



或者開發者也可以手動將所自訂的控制項先註冊到「Toolbox」工具箱中，然後再使用拖拉的方式把控制項拖拉到 Web Page 網頁應用程式中，這樣的做法 Visual Studio 2008 會將 ASP.NET Server Control 組件自動加入參考到專案中。





## 網頁使用 Web Custom Control

欲將開發好的自訂控制項新增到單一 Web Form 網頁中使用，可以使  
用拖拉的方式將控制項從「Toolbox」工具箱中拉進網頁，或者是在  
Web Form 中用`<%@ Register %>`來設定。

例如要將一個名稱為「myServerControl」的自訂控制項放在網頁中使  
用，則可以在網頁中寫入以下程式區段：

```
<%@Register TagPrefix="Ucom" Assembly=" myServerControl"
Namespace=" myServerControl" %>
```

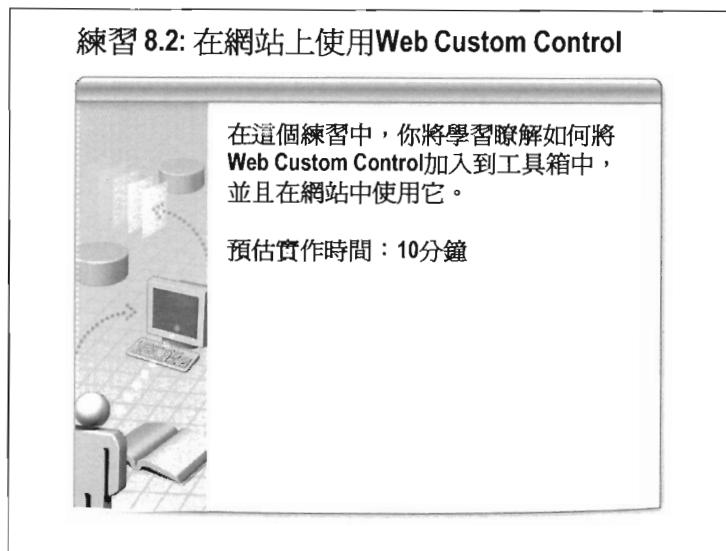
然而，在 ASP.NET 也可以透過在 Web.config 中的控制項設定，讓自  
訂控制項可以在整個專案應用程式中所有的網頁應用程式共用。設定  
的方式就是在 Web.config 中`<pages>`下的`<controls>`加上註冊的設定，  
設定如下：

```
<system.web>
<pages>
<controls>
<add TagPrefix="Ucom" namespace=" myServerControl"
```

```
assembly=" myServerControl" />
</controls>
</pages>
</system.web >
```

當所自訂的控制項在 Web.config 中被註冊了，就可以在整個應用程式所有的網頁中使用 下列語法來使用自訂控制項：

```
<Ucom:myLabel id="myLabel1" runat="server" />
```



## 練習 8.2 : 在網站上使用 Web Custom Control

目的：

在這個練習中，你將學習瞭解 Web Custom Control 加入到工具箱中，並且在網站中使用它。

功能描述：

在這個練習中，會建立一個 ASP.NET 網站放在和自訂控制項同一個 Solution 中，當建置完成，自訂控制項將會自動加入工具箱，以便在網頁中設計使用。

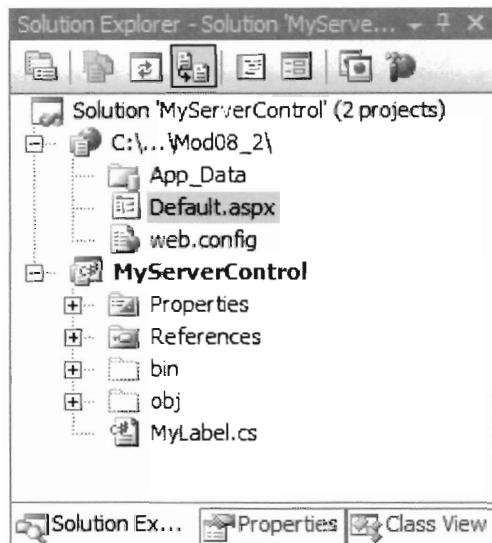
預估實作時間：10 分鐘

實作步驟：

1. 接續上一個練習，或是直接打開檔案總管，開啟「\U9544\Practices\VB 或 CSV\Mod08\_2\Starter\MyServerControl」目錄，使用滑鼠雙擊 MyServerControl.sln，開啟 Solution。
2. 從「File」→「Add」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或

CS\Mod08\_2\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod08\_2」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。



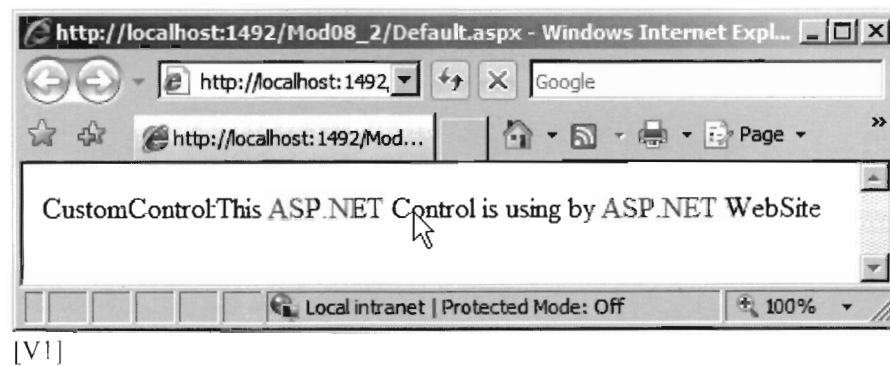
4. 建置整個 Solution，自主選單「Build」下，選取「Build Solution」。
5. 開啓網頁設計視窗，打開工具箱。因為網站專案和自訂控制項專案位在同一個 Solution，因此可以直接在工具箱看到自訂控制項。



6. 將 MyLabel 拖拉到網頁中。切換到 Source 設計視窗，可以檢視註冊控制項的設定資料。

```
<%@ Register assembly="MyServerControl" namespace="MyServerControl" tagprefix="cc1" %>
```

7. 切換回網頁 Design 視窗，設定 MyLabel 控制項的 Text 屬性為：「This ASP.NET Control is using by ASP.NET WebSite」。
8. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。



### 自訂控制項的狀態維護

- 控制項的狀態註冊預設不會保存至下一個要求
- 若要自行保留控制項資料
  - 改寫 SaveControlState 方法
  - 改寫 LoadControlState 方法
  - 使用 Page 物件的 RegisterRequiresControlState 方法

### 自訂控制項的狀態維護

在 ASP.NET 中，控制項在網頁中的執行狀態稱之為 ControlState。網頁在使用者的操作下，會在控制項當中輸入資料，在使用者執行了 Post Back 動作之後，控制項之前的輸入內容會自動保留下來，並不會像以往有些網頁資料會消失。

當我們要在自訂控制項中自行設計所謂的 ControlState 時，就必須改寫自訂控制項的兩個方法，分別是：

- SaveControlState：儲存目前控制項的資料和狀態
- LoadControlState：載入目前控制項的資料和狀態

```
Visual Basic
Protected Overrides Function SaveControlState() As Object
    ...
End Function

Protected Overrides Sub LoadControlState(ByVal savedState As Object)
    ...
End Sub
```

C#

```
protected override object SaveControlState()
{
    ...
}

protected override void LoadControlState(object state)
{
    ...
}
```

## 狀態維護設計

- 針對每次存取必須呼叫 **RegisterRequiresControlState**

建議寫在控制項的 Init 事件中

```
Visual Basic  
Protected Overrides Sub OnInit(ByVal e As System.EventArgs)  
    MyBase.OnInit(e)  
    Page.RegisterRequiresControlState(Me)  
End Sub
```

```
C#  
protected override void OnInit(EventArgs e)  
{  
    base.OnInit(e);  
    Page.RegisterRequiresControlState(this);  
}
```

## 狀態維護設計

當針對自訂控制項的控制項狀態要進行自行維護時，則必須在控制項的 Init 事件中註冊其事件程序，並且改寫事件程序，叫用 Page 物件的 RegisterRequiresControlState 方法才行。

```
Visual Basic  
Protected Overrides Sub OnInit(ByVal e As System.EventArgs)  
    MyBase.OnInit(e)  
    Page.RegisterRequiresControlState(Me)  
End Sub
```

```
C#  
protected override void OnInit(EventArgs e)  
{  
    base.OnInit(e);  
    Page.RegisterRequiresControlState(this);  
}
```

**練習 8.3: 設計維護自訂控制項狀態**



在這個練習中，你將學習瞭解如何設計一個自訂控制項可以擁有本身的狀態維護機制。

預估實作時間：20分鐘

### 練習 8.3 : 設計維護自訂控制項狀態

#### 目的：

在這個練習中，你將學習瞭解如何設計一個自訂控制項可以擁有本身的狀態維護機制。

#### 功能描述：

在這個練習中，在自訂控制項中改寫控制項的 SaveControlState 和 LoadControlState 方法，自行改變控制項狀態機制，並保留某些變數，而不會受到 Page 網頁本身的狀態影響。

#### 預估實作時間：20 分鐘

#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Project」→「Project types」選取「Web」→「templates」選取「ASP.NET Server Control」→點選「Browse...」按鈕將「Location」設為「\U9544\Practices\VB」

或 CS\Mod08\_3\Starter 目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將 Name 取名為「MyServerControl」。

3. 刪除預設的 ServerControl1.cs 或 ServerControl1.vb。
4. 自主選單「Project」下「Add New Item...」，Categorized 選取「Web」→，Templates 選取「ASP.NET Server Control」，並將名稱改為 MyButton→按下「Add」。
5. 修改 MyButton 繼承自 Button 類別，並將 RenderContents 方法註解起來。

```
Visual Basic
Public Class MyButton
    Inherits Button
    ...
End Class
```

```
C#
public class MyButton : Button
{
    ...
}
```

6. 宣告一個數值型別的私有變數 currentIndex，封裝為屬性，並且覆寫 SaveControlState 和 LoadControlState 方法以自行維護狀態，情境為控制項可以自行維護累加變數值，程式碼如下：

```
Visual Basic
Dim currentIndex As Integer = 0
Public Property CurrentCounter() As Integer
    Get
        Return currentIndex
    End Get
    Set(ByVal value As Integer)
        currentIndex = value
    End Set
End Property

Protected Overrides Function SaveControlState() As Object
    If currentIndex = 0 Then
        Return Nothing
    Else
        Return CType(currentIndex, Object)
    End If

```

```

End Function

Protected Overrides Sub LoadControlState(ByVal savedState As
Object)
    If savedState IsNot Nothing Then
        currentIndex = CType(savedState, Integer)
    End If
End Sub

```

```

C#
private int currentIndex = 0;
public int CurrentCounter
{
    get { return currentIndex; }
    set { currentIndex = value; }
}

protected override object SaveControlState()
{
    return currentIndex != 0 ? (object)currentIndex : null;
}

protected override void LoadControlState(object state)
{
    if (state != null)
    {
        currentIndex = (int)state;
    }
}

```

7. 覆寫控制項的 OnInit 事件程序，使用 RegisterRequiresControlState 將控制項的狀態註冊到頁面。

```

Visual Basic
Protected Overrides Sub OnInit(ByVal e As System.EventArgs)
    MyBase.OnInit(e)
    Page.RegisterRequiresControlState(Me)
End Sub

```

```

C#
protected override void OnInit(EventArgs e)
{
    base.OnInit(e);
    Page.RegisterRequiresControlState(this);
}

```

8. 設計完成之後，建置專案。

9. 從「File」→「Add」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod08\_3\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod08\_3」。

10. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。

11. 建置整個 Solution，自主選單「Build」下，選取「Build Solution」。

12. 設計網頁的 EnableViewState 屬性為 false。

```
<%@ Page Language="VB" EnableViewState="false" %>[V2]
```

```
<%@ Page Language="C#" EnableViewState="false" %>[V3]
```

13. 開啓網頁設計視窗，打開工具箱。因為網站專案和自訂控制項專案位在同一個 Solution，因此可以直接在工具箱看到自訂控制項。



14. 將 MyButton 拖拉到網頁中，設定 Text 屬性為「測試控制項 狀態」。切換到 Source 設計視窗，可以檢視註冊控制項的設定資料。

```
Visual Basic
<%@ Page[V4] Language="VB" EnableViewState="false" %>
<%@ Register assembly="MyServerControl" namespace="MyServerControl" tagprefix="cc1" %>
```

```
C#
<%@ Page[VS] Language="C#" EnableViewState="false" %>
<%@ Register assembly="MyServerControl" namespace="MyServerControl" tagprefix="cc1" %>
```

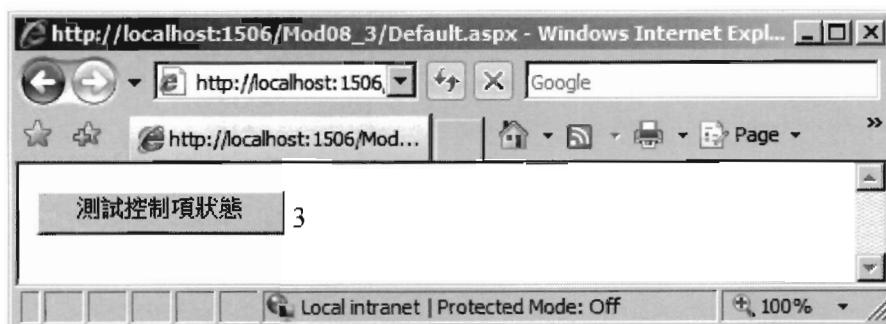
15. 從工具箱中，拉一個 Label 控制項到畫面中。

16. 在網頁的 Page\_Load 事件中，加入存取自訂控制項變數，執行累加的程式碼：

```
Visual Basic
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    MyButton1.CurrentCounter += 1
    Label1.Text = MyButton1.CurrentCounter.ToString()
End Sub
```

```
C#
protected void Page_Load(object sender, EventArgs e)
{
    MyButton1.CurrentCounter++;
    Label1.Text = MyButton1.CurrentCounter.ToString();
}
```

17. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。連著點選按鈕，可以看到控制項的變數值會被累加記錄起來。



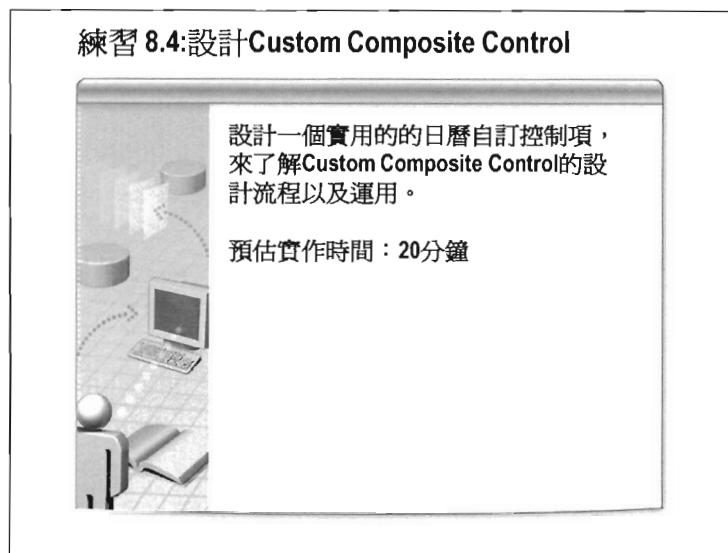
[V6]



## 設計 Custom Composite Control

Composite Control 可將 ASP.NET 中既有的網頁伺服器控制項組合在一起。在 ASP.NET 中有一個類別叫做 `System.Web.UI.WebControls.CompositeControl`，透過繼承此類別可以實作組合式控制項。

Composite Control 與 Custom Control 最大的不同則在於 Composite Control 設計時的所使用的控制項為既有已經存在的 Web Server Control，而利用程式在 `CreateChildControls` 方法中將所有的子控制項產生出來組合起來。



## 練習 8.4 : 設計 Custom Composite Control

目的：

設計一個實用的日曆自訂控制項，來了解 Custom Composite Control 的設計流程以及運用。

功能描述：

在這個練習中，將會在組合控制項中加入一個 TextBox、一個 Button 以及一個 Calendar 控制項，開發一個可以跨不同網站應用程式的日曆自訂控制項。

預估實作時間：20 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Project」→「Project types」選取「Web」→「templates」選取「ASP.NET Server Control」→點選「Browse...」按鈕將「Location」設為「\U9544\Practices\VB」

或 CS\Mod08\_4\Starter 目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將 Name 取名為「MyServerControl」。

3. 刪除預設的 ServerControl1.cs 或 ServerControl1.vb。
4. 自主選單「Project」下「Add New Item...」，Categorized 選取「Web」→，Templates 選取「ASP.NET Server Control」，並將名稱改為 CompositeCalendar → 按下「Add」。
5. 將 CompositeCalendar 原本繼承自「System.Web.UI.WebControls.WebControl」類別改為繼承自「System.Web.UI.WebControls.CompositeControl」類別。

```
Visual Basic
Public Class CompositeCalendar
    Inherits CompositeControl
    ...
End Class
```

```
C#
public class CompositeCalendar : CompositeControl
{
    ...
}
```

6. 將 RenderContents 方法註解起來。
7. 加入覆寫 CreateChildControls 方法，撰寫需要組合的 Child Controls，將既有的伺服器控制項加入：Calendar 控制項、 TextBox 控制項以及 Button 控制項，先將 Calendar、TextBox 以及 Button 控制項宣告為別層級層及變數。
8. 所有子控制項產生的程式碼部份必須在 CreateChildControls 方法裡面去完成。程式碼如下：

```
Visual Basic
Dim txt As TextBox
Dim btn As Button
Dim cal As Calendar
'建立子控制項
Protected Overrides Sub CreateChildControls()
    MyBase.CreateChildControls()
    '建立一個TextBox控制項
    txt = New TextBox()
```

```

txt.ID = "txt"
Me.Controls.Add(txt)

'建立一個Button控制項
btn = New Button()
btn.ID = "btn"
btn.Text = "日期"
Me.Controls.Add(btn)

'建立一個Calendar控制項
cal = New Calendar()
cal.ID = "cal"
cal.Visible = False
Me.Controls.Add(cal)
End Sub

```

```

C#
TextBox txt;
Button btn;
Calendar cal;
protected override void CreateChildControls()
{
    base.RecreateChildControls();
    //建立一個TextBox控制項
    txt = new TextBox();
    txt.ID = "txt";
    this.Controls.Add(txt);

    //建立一個Button控制項
    btn = new Button();
    btn.ID = "btn";
    btn.Text = "日期";
    this.Controls.Add(btn);

    //建立一個Calendar控制項
    cal = new Calendar();
    cal.ID = "cal";
    cal.Visible = false;
    this.Controls.Add(cal);
}

```

- 接著，在日曆上選擇日期後將選到的日期顯示在 TextBox 控制項上。要達到這樣的功能，就必須將剛剛在第五步驟加到自訂控制項中的 Button 控制項以及 Calendar 控制項動態註冊相關的事件程序。程式碼如下。

Visual Basic Protected Sub btnDate_Click(ByVal sender As Object, ByVal e As System.EventArgs) cal.Visible = True End Sub
-----------------------------------------------------------------------------------------------------------------------------------

```
Protected Sub CalDate_SelectionChanged(ByVal sender As Object,
ByVal e As System.EventArgs)
    txt.Text = cal.SelectedDate.ToShortDateString()
    cal.Visible = False
End Sub
```

```
C#
protected void btnDate_Click(object sender, EventArgs e)
{
    cal.Visible = true;
}
protected void CalDate_SelectionChanged(object sender, EventArgs e)
{
    txt.Text = cal.SelectedDate.ToShortDateString();
    cal.Visible = false;
}
```

10. Button 按下去之後會去執行對應的事件程序，Calendar 控制項選到日期之後也會執行對應的事件程序。兩個事件程序分別為「ButtonClick」以及「CalendarSelectionChanged」，透過「AddHandler」(Visual Basic)，如果是 C# 則使用「+=」來對應到 Button 控制項的 Click 事件以及 Calendar 控制項的 SelectionChanged 事件。將動態事件註冊的程式碼分別加入 CreateChildControls 中的 btn.Text = "日期" 以及 cal.Visible = False 這兩行程式碼後。程式碼參考如下。

```
Visual Basic
'在btn.Text = "日期"後加入
AddHandler btn.Click, AddressOf Me.btnDate_Click

'在cal.Visible = False後加入
AddHandler cal.SelectionChanged, AddressOf Me.CalDate_Selectio
```

```
C#
//在btn.Text = "日期"後加入
btn.Click += new EventHandler(this.btnDate_Click);

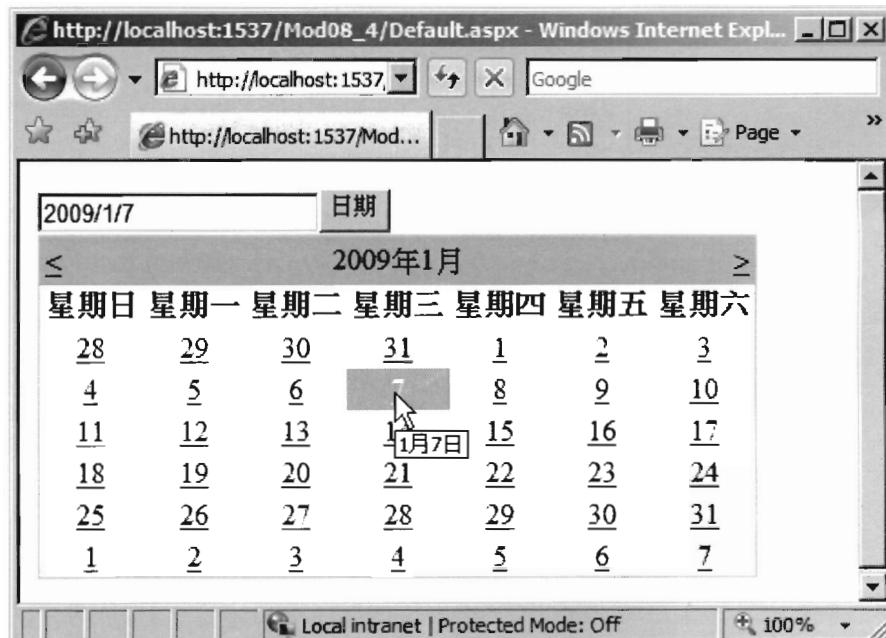
//在cal.Visible = false後加入
cal.SelectionChanged += new EventHandler(this.CalDate_Selectio
nChanged);
```

11. 設計完成之後，建置專案。

12. 從「File」→「Add」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod08\_4\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod08\_4」。
13. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
14. 建置整個 Solution，自主選單「Build」下，選取「Build Solution」。
15. 開啓網頁設計視窗，打開工具箱。將 CompositeCalendar 拖拉到網頁中。切換到 Source 設計視窗，可以檢視註冊控制項的設定資料。

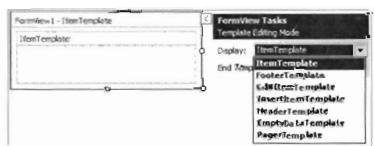
```
<%@ Register assembly="MyServerControl" namespace="MyServerControl" tagprefix="cc1" %>
```

16. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。點選按鈕，可以顯示日曆，點選日曆將會把日期帶到 TextBox 中。



### 支援樣版的Custom Control

- 樣版可以用來客製化控制項的配置
- **ASP.NET**支援兩種樣版：
  - 單向的資料繫結樣版，使用Eval
  - 雙向的資料繫結樣版，使用Bind
- 實作`ITemplate`介面的物件  
包含一個`InstantiateIn`方法建立樣版的內容



## 支援樣版的 Custom Control

樣版(Template)可以用來客製化控制項的配置(Layout)，以便透過一個美觀的外觀呈現在網頁之中。樣板之中能包含一些運算式以便在網頁執行時期動態加以解析。

ASP.NET 支援單向的資料繫結樣版，使用 Eval 運算式顯示資料項目的值，只能以唯讀方式操作，不能修改資料項目中的值。

ASP.NET 提供很多控制項都是支援樣板設計的方式，例如：  
GridView、FormView、DataList……等等，大部分的樣板直接在  
Visual Studio 設計階段就支援視覺化的設計。[V8]



## 如何設計支援樣版的 Custom Control

設計支援樣版自訂控制項的主要步驟為：

1. 繼承 CompositeControl 。
2. 設計一個屬性：回傳實作 ITemplate 介面的物件，並且在屬性上定義 TemplateContainer Attribute 和定義 PersistenceMode Attribute 。
3. 覆寫 CreateChildControls 方法。
4. 用 ItemTemplate 介面的 InstantiateIn 方法建立樣版 。

第 2 步驟中所要設定的兩個 Attribute 定義分別為：

- TemplateContainer Attribute : TemplateContainer attribute 是用來指定包含此樣版的控制項之型別。
- PersistenceMode Attribute : PersistenceMode attribute 指明屬性該如何保存在 ASP.NET 網頁或使用者控制項中。可能的值為 Attribute : EncodedInnerDefaultProperty、InnerDefaultProperty 與 InnerProperty[V9] 。

## 網頁中繫結控制項樣板的資料

- 在網頁中使用繫結的方式取資料

如在 ItemTemplate 中

```
<ItemTemplate>
    <h1> 產品代號 : <%# Container.ProductID %> </h1> <br />
    <h1> 產品名稱 : <%# Container.ProductName %></h1> <br />
    <h1> 單價 : <%# Container.UnitPrice %> </h1>
</ItemTemplate>
```

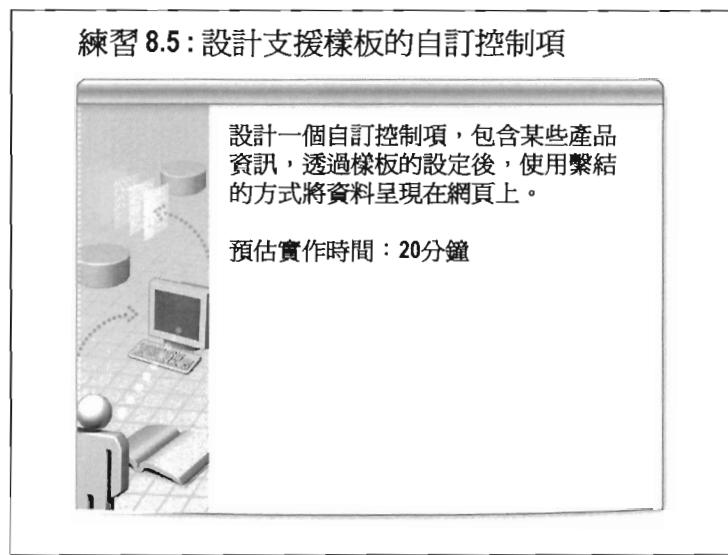
- 在 Web Form 中叫用控制項的 DataBind 方法

## 網頁中繫結控制項樣板的資料

使用<ItemTemplate>標籤之中定義資料展現使用的 HTML 標籤，並利用資料繫結語法來讀取控制項的屬性值來展示。其中 Container 關鍵字代表目前繫結中的容器物件，容器物件通常是實作 INamingContainer 介面的類別。

```
<ItemTemplate>
    <h1> 產品代號 : <%# Container.ProductID %> </h1> <br />
    <h1> 產品名稱 : <%# Container.ProductName %></h1> <br />
    <h1> 單價 : <%# Container.UnitPrice %> </h1>
</ItemTemplate>
```

DataBind 方法是定義在 System.Web.UI.Control 類別中，因 ASP.NET 所有的標準控制項與 HTML 控制項都是繼承自 Control 類別，所以所有控制項都支援 DataBind 方法。



## 練習 8.5 : 設計支援樣板的自訂控制項

### 目的：

在這個練習中，將了解如何設計一個可以支援樣板的自訂控制項，並且在網頁中透過 Template 的設定來對應資料。

### 功能描述：

在這個練習中，將設計一個用來代表產品的自訂控制項，此控制項包含某些產品資訊，透過樣板的設定後，使用繫結的方式將資料呈現在網頁上。

**預估實作時間：20 分鐘**

### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Project」→「Project types」選取「Web」→「templates」選取「ASP.NET Server Control」→點選「Browse...」按鈕將「Location」設為「\U9544\Practices\VB」或「CS\Mod08\_5\Starter」目錄，與使用的程式語言

(Language)，如 Visual Basic 或 C#，將 Name 取名為「MyServerControl」。

3. 刪除預設的 ServerControl1.cs 或 ServerControl1.vb。
4. 自主選單「Project」下「Add New Item...」，Categorized 選取「Web」→，Templates 選取「ASP.NET Server Control」，並將名稱改為 ProductControl →按下「Add」。
5. 將 ProductControl 原本繼承自「System.Web.UI.WebControls.WebControl」類別改為繼承自「System.Web.UI.WebControls.CompositeControl」類別。

```
Visual Basic
Public Class ProductControl
    Inherits CompositeControl
    ...
End Class
```

```
C#
public class ProductControl: CompositeControl
{
    ...
}
```

6. 將 RenderContents 方法註解起來。
7. ProductControl 控制項包含 ProductID、ProductName、UnitPrice 三個屬性記錄產品資料，另外 ProductControl 控制項包含一個屬性，名稱為 ItemTemplate，它回傳一個實作 ITemplate 介面的物件，程式碼如下：

```
Visual Basic
Dim _itemTemplate As ITemplate
Public ProductID As String
Public ProductName As String
Public UnitPrice As Decimal

Public Property ItemTemplate() As ITemplate
    Get
        Return _itemTemplate
    End Get
    Set(ByVal value As ITemplate)
        _itemTemplate = value
    End Set
```

End Property

```
C#
private ITemplate _itemTemplate;
public string ProductID { get; set; }
public string ProductName { get; set; }
public decimal UnitPrice { get; set; }

public ITemplate ItemTemplate
{
    get { return _itemTemplate; }
    set { _itemTemplate = value; }
}
```

8. 針對 ItemTemplate 屬性加上 TemplateContainer 和 PersistenceMode Attribute 設定：

Visual Basic

```
<TemplateContainer(GetType(ProductControl))> _
<PersistenceMode(PersistenceMode.InnerProperty)> _
Public Property ItemTemplate() As ITemplate
    Get
        Return _itemTemplate
    End Get
    Set(ByVal value As ITemplate)
        _itemTemplate = value
    End Set
End Property
```

```
C#
[TemplateContainer(typeof(ProductControl))]
[PersistenceMode(PersistenceMode.InnerProperty)]
public ITemplate ItemTemplate
{
    get { return _itemTemplate; }
    set { _itemTemplate = value; }
}
protected override void CreateChildControls()
{
    _itemTemplate.InstantiateIn(this);
}
```

9. 改寫 CreateChildControls 方法，在此方法中叫用 ItemTemplate 介面的 InstantiateIn 方法建立樣版，ItemTemplate 成為 ProductControl 控制項的子控制項：

Visual Basic Protected Overrides Sub CreateChildControls()
---------------------------------------------------------------

```
    itemTemplate.InstantiateIn(Me)
End Sub
```

```
C#
protected override void CreateChildControls()
{
    _itemTemplate.InstantiateIn(this);
}
```

10. 設計完成之後，建置專案。

11. 從「File」→「Add」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod08\_5\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod08\_5」。

12. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。

13. 建置整個 Solution，自主選單「Build」下，選取「Build Solution」。

14. 開啓網頁設計視窗，打開工具箱，將 ProductControl 拖拉到網頁中。

15. 切換到 Source 設計視窗，針對 ProductControl1 加上 ItemTemplate 設定。

```
<cc1:ProductControl ID="ProductControl1" runat="server">
<ItemTemplate>
    <h1> 產品代號：<%# Container.ProductID %> </h1><br />
    <h1> 產品名稱：<%# Container.ProductName %></h1><br />
    <h1> 單價：<%# Container.UnitPrice %> <h1>
</ItemTemplate>
</cc1:ProductControl>
```

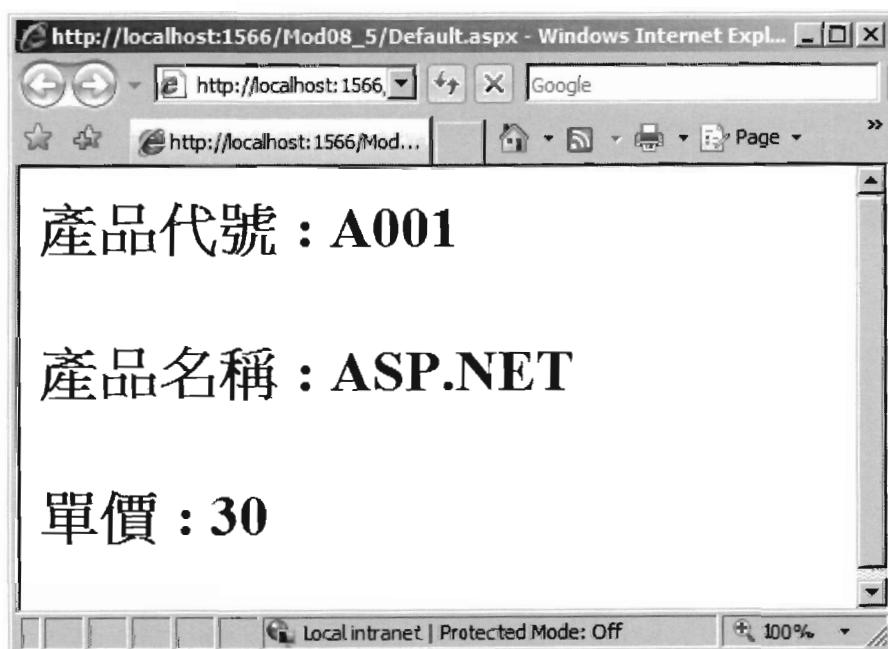
16. 在網頁的 Page\_Load 事件中，加入資料繫結程式碼：

```
Visual Basic
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    MyButton1.CurrentCounter += 1
    Label1.Text = MyButton1.CurrentCounter.ToString()
```

```
End Sub
```

```
C#
protected void Page_Load(object sender, EventArgs e)
{
    ProductControl1.ProductID = "A001";
    ProductControl1.ProductName = "ASP.NET";
    ProductControl1.UnitPrice=30;
    ProductControl1.DataBind();
}
```

17. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。



## 總結

- 什麼是 Web Custom Control
- 設計 ASP.NET Server Control
- 使用 ASP.NET Server Control
- 自訂控制項的狀態維護
- 設計 Composite Web Server Control
- 自訂控制項的樣版設計

# 第九章：效能調校與網站監控

## 本章大綱

ASP.NET 網頁快取架構.....	4
網頁輸出快取 .....	5
建立 OutputCache 組態檔 .....	7
使用程式操作快取 .....	9
練習 9.1 : 使用網頁輸出快取 .....	11
網頁區塊快取 .....	14
更新快取網頁的部分內容 .....	15
利用 API 或程式取代網頁快取內容 .....	17
資料快取 .....	19
練習 9.2 : 使用資料快取 .....	21
Cache 逾期與相依性設定 .....	24
使用 SQL Server 快取相依性 .....	26
練習 9.3 : 使用 SqlCacheDependency .....	28
使用健康監視 (Health Monitoring) .....	31
健康監視運作方式 .....	32
Web 事件與提供者 .....	33
啓用健康監視功能 .....	35
練習 9.4 : 使用健康監視功能監控網站活動 .....	37

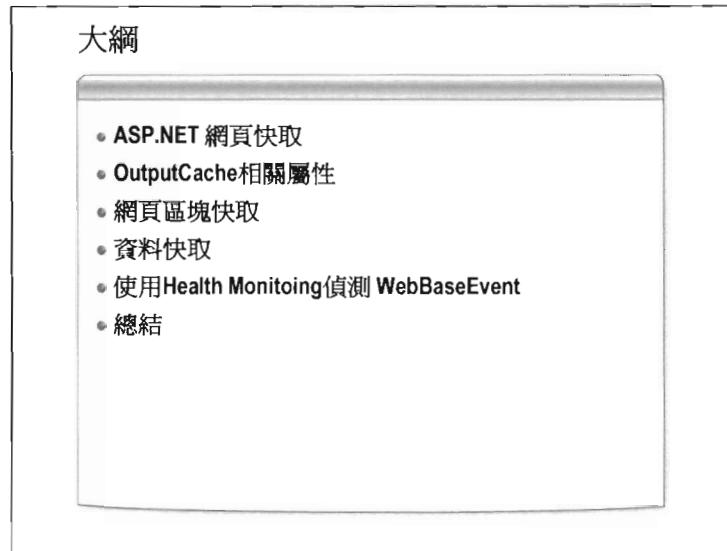
作者：

許薰尹





---

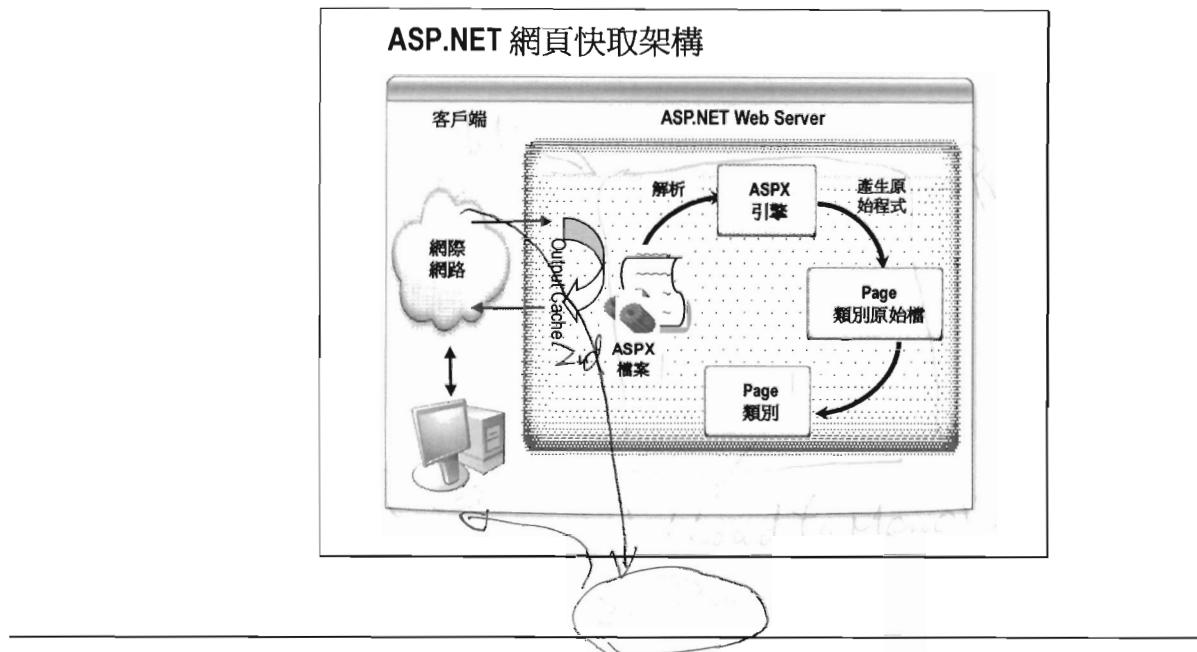


---

使用 ASP.NET 快取的技術可以讓網站效能大大提升。健康監視 (Health Monitoring)，能夠讓系統管理者監控 ASP.NET 網站應用程式的運作情況。健康監視的主要目地在於讓 ASP.NET 應用程式能夠順利地運作，並在網站執行發生問題時，能夠快速地找出問題所在，以及及時通知網站管理者。

在這個章節中將學習到：

- 網頁快取的技巧
- OutputCache 相關屬性
- 網頁區塊快取
- 資料快取
- 啓用 SQL Server 資料庫快取通知功能
- SqlDependency 快取設計
- 使用 Health Monitoring



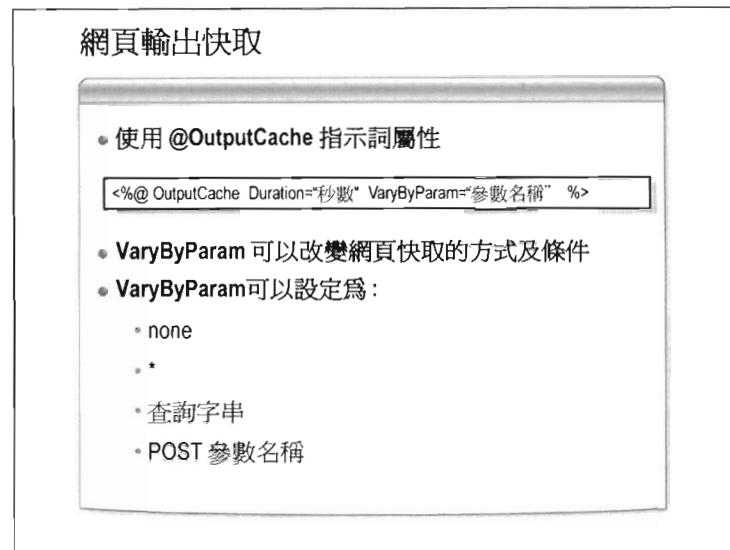
## ASP.NET 網頁快取架構

ASP.NET 提供快取機制，以讓您建立高效能的 Web 應用程式。

OutputCache(輸出快取)可將網頁和使用者控制項 (User Control) 回應儲存至 Cache，下次要執行相同的網頁或使用者控制項時，並不會再次執行網頁或使用者控制項程式碼，而是使用快取輸出來回應要求。利用 OutputCache 快取機制將首頁的網頁結果存起來，就能夠大量降低網站伺服器的負載，也能提升效能，簡單來說，OutputCache 快取機制就是將網頁執行結果暫存到記憶體。

例如當使用者要求網頁程式執行時，網頁會遵循圖中的流程執行並產生 Html 資料，在把網頁執行結果回傳給使用者之前，如果有設定快取機制，網頁執行結果將會存一份在網站的 OutputCache 區(網站伺服器的記憶體)。

當其他使用者要求同樣一張網頁程式執行時，ASP.NET 執行環境會檢查 OutputCache 區是否有已存在的網頁結果資料，如果有就會直接讓使用者下載網頁結果；如果沒有，則仍然遵循一般流程執行網頁程式來產生網頁結果。



## 網頁輸出快取

網頁輸出快取適用於網頁內容不常變更，但是需要大量處理時間才能建立的網頁。網頁輸出快取提供兩種模型：完整網頁快取和部分網頁快取 (Partial Page Caching)。完整網頁快取將網頁的全部內容存放在記憶體中。部分網頁快取可以讓您只選擇快取網頁的一小部分，其他部分則不快取而會動態執行。

我們先來探討完整網頁快取。要將網頁執行的結果轉變成輸出快取有兩種作法：一是使用 `OutputCache` 指示詞；二是利用程式碼來設定。

利用網頁的`@OutputCache` 指示詞的相關屬性設定，能輕易讓網頁具備快取功能。`@OutputCache` 指示詞中的必要屬性 `VaryParam`，藉由設定此屬性可以改變網頁快取的方式及條件，另一個必要屬性 `Duration` 則是用來設定快取存在時間，單位為秒。使用 `@OutputCache` 指示詞設定網頁的快取性必須宣告 `Duration` 屬性，以及 `VaryByControl` 屬性或 `VaryByParam` 屬性和 `Location` 屬性。`Duration` 要設成大於零的值。若不想要使用 `VaryByParam` 或 `VaryByControl` 的功能，可以直接將 `VaryByParam` 屬性設定為「None」。

在網頁\*.aspx 中加入下列設定，網頁執行後所產生的 Html 結果將保存在記憶體中 5 秒，在 5 秒後會自動清除，其中 Duration 為必要屬性，VaryByParam 設為 none 代表沒有設定快取的條件：

```
<%@ OutputCache Duration="5" VaryByParam="none" %>
```

以下利用 VaryByParam 設定快取條件，當參數 EmployeeID 的資料內容不同時便產生新的 Cache

```
<%@ OutputCache Duration="5"
    VaryByParam="EmployeeID" %>
```

**建立OutputCache組態檔**

- 在不同的網頁之中，套用相同的Cache設定
- 組態檔案

```
<system.web>
  <caching>
    <outputCacheSettings>
      <outputCacheProfiles>
        <add name="mycache" duration="20" varyByParam="none" />
      </outputCacheProfiles>
    </outputCacheSettings>
  </caching>
</system.web>
```

在網頁中套用

```
<%@ OutputCache CacheProfile=" mycache " %>
```

## 建立 OutputCache 組態檔

若網站中有許多的網頁都需要設定 OutputCache，而 OutputCache 的屬性又都相同，您可以在組態檔案之中建立 Cache 設定檔(Cache Profile)，這樣就可以在不同的網頁之中，套用相同的 Cache 設定。

以下範例在組態檔案中，設定名為 mycache 的設定檔，逾期時間設定為 20 秒：

```
<system.web>
  <caching>
    <outputCacheSettings>
      <outputCacheProfiles>
        <add name="mycache"
          duration="20" varyByParam="none"
        />
      </outputCacheProfiles>
    </outputCacheSettings>
  </caching>
</system.web>
```

網頁中如果要套用這個設定，可以在 OutputCache 指示詞加上以下設定：

```
<%@ OutputCache CacheProfile=" mycache " %>
```

outputCacheProfiles 也可以搭配 varyByParam、varyByControl、varyByHeader，或 varyByCustom 一起使用。

### 使用程式操作快取

- 較@OutputCache 指示詞更能精確控制快取

- 使用 **HttpCachePolicy** 類別

Visual Basic

```
Response.Cache.SetCacheability(HttpCacheability.Public)
Response.Cache.SetExpires(DateTime.Now.AddSeconds(10))
Response.Cache.SetValidUntilExpires(true)
```

C#

```
Response.Cache.SetCacheability(HttpCacheability.Public);
Response.Cache.SetExpires(DateTime.Now.AddSeconds(10));
Response.Cache.SetValidUntilExpires(true);
```

- **HttpCacheability** 列舉型別

設定 Cache-Control 標頭以指定 Cache 位置

### 使用程式操作快取

如果在設計階段就能決定網頁的快取的設定需求，可以直接用宣告方式設定快取性，或是在程式執行時期才能決定，或需要更仔細地控制 Cache 的運作，您可以直接使用 **HttpCachePolicy** 類別，只要透過 Response 物件的 Cache 屬性就可以取得它。

**HttpCachePolicy** 類別提供許多屬性與方法，能夠透過程式碼來設定 Cache，這樣就不必使用<%@ OutputCache %> 指示詞來設定 Cache。此類別常用的方法包含：

- **AddValidationCallback**：指定一個方法，以便在從 Cache 中取得網頁之前先執行。
- **AppendCacheExtension**：可用來自訂文字到 HTTP 標頭。
- **SetAllowResponseInBrowserHistory**：避免網頁出現在歷史清單。
- **SetCacheability**：設定 Cache。
- **SetExpires**：設定快取的絕對逾期時間。傳入逾期時間 (DateTime)。
- **SetSlidingExpiration**：設定相對逾期。

- SetValidUnitExpires：是否忽略用戶端傳來的無效之 Cache-Control 標頭。若傳入 True 代表忽略。使用@ OutputCache 指示詞，會自動忽略。

HttpCachePolicy 類別 SetCacheability 方法設定 HTTP 標頭 Cache-Control 的值。傳入 HttpCacheability 列舉型別的值，列舉項目如下：

- NoCache：設定標頭為 Cache-Control:no-cache。
- Private：預設值，設定標頭為 Cache-Control:private。指定 Response 快取只能在用戶端。
- Public：設定標頭為 Cache-Control: public。指定 Response 快取在用戶端及 Proxy。
- Server：設定標頭為 Cache-Control:no-cache，指定 Response 快取在原始伺服器。
- ServerAndNoCache：同時套用 Server 和 NoCache 兩種設定。
- ServerAndPrivate：指定 Response 快取只能在伺服器及用戶端。

以下範例設定使用 OutputCache，將 Cache 的逾期時間設為 10 秒：

```
Visual Basic
Response.Cache.SetCacheability(HttpCacheability.Public)
Response.Cache.SetExpires(DateTime.Now.AddSeconds(10))
Response.Cache.SetValidUntilExpires(true)
```

```
C#
Response.Cache.SetCacheability(HttpCacheability.Public);
Response.Cache.SetExpires(DateTime.Now.AddSeconds(10));
Response.Cache.SetValidUntilExpires(true);
```

### 練習9.1：使用網頁輸出快取

在練習中，將利用程式方式來產生輸出快取，以及透過CacheProfile設定，讓多個網頁共用快取設定。

•預估實作時間：10分鐘

### 練習 9.1：使用網頁輸出快取

#### 目的：

在這個練習中將熟悉網頁輸出快取的應用。

#### 功能描述：

在練習中，將利用程式方式來產生輸出快取，以及透過 CacheProfile 設定，讓多個網頁共用快取設定。

預估實作時間：10 分鐘

#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「U9544\Practices\VB 或 CS\Mod09\_1\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod09\_1」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。

4. 從「Toolbox」工具箱中拖拉一個 Label 控制項到網頁中。

5. 在網頁 Page\_Load 事件處理常式，加入以下程式：

Visual Basic

```
Label1.Text = "Page :" + System.DateTime.Now.ToString()
```

C#

```
Label1.Text = "Page :" + System.DateTime.Now.ToString();
```

6. 在 Web.config 檔案<system.web>標籤下方加入以下 outputCacheSettings 設定，新增一個名為 myprofile 的設定，逾期時間為 10 秒：

```
<caching>
  <outputCacheSettings>
    <outputCacheProfiles>
      <add name="myprofile"
           duration="10" varyByParam="None"
           />
    </outputCacheProfiles>
  </outputCacheSettings>
</caching>
```

7. 切換到網頁的「Source」畫面。在 ASPX 網頁最上方加入以下設定：

```
<%@ OutputCache CacheProfile="myprofile" %>
```

8. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」，確認程式執行結果。網頁執行時，會印出執行時間，保留 10 秒不變動。

9. 移除步驟 8 的 CahceProfile 設定。

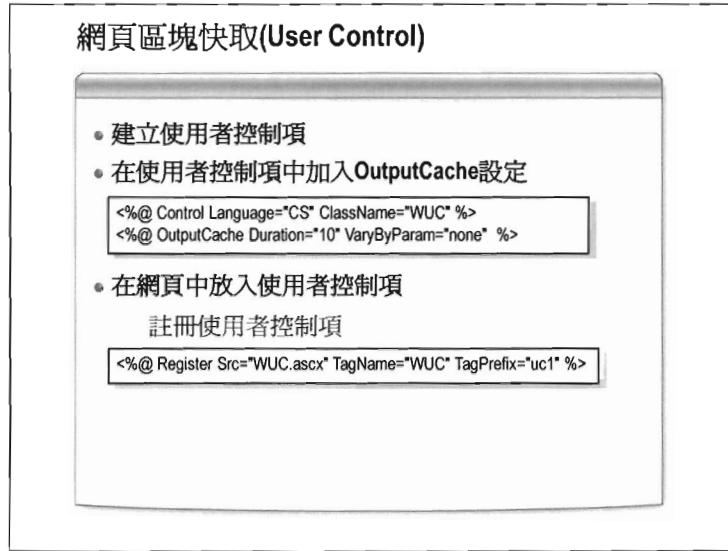
10. 在網頁 Page\_Load 事件處理常式最後加入以下程式：

Visual Basic

```
Response.Cache.SetCacheability(HttpCacheability.Public)
Response.Cache.SetExpires(DateTime.Now.AddSeconds(10))
Response.Cache.SetValidUntilExpires(true)
```

```
C#
Response.Cache.SetCacheability(HttpCacheability.Public);
Response.Cache.SetExpires(DateTime.Now.AddSeconds(10));
Response.Cache.SetValidUntilExpires(true);
```

執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」，確認程式執行結果。網頁執行時，會印出執行時間，保留 10 秒不變動。



## 網頁區塊快取

網頁區塊快取(也稱部分網頁快取)的運作方式有兩種：使用使用者控制項快取和快取後置替換 (Post-Cache Substitution)。

ASP.NET 提供了網頁區塊快取的功能，但是要搭配使用者控制項 (User Control)，在使用者控制項可以加入 OutputCache 設定，就能讓使用者控制項能夠在記憶體中產生快取，將來加入到網頁當中，只有使用者控制項這個部份有快取功能，而不是整張網頁都快取，這就是所謂的網頁區塊快取。

網頁區塊快取實作的方法如下：

- 建立使用者控制項
- 在使用者控制項中加入 OutputCache 設定
 

```
<%@ OutputCache Duration="10" VaryByParam="none" %>
```
- 在網頁中放入使用者控制項

### 更新快取網頁的部分內容

- 網頁有快取功能，部分區塊即時執行
- 撰寫靜態方法

```
Visual Basic
Shared Function GetCurrentDate(ByVal context As HttpContext) As String
    Return DateTime.Now.ToString()
End Function
```

```
C#
static string GetCurrentDate (HttpContext context)
{
    return DateTime.Now.ToString();
}
```

### 更新快取網頁的部分內容

若整張網頁都希望有快取功能，但是只有部分區塊希望能夠即時執行，取得最新的執行結果，可以選擇動態更新快取網頁的部分內容，稱為快取後置替換(Post-Cache Substitution)。

例如一般網站的首頁大都不常變動，所以可以利用 OutputCache 的設定提升效能，但是首頁中的現在時間卻是經常變動的一個資料區塊，所以除了提昇效能之後也必須考慮類似這種情形的資料區塊。

Post-Cache Substitution 實現方式包含：

- 使用 API
- 使用 Substitution 控制項
- 使用 AdRotator 控制項 (隱含式)

您可以在網頁加入靜態方法，撰寫產生即時資料的程式。以下範例程式回傳最新時間：

```
Visual Basic
Shared Function GetCurrentDate(ByVal context As HttpContext) As
```

```
String  
    Return DateTime.Now.ToString()  
End Function
```

```
C#  
static string GetCurrentDate (HttpContext context)  
{  
    return DateTime.Now.ToString();  
}
```

## 利用 API 或程式取代網頁快取內容

- 使用 `Response.WriteSubstitution` 方法

```
Visual Basic
<% Response.WriteSubstitution(New
HttpResponseSubstitutionCallback(AddressOf GetCurrentDate)) %>
```

```
C#
<% Response.WriteSubstitution(new
HttpResponseSubstitutionCallback(GetCurrentDate)) %>
```

- Substitution** 控制項

```
<asp:Substitution ID="Substitution1"
    MethodName="GetCurrentDate" runat="server" />
```

## 利用 API 或程式取代網頁快取內容

以下範例說明如何在網頁中呼叫靜態方法取代網頁快取內容：

```
Visual Basic
<% Response.WriteSubstitution(New HttpResponseSubstitutionCal
lback(AddressOf GetCurrentDate)) %>
```

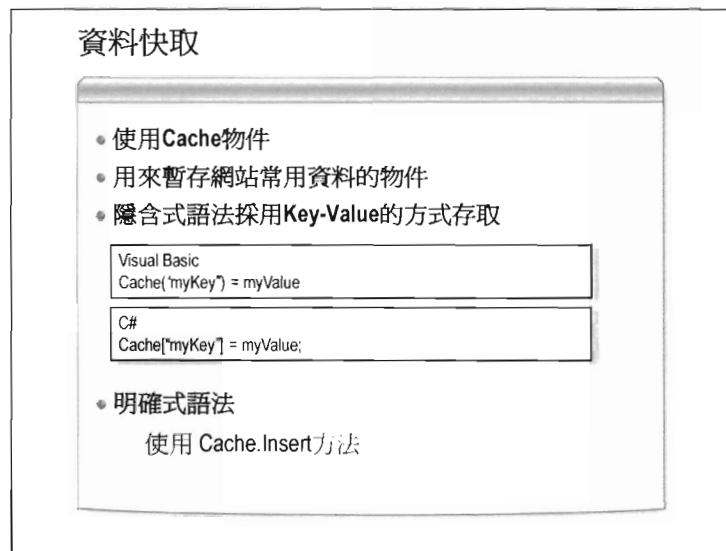
```
C#
<% Response.WriteSubstitution(new HttpResponseSubstitutionCal
lback(GetCurrentDate)) %>
```

使用 `Response.WriteSubstitution` 方法呼叫靜態方法以便即時取得最新資料，在 `WriteSubstitution` 方法的參數中傳入 `HttpResponseSubstitutionCallback` 型別委派。也就是只要將靜態方法的記憶體位置傳入 `WriteSubstitution` 方法，就能完成即時呼叫。

## 使用 Substitution 控制項

您可以在網頁中加入一個 `Substitution` 控制項。另一種方法可以藉由 `Substitution` 控制項 `MethodName` 屬性直接指定靜態方法名稱，也能做到即時呼叫。

```
<asp:Substitution ID="Substitution1"
    MethodName="GetCurrentDate" runat="server" />
```



## 資料快取

資料快取的功能可以大大降低後端伺服器(例如資料庫)的負擔，網頁程式經常需要大量存取後端伺服器的資源，以資料庫為例，如果能將資料暫時存放在網站的記憶體中，當使用者要求的時候就能直接取用，不須重複對後端資料庫存取。

### **Cache** 物件用來暫存網站常用資料

- 類似 Application 物件用來存放應用程式相關資料
- 並非類似 Session 物件用來存放使用者相關資料

雖然它類似 Application 物件的性質，但是提供較大的彈性，另外，Application 物件的使用有個較大的缺點，那就是就算沒有使用到也會一直佔用記憶體，除非有程式去清除，或是網站應用程式重新啟動才會清除掉。

而對於某些資料，我們期望如果這樣的資料沒有使用到就能先釋出記憶體，或者是固定一段時間就清除，又或者是能固定更新快取資料，

這些需求就可以使用 Cache 物件。基本語法可以使用隱含式宣告法如下：

```
Visual Basic  
Cache("myKey") = myValue
```

```
C#  
Cache["myKey"] = myValue;
```

但是這樣的宣告法則釋出記憶體的時機便會由 GC 控制了。

### 練習9.2：使用資料快取

•在練習中，將利用程式讀取資料庫的資料，將資料存放到DataSet，然後將DataSet放到快取中，下回執行網頁，當快取未過期時，從快取中取出資料顯示在網頁GridView控制項上。

•預估實作時間：12分鐘

### 練習 9.2：使用資料快取

#### 目的：

在這個練習中將熟悉網頁資料快取 Cache 物件的應用。

#### 功能描述：

在練習中，將利用程式讀取資料庫的資料，將資料存放到 DataSet，然後將 DataSet 放到快取中，下回執行網頁，當快取未過期時，從快取中取出資料顯示在網頁 GridView 控制項上。

**預估實作時間：12分鐘**

#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「U9544\Practices\VB 或 CS\Mod09\_2\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod09\_2」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 從「Toolbox」工具箱中拖拉一個 Label、GridView 標控項到網頁中。
5. 切換到網頁的「Source」畫面。在 ASPX 網頁最上方匯入以下命名空間：

```
<%@ import Namespace="System.Data" %>
<%@ import Namespace="System.Data.SqlClient" %>
```

6. 加入以下程式碼，設計一個 GetCacheData 方法，若 Cache 中無資料，便利用 ADO.NET 物件將資料查詢出，然後放到 Cache 物件中後回傳，並印出查詢時間：

```
Visual Basic
Public Function GetCacheData() As Object
    If Cache("Region") Is Nothing Then
        Dim cn As New SqlConnection("Data Source=.;Initial Catalog=Northwind;integrated Security=sspi;")
        Dim cmd As New SqlCommand("Select * from Region", cn)
        Dim da As New SqlDataAdapter(cmd)
        Dim ds As New DataSet()
        da.Fill(ds)
        Cache.Insert("Region", ds)
        Label1.Text = "From DB :" + System.DateTime.Now.ToString()
    Else
        Label1.Text = "From Cache!"
    End If
    Return Cache("Region")
End Function
```

```
C#
public object GetCacheData()
{
    if (Cache["Region"] == null)
    {
        SqlConnection cn = new SqlConnection("Data Source=.;Initial Catalog=Northwind;integrated Security=sspi;");
        SqlCommand cmd = new SqlCommand("Select * from Re
```

```

gion", cn);
SqlDataAdapter da = new SqlDataAdapter(cmd);
DataSet ds = new DataSet();
da.Fill(ds);
Cache.Insert("Region", ds);
Label1.Text = "From DB :" + System.DateTime.Now.ToString();
}
else
{
    Label1.Text = "From Cache!";
}
return Cache["Region"];
}

```

7. 在 Page\_Load 事件處理常式，加入以下程式：

Visual Basic

```

Dim c As DataSet = CType(GetCacheData(), DataSet)
GridView1.DataSource = c
GridView1.DataBind()

```

C#

```

DataSet c = (DataSet)GetCacheData();
GridView1.DataSource = c;
GridView1.DataBind();

```

8. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」，確認程式執行結果。網頁第一次執行時，會印出執行時間，若重新整理則印出「From Cache!」字串。

### Cache 逾期與相依性設定

- 使用 Cache.Insert 方法可進一步設定 Cache 物件的逾期時間與相依性
- 逾期條件可以是：
  - 絕對過期時間
  - 相對過期時間
  - 檔案相依性 (使用 CacheDependency 物件)
  - SQL Server 相依性 (使用 CacheDependency 物件)

## Cache 逾期與相依性設定

若要設定 Cache 逾期時間、移除通知、到期原則、定義快取項目的相對優先權與相依性，可以利用 Cache 物件的 Insert 方法，這樣就能讓快取更具彈性。若使用 Insert 方法將資料加入快取，但快取中已有相同 key 值的資料，則新加入的會取代快取中原有的資料。

### 設定 Cache 物件的時效性

以下範例設定快取資料在寫入 5 秒後過期：

Visual Basic

```
Cache.Insert(myKey, myValue, Nothing,
            DateTime.Now.AddSeconds(5),
            System.Web.Caching.Cache.NoSlidingExpiration)
```

C#

```
Cache.Insert(myKey, myValue, null,
            DateTime.Now.AddSeconds(5),
            System.Web.Caching.Cache.NoSlidingExpiration);
```

使用 Cache 物件的 Insert 方法傳入欲快取物件的 Key 及 Value，並且指定絕對過期時間為當時時間以後 5 秒，最後一個參數為相對過期時

間設定為 System.Web.Caching.Cache.NoSlidingExpiration，代表沒有指定。

### 設定 Cache 物件的相對過期時間

以下範例設定快取資料相對於存取時間 5 秒後過期：

Visual Basic

```
Cache.Insert(myKey, myValue, Nothing,
    System.Web.Caching.Cache.NoAbsoluteExpiration,
    TimeSpan.FromSeconds(5))
```

C#

```
Cache.Insert(myKey, myValue, null,
    System.Web.Caching.Cache.NoAbsoluteExpiration,
    TimeSpan.FromSeconds(5));
```

使用 Cache 物件的 Insert 方法傳入欲快取物件的 Key 及 Value，絕對過期時間的參數設定為

System.Web.Caching.Cache.NoAbsoluteExpiration，代表沒有指定。最後一個參數則設定相對過期時間為 5 秒，也就是如果快取資料一直被使用中，會一直將過期時間往後延 5 秒；如果 5 秒內沒有任何存取動作就會過期。

### 設定 Cache 物件的檔案相依性

當快取資料的來源是某個檔案時，如果檔案被修改時，為了防止快取資料與實際檔案內容不一致，可以設定快取物件的相依性，當實際資料檔案內容被修改時，立即讓快取資料過期。

以下範例設定相依檔案為實體路徑下的 Books.xml：

Visual Basic

```
Cache.Insert(myKey, myValue,
    New System.Web.Caching.CacheDependency(Server.MapPath("Bo
oks.xml")))
```

C#

```
Cache.Insert(myKey, myValue,
    new System.Web.Caching.CacheDependency(Server.MapPath("Bo
oks.xml")));
```

### 使用 SQL Server 快取相依性

- 資料表內容在發生異動時主動將快取移除
- 支援資料庫是SQL Server 7.0或2000以上版本
- 不支援其他類型的資料庫
- 使用輪詢模式的步驟
  1. 啓用SQL Server快取相依性
  2. 在Web.Config啟用ASP.NET 資料庫相依性
  3. 在網頁中使用SQL快取相依性

## 使用 SQL Server 快取相依性

SQLCacheDependency 可以針對 SQL Server 7.0 以上版本的資料表內容在發生異動時主動將快取移除。若資料庫是 SQL Server 7.0 或 2000 以上版本的，您可以採用輪詢（Polling）模式設計快取相依性。SQL Server 6.5 之前的版本或者其他類型的資料庫則不支援。

輪詢模式是由 ASP.NET 應用程式間隔一段時間透過指定的連線到資料庫去詢問某個資料表是否有更新版本。資料庫若要支援輪詢模式必須啓用快取相依性。

使用輪詢模式的步驟如下：

- 啓用 SQL Server 快取相依性。利用 AspNet\_Regsql.exe 啓用相依性：

```
AspNet_Regsql -S 伺服器名稱 -E -d 資料庫 -ed -t 資料表 -et
```

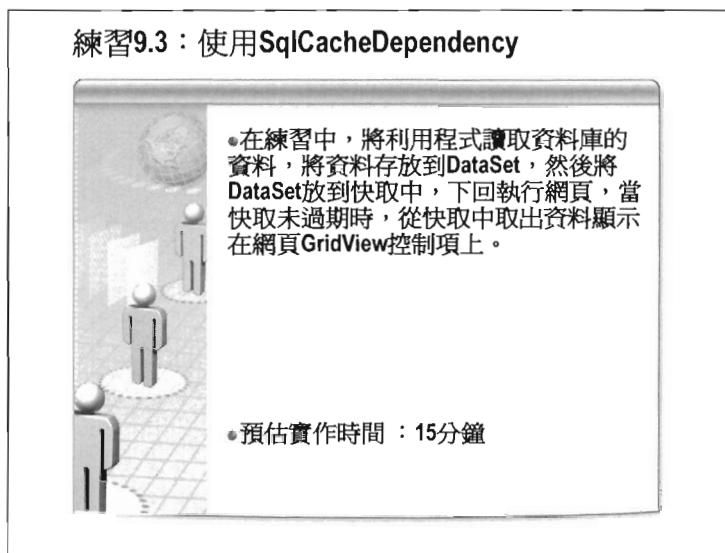
- 在 Web.Config 啓用 ASP.NET 資料庫相依性。

```
<caching>
<sqlCacheDependency enabled="true" pollTime="1000">
<databases>
<add connectionStringName="NorthwindConnectionString1"
```

```
" name="Northwind" pollTime="1000"/>
  </databases>
  </sqlCacheDependency>
</caching>
```

- 在網頁中使用 SQL 快取相依性

```
<%@ OutputCache Duration="秒數" VaryByParam="參數清單"
SqlDependency = "資料庫快取相依性名稱:資料表" %>
```



## 練習 9.3：使用 SqlCacheDependency

目的：

在這個練習中，將了解 SqlCacheDependency 物件快取的使用方式。

功能描述：

在練習中，將利用程式讀取資料庫的資料，將資料存放到 DataSet，然後將 DataSet 放到快取中，下回執行網頁，當快取未過期時，從快取中取出資料顯示在網頁 GridView 控制項上。

預估實作時間：15 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod09\_3\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod09\_3」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 從「Toolbox」工具箱中拖拉一個 Label 控制項到網頁中。
5. 從「Server Explorer」視窗將 Northwind 連線下 Employees 資料表拖拉到網頁設計畫面。網頁會自動產生一個 GridView 與 SqlDataSource 控制項。
6. 在 Page\_Load 事件處理常式，加入以下程式：

Visual Basic

```
Label1.Text = System.DateTime.Now.ToString()
```

C#

```
Label1.Text = System.DateTime.Now.ToString();
```

7. 啓動 Northwind 資料庫的 Employees 資料表快取通知功能。從「開始」→「程式集」→「Microsoft Visual Studio 2005」→「Visual Studio Tools」→「Visual Studio Command Prompt」，執行下列指令：

```
Aspnet_regsql -S (local) -E -ed -d Northwind -et -t Employees
```

8. 開啓 SQL Server Management Studio，新增一個查詢，輸入以下指令後執行：

```
ALTER DATABASE Northwind SET ENABLE_BROKER;
```

9. 在 Web.config 檔案的<system.web>區段內加入使用 SqlDependency 功能的設定，如下：

```
<caching>
  <sqlCacheDependency enabled="true" pollTime="1000">
    <databases>
      <add
        connectionStringName="NorthwindConnectionString1"
        name="Northwind" pollTime="1000"/>
    </databases>
  </sqlCacheDependency>
</caching>
```

10. 在網頁上進行 SQL 快取相依性的設定：

```
<%@ OutputCache Duration="3600" VaryByParam="None" SqlDependency = "Northwind:Employees" %>
```

11. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」，確認程式執行結果。當資料庫的資料沒有異動時，網頁執行結果將存放在 Cache，時間不會變動。

### 使用健康監視 (Health Monitoring)

- ASP.NET 健康監視可讓系統管理員監視已部署之 Web 應用程式的狀態

監視應用程式的效能以確保它是健康的

快速診斷失敗的應用程式或系統

評估指定之應用程式存留週期間的事件

- 主動式的監控

- 監控到的資訊可以記錄在事件日誌、資料庫，或透過電子郵件傳送至管理者

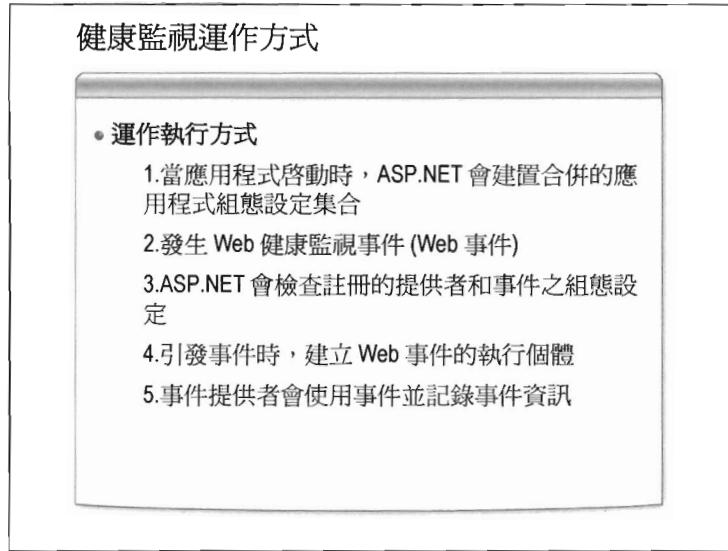
### 使用健康監視 (Health Monitoring)

ASP.NET 能夠輕鬆監視已部署 ASP.NET 應用程式的健康狀態，並提供 ASP.NET 在執行階段使用資源的詳細資訊。ASP.NET 健康監視功能可以用來執行下列工作：

- 在過去，網站系統管理員必須依賴 HTTP 錯誤、偵錯輸出或有限的事件記錄訊息組，被動地監視個別地或跨 Web 伺服陣列監視實體 ASP.NET 應用程式，確認是否正常運作。
- 診斷可能會失敗的 ASP.NET 應用程式。
- 記錄不一定與 ASP.NET 應用程式有關的錯誤，但是對進行診查有所幫助的事件。

在過去，網站系統管理員必須依賴 HTTP 錯誤、偵錯輸出或有限的事件記錄訊息組，被動地監視或診斷應用程式。ASP.NET 健康監視藉由引發 Web 事件，並透過各種機制提供健康監視資料提供主動式的方法。

透過健康監視功能，您可以使用適當的提供者，將網站運作過程中的監控到的資訊記錄在事件日誌、資料庫，或將資訊透過電子郵件傳送至管理者。



## 健康監視運作方式

ASP.NET 在應用程式處理過程中，能夠在各種時機引發 Web 健康監視事件 (也稱為 Web 事件)。提供者 (接聽程式) 會使用 Web 事件，讀取事件中封裝的資訊然後加以記錄。每個提供者記錄 Web 事件資料的方式都不相同。例如，某個提供者會寫入 Windows 事件記錄檔，而其他提供者可能會傳送電子郵件訊息。

下列步驟說明健康監視的運作方式：

1. 當應用程式啟動時，ASP.NET 會建置合併的應用程式組態設定集合。
2. 發生 Web 健康監視事件 (Web 事件)。例如，可能是安全性作業失敗所造成的事件，如 Web 要求的 URL 授權失敗。
3. ASP.NET 會檢查註冊的提供者和事件之組態設定。
4. 引發事件時，建立 Web 事件的執行個體。
5. 事件提供者會使用事件並記錄事件資訊。

### Web 事件與提供者

- 每個 Web 事件對應到一個類別，繼承自 **WebBaseEvent**
- 常用的內建 Web 事件
  - WebAuditEvent
  - WebErrorEvent
- 可利用提供者記錄事件之相關資訊
- 提供者都是繼承自 **WebEventProvider**
  - EventLogWebEventProvider
  - WmiWebEventProvider

## Web 事件與提供者

在網站應用程式中，欲監控的每一個事件稱為「Web 事件(Web event)」。每個 Web 事件以一個.NET 類別代表之。當 ASP.NET 網站運行時，會自動產生相關的 Web 事件，如網站應用程式啓動、關閉或網站發生錯誤等。

所有的 Web 事件都是繼承自 **WebBaseEvent** 類別而來，此類別定義共用的屬性，如事件發生的時間、唯一的識別代碼、訊息…等。內建的事件類別包含：

- **WebManagementEvent** 類別：處理程序的資訊，如代號、名稱，使用來執行的帳號。
- **WebApplicationLifetimeEvent** 類別：記錄應用程式生命周期，在應用程式編譯、啓動或關閉時觸發。
- **WebHeartbeatEvent** 類別：監控應用程式是否持續在運作中，並提供 Web 伺服器處理程序的基本狀態統計資訊，如目前同時執行的要求數量。此事件預設不啓用，您可以在 **Web.Config** 檔案中設定啓用之。

- **WebRequestEvent** 類別：此為自訂事件的父類別，可以偵測 Web 請求的細節。
- **WebAuditEvent** 類別：所有安全性稽核事件的父類別。
- **WebFailureAuditEvent** 類別：所有安全性稽核錯誤事件的父類別。
- **WebAuthenticationFailureAuditEvent** 類別：當網站的表單驗證或成員驗證動作發生失敗時觸發。
- **WebViewStateFailureAuditEvent** 類別：當使用者提交的網頁中包含無效的狀態資訊時觸發，這代表可能有惡意使用者正在進行一些入侵網站的行爲。
- **WebSuccessAuditEvent** 類別：偵測存取特定資源的動作是否成功。
- **WebBaseErrorEvent** 類別：代表 ASP.NET 網站工作處理序產生的例外。
- **WebErrorEvent** 類別：當應用程式組態或編譯過程發生問題時觸發。
- **WebRequestErrorEvent** 類別：特定請求產生錯誤時觸發。

在健康監視系統架構下，您可以使用內建的提供者，將發生的事件之相關資訊，記錄起來，所有的提供者都是繼承自 **WebEventProvider** 而來，內建的提供者包含：

- **WebEventProvider**
- **EventLogWebEventProvider**
- **WmiWebEventProvider**
- **TraceWebEventProvider**
- **BufferedWebEventProvider**
- **SqlWebEventProvider**
- **MailWebEventProvider**
- **SimpleMailWebEventProvider**
- **TemplatedMailWebEventProvider**

## 啓用健康監視功能

- 預設健康監視功能是關閉的
- 使用組態檔案來啓用之
  - 使用 providers 設定適當的提供者
  - 使用 eventMappings 與 Rule 註冊想要監控的事件
  - 使用 Profile 設定節流
- 監控網站是否在活動中
 

```
<healthMonitoring enabled="true" heartbeatInterval="10">
  <rules>
    <add name="MyHeartBeats" eventName="Heartbeats"
        provider="EventLogProvider" profile="Critical"/>
  </rules>
</healthMonitoring>
```

## 啓用健康監視功能

預設健康監視功能是關閉的，您可以設定組態檔案來啓用它，並設定適當的提供者，以便在 Web 事件發生時，自動記錄相關的資訊。在 Machine.config 相同目錄下的 Web.Config 檔案中包含健康監視預設的組態設定，在 < providers > 區段描述包含了 EventLogProvider 、 SqlWebEventProvider 、 WmiWebEventProvider 三個提供者。

組態檔案中的 <eventMappings> 定義可監控的事件。預先定義的 Web 事件對應(Event Mapping)與 Web 事件類別的對照表如下所示：

事件對應	Web 事件類別
All Events	WebBaseEvent
Heartbeats	WebHeartbeatEvent
Application Lifetime Events	WebApplicationLifetimeEvent
Request Processing Events	WebRequestEvent
All Errors	WebBaseErrorEvent
Infrastructure Errors	WebErrorEvent
Request Processing Errors	WebRequestErrorEvent
All Audits	WebAuditEvent
Failure Audits	WebFailureAuditEvent

Success Audits	WebSuccessAuditEvent
----------------	----------------------

組態檔中可使用規則(Rule)註冊提供者來接聽事件。

## 使用節流機制

您可以設定以下節流屬性，避免大量 Web 事件同時產生之後不及處理的現象：

- minInstances：同樣的事件發生指定的次數時，觸發事件給提供者。預設值為 1，將此功能關閉。
- minInterval：若相同的事件未在指定的時間間隔內傳送，觸發事件給提供者。
- maxLimit：記錄事件的數量上限。預設為 Infinite，關閉此功能。

若要重複使用這些節流的屬性，您可以將這些屬性，封裝成 Profile，然後從規則中去參考這個 Profile。以下健康監視組態設定每隔 10 秒監控網站是否在活動中，將監控的資訊寫到事件日誌，使用 profile 設定節流屬性：

```
<healthMonitoring enabled="true" heartbeatInterval="10">
    <rules>
        <add name="MyHeartBeats" eventName="Heartbeats"
            provider="EventLogProvider" profile="Critical"/>
    </rules>
</healthMonitoring>
```

#### 練習9.4：使用健康監視功能監控網站活動

•在練習中，將在Web.config檔案中，設定啓用健康監視功能，並使用Profile來設定參考到預設的節流設定。

•預估實作時間：10分鐘

#### 練習 9.4：使用健康監視功能監控網站活動

##### 目的：

在這個練習中將了解如何使用健康監視功能監控網站活動。

##### 功能描述：

在練習中，將在 Web.config 檔案中，設定啓用健康監視功能，並使用 Profile 來設定參考到預設的節流設定。

**預估實作時間：10 分鐘**

##### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod09\_4\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod09\_4」。

3. 在 Web.config 檔案的<system.web>區段內加入設定，如下：

```
<healthMonitoring enabled="true" heartbeatInterval="10">
  <rules>
    <add name="MyHeartBeats" eventName="Heartbeats"
         provider="EventLogProvider" profile="Critical"/>
  </rules>
</healthMonitoring>
```

4. 執行網站。靜待一些時間後，檢視作業系統事件日誌，將會有應用程式活動中的記錄。

## 總結

- 了解網頁快取
- 了解OutputCache相關屬性
- 了解網頁區塊快取
- 了解資料快取
- 了解ASP.NET 健康監視運作方式

---

# 第十章：

# HTTP 執行時期

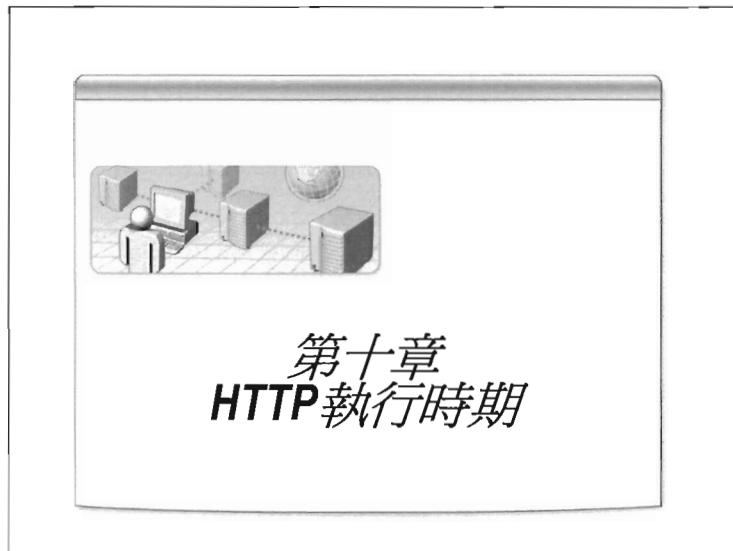
## 本章大綱

ASP.NET Request 處理過程.....	4
HttpApplication 事件.....	6
Global.asax.....	9
HTTP Module 運作模式.....	11
HTTP Module 與 Global.asax 比較.....	12
設計 HTTP Module.....	13
練習 10.1 : 設計 HTTP Module.....	15
註冊 HTTP Module.....	19
練習 10.2 : 註冊及測試 HTTP Module.....	20
何謂 HTTP Handler.....	22
設計 HTTP Handler.....	23
練習 10.3 : 設計 BMPHandler.....	25
註冊 HTTP Handler.....	29
練習 10.4 : 註冊及測試 HTTP Module.....	32

作者：

趙敏翔





---

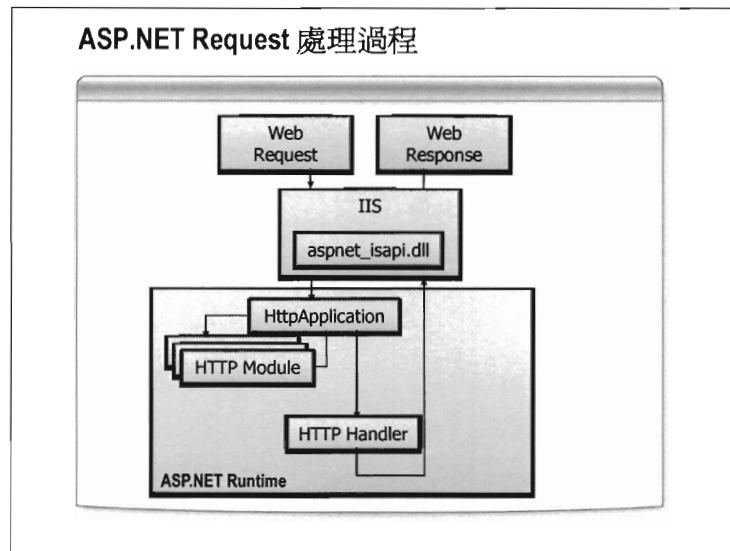
## 大綱

- ASP.NET Request 處理過程
- ASP.NET 應用程式
- 設計 HTTP Module
- 設計 HTTP Handler

要設計一個完善的 Web 應用程式，除了在網頁裡下功夫外，了解整個 ASP.NET 應用程式運作的全貌也是十分重要的一件事。本章將詳細介紹 HTTP Request 的逐步處理過程。

在這一章，你將學會：

- ASP.NET Request 處理過程
- ASP.NET 應用程式運作方式
- 設計及使用 HTTP Module
- 設計及使用 HTTP Handler



## ASP.NET Request 處理過程

在 ASP.NET 中，Request 會先送到 Web 伺服器，再被 ASP.NET 的 ISAPI(Internet Server Application Programming Interface)傳入 ASP.NET 執行時期(Runtime)。

當 IIS 收到一個 Request，會參考 IIS 的設定，根據其 URL 最後的副檔名查詢對應的 ISAPI。在預設的情況下凡是副檔名為.aspx、.asmx、.axd 的 Request 對應的 ISAPI 都是 aspnet\_isapi.dll。aspnet\_isapi.dll 實際只是個單純的 ISAPI，只負責啟動 ASP.NET 執行時期。

### HttpApplication

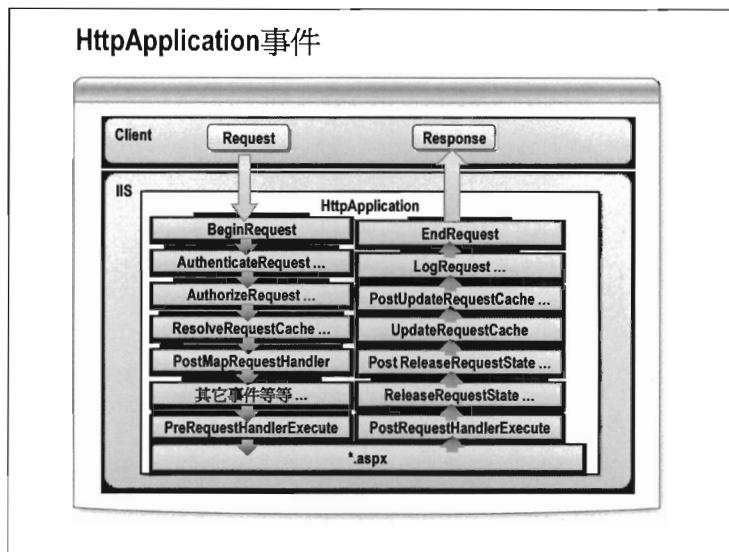
一旦 Request 啓動了 ASP.NET 執行時期，便從 HttpApplication 物件開始運作。HttpApplication 物件是 ASP.NET 的宿主(Host)程式，HttpApplication 物件會讀取此台電腦與此 Web 應用程式的設定檔—Web.config，根據設定檔的設定將 Request 交給一個或多個 HTTP Module 處理。

## HTTP Module

每個 HTTP Module 提供一種服務，像是狀態(Session)管理、身份驗證(Authentication)或個人資訊(Profile)管理。每個 HTTP Module 處理完相關的服務後，會再將 Request 交回給 HttpApplication。

## HTTP Handler

接著 HttpApplication 再參考讀取到的設定檔內容，根據 Request 的存取方法(Verb)及其 URL，將 Request 交給某一個 HTTP Handler 處理。Request 可能的 HTTP 存取方法有：GET、POST、FTP…等。視其設定內容，Request 可能單純地以 ASP.NET 網頁的方式處理，或是啟動其他處理方式，如：顯示偵錯資訊(trace.axd)。



## HttpApplication 事件

ASP.NET 3.5 `HttpApplication` 物件在執行過程中，總共會觸發 10 幾個公用事件，以下說明 ASP.NET 處理 Request 的基本過程及相關事件。

首先是先檢查 Request 內容，Request 交給 `HttpApplication` 物件處理的第一件事就是檢查瀏覽器送來的 Request 內容，判斷是否包含任何可能會有攻擊性的標籤、Script。若要取消此項檢查可在`<%@ Page %>` 指示詞中將 `ValidateRequest` 屬性設定為 `false`。

## URL 對應

檢查 `Web.config` 檔中的 `urlMappings` 區段是否有 URL 設定與目前這個 Request 欲存取的 URL 相同。在 ASP.NET 裡可用 `urlMappings` 將真實的 URL 隱藏起來。

以如下設定為例，當使用者網址輸入 `Home.aspx`，實際上會存取到 `Default.aspx`。

```
<urlMappings enabled="true">
```

```
<add url="~/Home.aspx" mappedUrl="~/Default.aspx" />
<urlMappings>
```

接著將 `HttpApplication` 所觸發的事件順序，條列如下，這邊僅列出常用的事件，若要參考 ASP.NET 所有 `HttpApplication` 完整的事件，可連結到以下微軟 MSDN 網址參考：

<http://msdn.microsoft.com/en-us/library/system.web.httpapplication.aspx>

`HttpApplication` 觸發以下事件的順序為，：

- `BeginRequest`：於新 Request 進入時觸發。每個 Request 都會觸發這個事件。
- `AuthenticateRequest`：驗證 Request 時觸發的事件。
- `AuthorizeRequest`：授權 Request 時觸發的事件。
- `ResolveRequestCache`：與快取處理相關的 HTTP Module 可透過這個事件，檢查是否直接以快取回應 Request 的要求，不需重複執行網頁程式。
- `MapRequestHandler`：發生於選取處理常式以回應要求時，當伺服器是以 Integrated 模式執行 IIS 7.0，而且已安裝 .NET Framework 3.0 以上的版本時，會引發 `MapRequestHandler` 事件。

### 檢查對應的 HTTP Handler

根據 Request 存取的副檔名，自 `Web.config` 檔檢查是否有任何註冊的 HTTP Handler 要處理這種 Request。所有的 HTTP Handler 都要實作 `IHttpHandler` 介面才能處理 Request。

接著觸發以下事件：

- `AcquireRequestState`：取得 Request 的狀態資訊一如：Session 狀態資訊一時觸發的事件。
- `PreRequestHandlerExecute`：在呼叫 HTTP Handler 前觸發的事件。

## 交由 HTTP Handler 處理

呼叫相關 IHttpHandler 的 ProcessRequest 方法(若是非同步模式則是 BeginProcessRequest 方法)。如：若 Request 存取的是個網頁，則由目前網頁的物件來處理這個 Request。

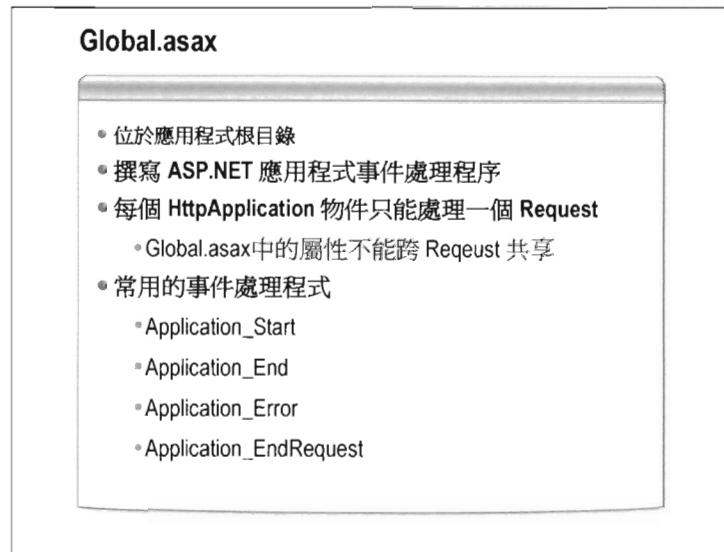
接著觸發以下事件：

- PostRequestHandlerExecute：HTTP Handler 處理完 Request 之後觸發的事件。
- ReleaseRequestState：開始儲存 Request 狀態資訊前觸發的事件。
- PostReleaseRequestState：儲存 Request 狀態資訊後觸發的事件。

接著觸發以下事件：

- UpdateRequestCache：更新快取時觸發的事件。
- LogRequest：發生於 ASP.NET 執行目前要求的任何記錄之前。
- PostLogRequest：發生於 ASP.NET 完成處理 LogRequest 事件的所有事件處理常式時。
- EndRequest：最後觸發的事件。

上列事件中的 LogRequest 和 PostLogRequest，只有在以 Integrated 模式執行 IIS 7.0，且使用 .NET Framework 3.0 以上的版本時，才支援此事件，最後 Error 事件不是每個 Request 都會觸發，只有例外或錯誤發生時才會產生。



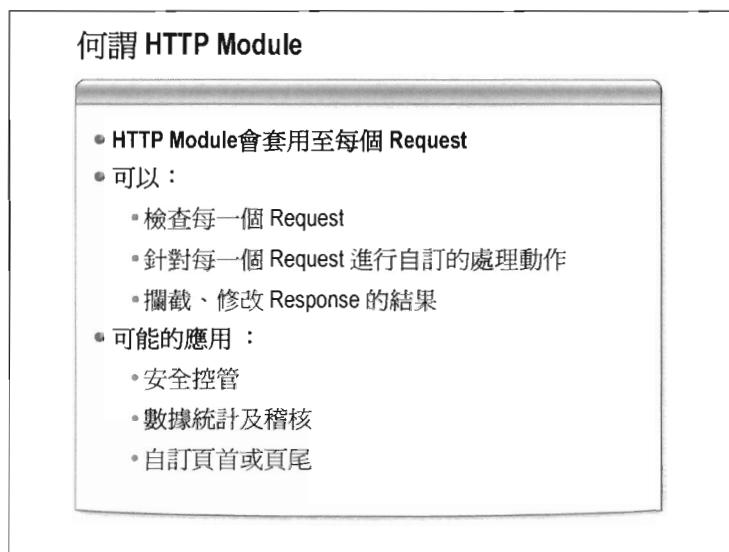
## Global.asax

在 `HttpApplication` 處理 `Request` 的生命週期中，`HttpApplication` 會觸發前面所列的這些事件，若要處理這些事件最簡單的做法就是在應用程式的根目錄下建立一個名為 `Global.asax` 的檔案。

### 常見事件處理程序

Web 應用程式的事件比以下列的還多，不過這些是最常用的幾個事件：

- `Application_Start`：只會在 Web 應用程式啓動時執行。
- `Application_End`：只會在 Web 應用程式結束時執行。
- `Application_Error`：`Request` 處理的任何階段都可能因為發生例外觸發 `Error` 事件。若要處理所有例外發生的情況可加入 `Application_Error` 事件處理程序。
- `Application_EndRequest`：每個 `Request` 都會觸發 `EndRequest` 事件。因此每個 `Request` 都會執行 `Application_EndRequest` 事件處理程序。

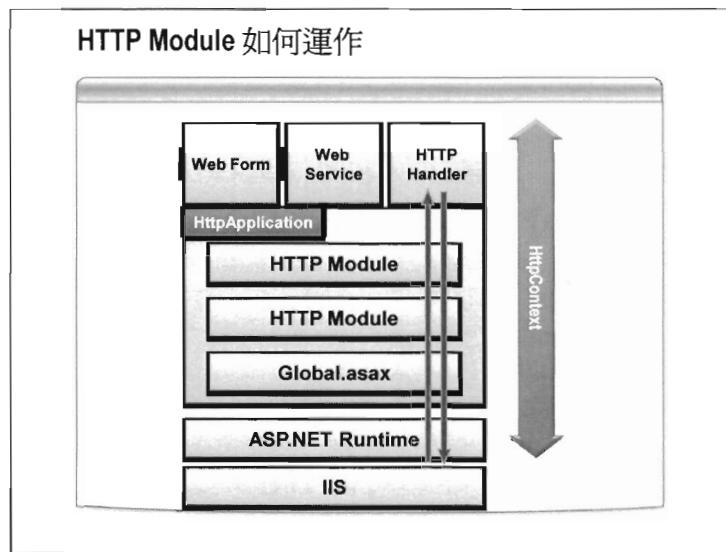


## 何謂 HTTP Module

每個 HTTP Module 提供一種服務，像是狀態(Session)管理、身份驗證(Authentication)或個人資訊(Profile)管理。每個 HTTP Module 處理完相關的服務後，會再將 Request 交回給 HttpApplication。

ASP.NET 應用程式可透過實作 IHttpModule 介面的類別加以擴充。新增的 HTTP Module 本身也可觸發一些事件。若要處理這些事件可在 Global.asax 檔中註冊，預設的命名方式為 *modulename\_event-name*。舉例來說，若要處理 FormsAuthenticationModule 的 Authenticate 事件，可將事件處理程序命名為 FormsAuthentication\_Authenticate。

ASP.NET 透過 HTTP Module 實作許多應用程式的功能，像是身份驗證、快取、狀態管理及 client script service。當某個功能啟動時，某個 HTTP Module 便會在處理 Request 時被呼叫以進行相關作業。



## HTTP Module 運作模式

HTTP Module 作業是獨立於網頁程式之外的。HTTP Module 也可處理或觸發應用程式事件，若要這麼做可在 `Global.asax` 加入程式。  
ASP.NET Request 處理過程時提到每個 Request 都會呼叫 `Web.config` 中註冊的 HTTP Module 組件。

ASP.NET 的 HTTP Module 就像 ISAPI 的篩選器(filters)會套用到所有的 Request，但 HTTP Module 是以.NET 的 managed code 撰寫的，因此可以完整地整合到 `HttpApplication` 的生命週期中。

## HttpContext

而且 HTTP Module 可直接存取每個 Request 的 Context 屬性，如此可將 Request 重新導向到另一個網頁、修改 Request 或是進行任何處理動作。

### HTTP Module 與 Global.asax 檔的差別

- 相同處：
  - 皆可處理 Web 應用程式事件
- HTTP Modules 好處：
  - 可單獨封裝成一個組件以便重複使用
- Global.asax 好處：
  - Session\_Start 事件
  - Session\_End 事件

## HTTP Module 與 Global.asax 比較

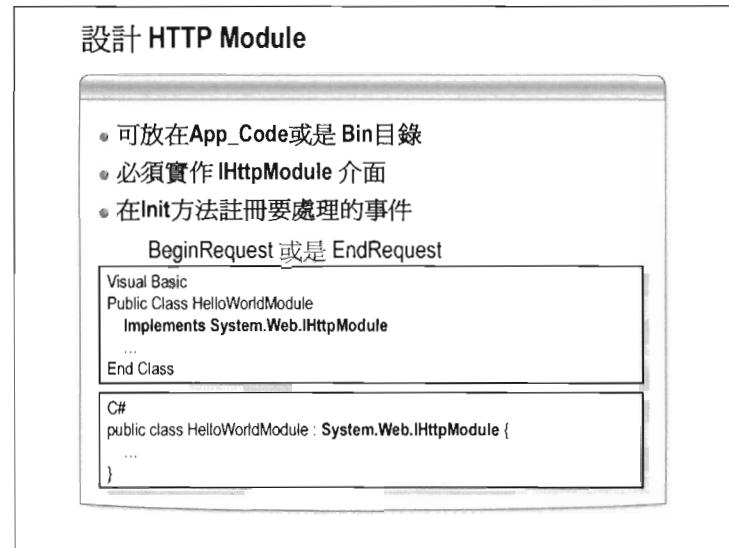
你可在 Global.asax 檔中加入程式產生類似 HTTP Module 的效果：透過 Global.asax 一樣也可以處理 HttpApplication 的事件。經測試結果，若兩者處理同一事件，HTTP Module 中的事件處理程序會比 Global.asax 的事件處理程序早一步執行。

### HTTP Module 的好處

但是 HTTP Module 有一個勝過 Global.asax 的好處：可單獨封裝成一個組件，在多個 Web 應用程式中重複使用。當它被加入 GAC(Global Assembly Cache)時，若再被註冊到掌管整台機器的 Web.config 檔，便可更方便地跨 Web 應用程式使用。

### Global.asax 的好處

不過 Global.asax 也是有勝過 HTTP Module 的地方—提供 HTTP Module 無法攔截得到的事件，像是：Session\_Start 及 Session\_End。



## 設計 HTTP Module

所有的 HTTP Module 都必須實作 IHttpModule 介面。這個類別可直接撰寫在 ASP.NET 的網站中，或是使用編譯好的組件。首先，確認 ASP.NET 網站底下是否有 App\_Code 目錄，若沒有則在專案的根目錄下建立它。

HTTP Module 模組檔可放置在這個目錄下。或者將 HTTP Module 類別撰寫好後，事先編譯成一個.dll 組件，將它放在 Web 應用程式的 Bin 目錄下。

## 實作 IHttpModule 介面

透過「Implements」(Visual Basic)或「:」(C#)將 IHttpModule 介面實作至類別檔中。 IHttpModule 定義在 System.Web.dll 組件中，命名空間也是 System.Web。

```

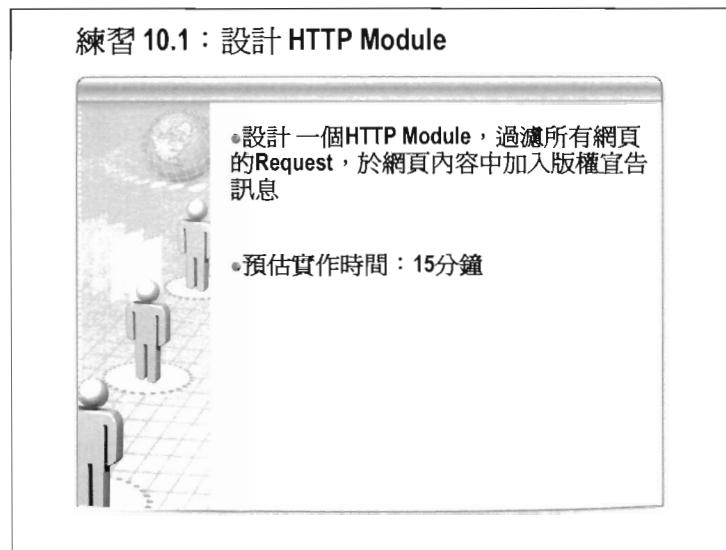
Visual Basic
Public Class HelloWorldModule
    Implements System.Web.IHttpModule
    ...
End Class

```

```
C#
public class HelloWorldModule : System.Web.IHttpModule {
    ...
}
```

## Init 方法

在 IHttpModule 介面的 Init 方法中可註冊要處理的應用程式事件，像是 BeginRequest 或 EndRequest，將應用程式事件與 HTTP Module 裡的方法繫結在一起：當某個特定事件發生時便執行 HTTP Module 中的某個方法，換句話說就是應用程式的事件處理程序。



## 練習 10.1：設計 HTTP Module 於所有網頁加入版權宣告訊息

目的：

設計一個 HTTP Module，過濾所有網頁的 Request，於網頁內容中加入版權宣告訊息。

功能描述：

在這個練習中，將先建立一個 ASP.NET 網站，接著加入 HTTP Module 類別，在其中處理 HttpApplication 物件的 EndRequest 事件，於所有網頁的最後加入版權宣告訊息。

預估實作時間：15 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod10\_1\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod10\_1」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 在「Solution Explorer」視窗中選取專案，點選滑鼠右鍵，選擇「Add ASP.NET Folder」→點選「App\_Code」。
5. 點選 App\_Code 資料夾，自主選單「Web Site」下「Add New Item...」，選取「Class」，命名為「CopyrightModule」。
6. 於類別程式的第一行檢查或匯入以下命名空間：

```
Visual Basic
Imports System.Web
```

```
C#
using System.Web;
```

7. 若使用 C# 語言設計，請跳過此步驟。針對 CopyrightModule 類別設計實作 IHttpModule 介面，繼承後將會自動帶出相關事件程序程式碼：

```
Visual Basic
Public Class CopyrightModule
    Implements IHttpModule

    Public Sub Dispose() Implements System.Web.IHttpModule.Dispose
    End Sub

    Public Sub Init(ByVal context As System.Web.HttpApplication) Implements System.Web.IHttpModule.Init
    End Sub
End Class
```

8. 若使用 Visual Basic 語言設計，請跳過此步驟。使用「:」設計繼承，並刪除預設的建構函式。

```
C#
public class CopyrightModule : IHttpModule {
```

```
{} ...
```

9. 若使用 Visual Basic 語言設計，請跳過此步驟。Visual C#可於繼承後，使用滑鼠右鍵點擊 IhttpModule→選取 Implement Interface→按下 Implement Interface，將會自動帶出事件函式程式碼，並將各事件函式中的 throw new NotImplementedException() 清除。



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

public class CopyrightModule:IHttpModule
{
    #region IHttpModule Members

    public void Dispose()
    {
        throw new NotImplementedException();
    }

    public void Init(HttpContext context)
    {
        throw new NotImplementedException();
    }

    #endregion
}
```

10. 在 Init 方法中註冊要處理的應用程式事件—EndRequest：

Visual Basic AddHandler context.EndRequest, AddressOf Application_EndRequest
---------------------------------------------------------------------------------

C# context.EndRequest += new EventHandler(context_EndRequest);
-------------------------------------------------------------------

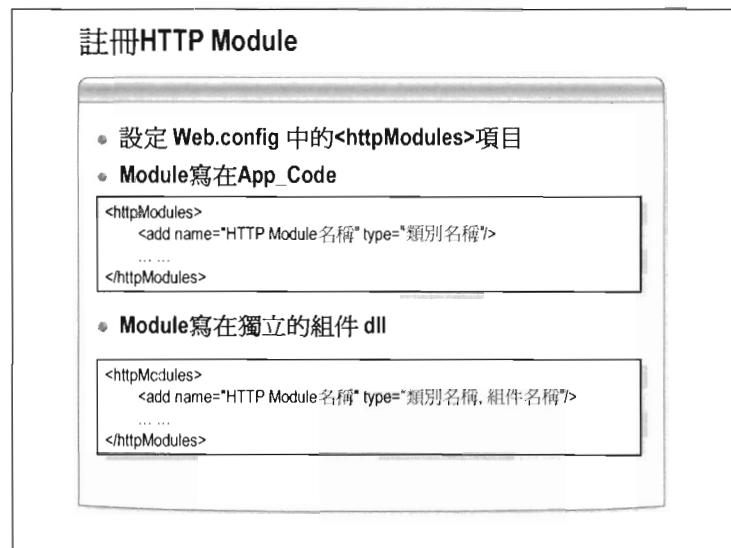
11. 在相對的事件函式 EndRequest 中，撰寫事件處理程序：

Visual Basic <pre>Private Sub Application_EndRequest(ByVal sender As Object, ByVal e As EventArgs)     Dim application As HttpApplication = CType(sender, HttpApplication)     Dim context As HttpContext = application.Context</pre>
---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

```
    context.Response.Write("<h3>本網頁由精誠恆逸提供</h3>
")
End Sub
```

```
C#
void context_EndRequest(object sender, EventArgs e)
{
    HttpApplication application = (HttpApplication)sender;
    HttpContext context = application.Context;
    context.Response.Write("<h3>本網頁由精誠恆逸提供</h3>
");
}
```

12. 點選主選單中的「Build」→「Build Web Site」。確認程式可成功建置。



## 註冊 HTTP Module

設計完 HelloWorldModule 類別後，必須在 Web.config 檔中註冊才會啟動此 HTTP Module 的功能。若 Web 應用程式根目錄下沒有 Web.config 檔，則建立它。

以 HelloWorldModule 為例，假設這個 HTTP Module 的類別完整名稱若為「U9544.HelloWorldModule」、所在的組件檔為「Mod10.dll」，則需加入以下內容至 Web.config 檔，以註冊名為 HelloWorldModule 的 Module：

```
<configuration>
<system.web>
  <httpModule>
    <add name="HelloWorldModule"
        type="U9544.HelloWorldModule, Mod10"/>
  </httpModule>
</system.web>
</configuration>
```

但若 HTTP Module 沒有編譯成單獨的組件，而是直接在「App\_Code」目錄下撰寫 HTTP Module 的類別程式，則不用指定組件名稱。

### 練習 10.2：註冊及測試 HTTP Module

•在ASP.NET網址的Web.config檔中註冊  
自訂的HTTP Module，並加入一個網頁進  
行測試

•預估實作時間：10分鐘

### 練習 10.2：註冊及測試 HTTP Module

#### 目的：

在 ASP.NET 網址的 Web.config 檔中註冊自訂的 HTTP Module，並加入一個網頁進行測試。

#### 功能描述：

延續練習 Mod10\_1 所建立的網站，在其中設定 Web 組態檔 (Web.config)，將 CopyrightModule 註冊為此網站的 HTTP Module。並測試確認 CopyrightModule 是否有作用。

#### 預估實作時間：10分鐘

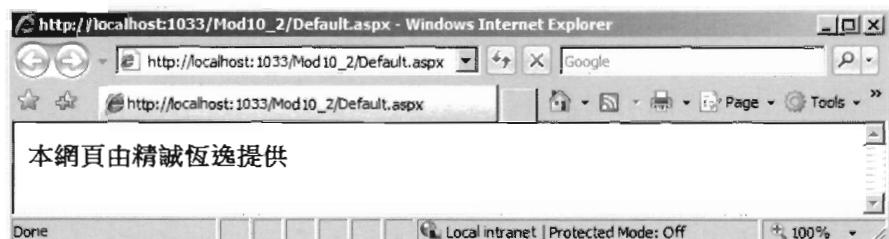
#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 請延續上一個練習專案 Mod10\_1，若是上一個練習未完成，請開啓「\U9544\Practices\VB 或 CS\Mod10\_2\Starter\Mod10\_2」網站。

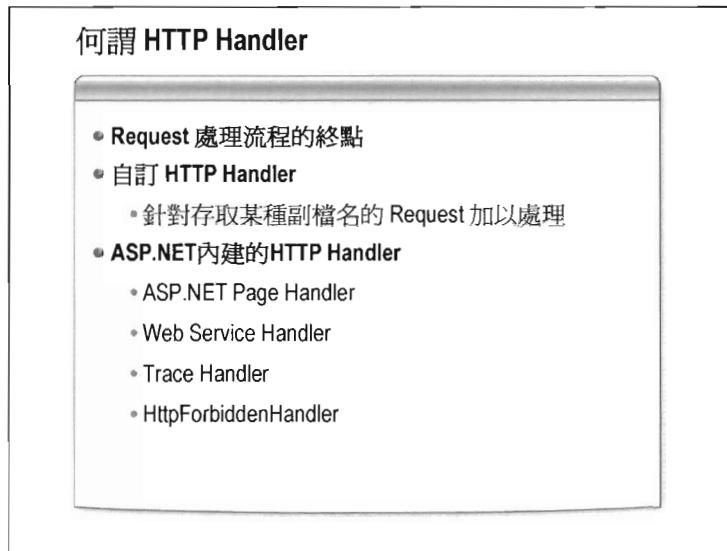
3. 從「File」→「Open Web Site」→選取「ASP.NET Web Site」→選擇「File System」設定 Folder 選取「\U9544\Practices\VB 或 CS\Mod10\_2\Starter\Mod10\_2」目錄，點選「Open」。
4. 開啓 Web.config 檔，找到在其中的  
 <system.web>...</system.web>區段中的 httpModules 項目，加入以下內容：

```
<configuration>
...
<httpModules>
  <add name="CopyrightModule" type="CopyrightModule"/>
...
</httpModules>
...
</system.web>
</configuration>
```

5. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。
6. 確認網頁中是否出現版權宣告訊息。



7. 可在網站中加入第二個、第三個網頁...，都會看到相同的版權宣告訊息。



## 何謂 HTTP Handler

HTTP Handler 會因使用者發出某一種 Request 而被執行，透過 HTTP Handler 裡的程式可產生動態內容。每個 ASP.NET 的網頁其實也是一種 HTTP Handler。你可以透過 HTTP Handler 產生各種內容，而不只是單純、固定的文字或是圖檔。

ASP.NET 的 HTTP Handler 是整個 Request 處理過程的終點(end-point)。最常見的 HTTP Handler 就是處理 ASP.NET 網頁(.aspx)的 Page Handler – PageHandlerFactory。當使用者發出一個 Request 來存取某個.aspx 檔案，這個 Request 就是由 PageHandlerFactory 處理的。

## ASP.NET 的 HTTP Handler 功能

ASP.NET 既有的一些 HTTP Handler 提供很多強大新功能。像是處理 \*.axd 的 HTTP Handler 就是其中一個例子。



## 設計 HTTP Handler

若要撰寫自訂的 HTTP Handler，可在 ASP.NET 網站中加入一個類別，或是先將類別編譯成組成加入 Bin 目錄。所有的 HTTP Handler 都必須實作 IHttpHandler 介面。

### 模組化成類別

確認網站底下是否有 App\_Code 目錄，若沒有則在專案的根目錄下建立它。新增的 HTTP Module 模組可放置在這個目錄下。或者將 HTTP Handler 類別撰寫好後事先編譯成一個.dll 組件，將它放在 Web 應用程式的 Bin 目錄下。

以這種方式建立的好處是可跨 Web 應用程式使用，而且可依需要註冊不同的副檔名進行處理。

### 以.ashx 方式建立

預設 ASP.NET 將副檔名.ashx 保留給 HTTP Handler 使用。以如此方式建立的 HTTP Handler 不需額外的 IIS 或 Web.config 註冊動作便可使用。

```
Visual Basic
<%@ WebHandler language="VB" Class="myHandler" %>
Public Class myHandler
    Implements IHttpHandler
    ...
End Class
```

```
C#
<%@ WebHandler language="C#" Class="myHandler" %>
public class myHandler : IHttpHandler
{
    ...
}
```

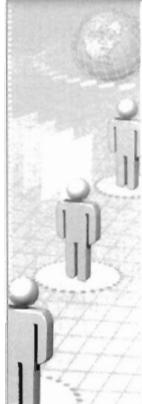
### 實作 IHttpHandler 介面

透過「Implements」(Visual Basic)或「:」(C#)將 IHttpModule 介面實作至類別檔中。IHttpHandler 定義在 System.Web.dll 組件中，命名空間也是 System.Web。

```
Visual Basic
Public Class HelloWorldHandler
    Implements IHttpHandler
    ...
End Class
```

```
C#
public class HelloWorldHandler : IHttpHandler {
    ...
}
```

### 練習 10.3：設計 HTTP Handler 處理圖檔



• 設計一個處理 jpg 的 HTTP Handler，針對使用者存取的 JPG 圖檔，確認是可以回應輸出到網頁上。

• 預估實作時間：15分鐘

### 練習 10.3：設計 BMPHandler

#### 目的：

設計一個處理 jpg 的 HTTP Handler，針對使用者存取的 JPG 圖檔，確認是可以回應輸出到網頁上。

#### 功能描述：

在這個練習中，將先建立一個 ASP.NET 網站，接著加入 HTTP Handler 類別，實作 IHttpHandler 介面，於其中的 ProcessRequest 方法設計確認可以將指定的圖片回應輸出到網頁上。

#### 預估實作時間：15 分鐘

#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod10\_3\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod10\_3」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 將「\U9544\Practices\VB 或 CS\Mod10\_3\Starter」目錄下的 ASP.JPG 複製到專案中。
5. 在「Solution Explorer」視窗中選取專案，點選滑鼠右鍵，選擇「Add ASP.NET Folder」→點選「App\_Code」。
6. 點選 App\_Code 資料夾，自主選單「Web Site」下「Add New Item...」，選取「Class」，命名為「JPGHandler」。
7. 於類別程式的第一行匯入以下命名空間：

```
Visual Basic
Imports System.Web
Imports System.IO
```

```
C#
using System.Web;
using System.IO;
```

8. 若使用 C# 語言設計，請跳過此步驟。針對 JPGHandler 類別設計實作 IHttpHandler 介面，繼承後將會自動帶出相關事件程序碼：

```
Visual Basic
Public Class JPGHandler
    Implements IHttpHandler

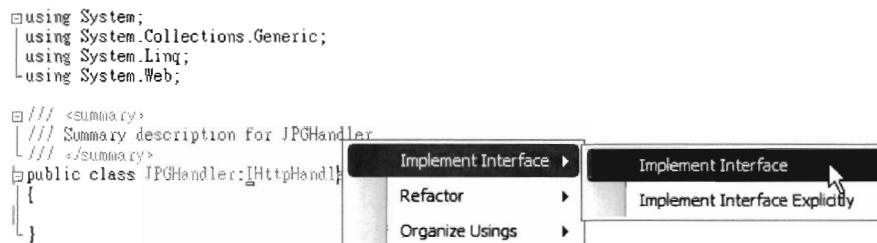
    Public ReadOnly Property IsReusable() As Boolean Implements
        System.Web.IHttpHandler.IsReusable
        Get
            End Get
        End Property

        Public Sub ProcessRequest(ByVal context As System.Web.HttpC
        ontext) Implements System.Web.IHttpHandler.ProcessRequest
            End Sub
    End Class
```

9. 若使用 Visual Basic 語言設計，請跳過此步驟。使用「:」設計繼承，並刪除預設的建構函式。

```
C#
public class JPGHandler:IHttpHandler
{
}
```

10. 若使用 Visual Basic 語言設計，請跳過此步驟。Visual C#可於繼承後，使用滑鼠右鍵點擊 IHttpHandler → 選取 Implement Interface → 按下 Implement Interface。



11. 若使用 Visual Basic 語言設計，請跳過此步驟。Visual C#將會自動帶出事件函式程式碼，並將各事件函式中的 throw new NotImplementedException() 清除。

```
public class JPGHandler:IHttpHandler
{
    #region IHttpHandler Members

    public bool IsReusable
    {
        get { }
    }

    public void ProcessRequest(HttpContext context)
    {
    }

    #endregion
}
```

12. 實作 IsReusable 屬性，回傳值為 true：

```
Visual Basic
Public ReadOnly Property IsReusable() As Boolean Implements System.Web.IHttpHandler.IsReusable
```

```

Get      Return True
End Get
End Property

```

```

C#
public bool IsReusable
{
    get { return true; }
}

```

13. 實作 ProcessRequest 方法。讀取網站中的特定圖片透過  
HttpHandler 輸出到網頁上：

```

Visual Basic
Public Sub ProcessRequest(ByVal context As System.Web.HttpContext) Implements System.Web.IHttpHandler.ProcessRequest
    context.Response.ContentType = "image/jpg"
    Dim fs As FileStream = File.OpenRead(context.Server.MapPath("ASP.JPG"))
    Dim b As Integer
    While (b = fs.ReadByte()) <> -1
        context.Response.OutputStream.WriteByte(CType(b, Byte))
    End While
    fs.Close()
End Sub

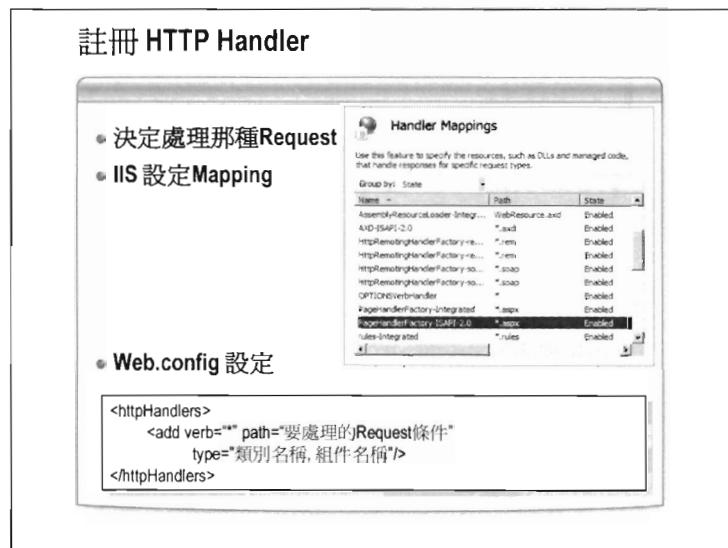
```

```

C#
public void ProcessRequest(HttpContext context)
{
    context.Response.ContentType = "image/jpg";
    FileStream fs =
        File.OpenRead(context.Server.MapPath("ASP.JPG"));
    int b;
    while ((b=fs.ReadByte()) != -1)
    {
        context.Response.OutputStream.WriteByte((byte)b);
    }
    fs.Close();
}

```

14. 點選主選單中的「Build」→「Build Web Site」。確認程式可  
成功建置。



## 註冊 HTTP Handler

與 HTTP Module 一樣，若要啓用某種 HTTP Handler，也必須在 Web.config 檔中註冊才有作用。

### 決定要處理那一種 Request

設計 HTTP Handler 時便需決定它要處理那一種副檔名的 Request。前提是不可與已在 IIS 及 ASP.NET 註冊過的副檔名重複。舉例來說，若設計一個 HTTP Handler 產生 RSS feed，可將此 HTTP Handler 對應到副檔名為.rss 的 Request。

為了讓 ASP.NET 知道那個 HTTP Handler 對應到那種副檔名，必須先在 IIS 裡將這個副檔名的 Request 交給 ASP.NET 處理，接著再於 Web 應用程式的 Web.config 檔裡進行註冊，將這種副檔名的 Request 對應到指定的 HTTP Handler。

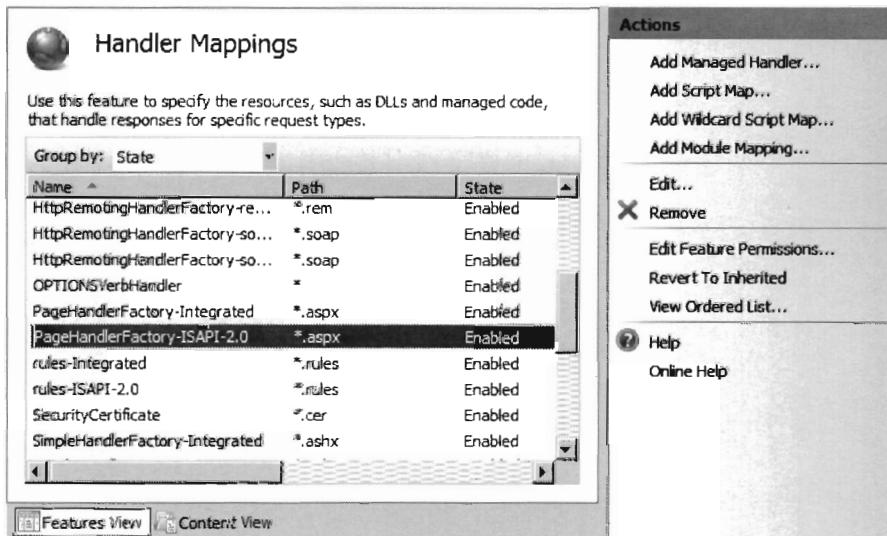
### IIS 7 設定

IIS(Internet Information Services)必需要知道那一種副檔名的 Request 必須轉交給 ASP.NET。IIS7 的設定步驟如下：

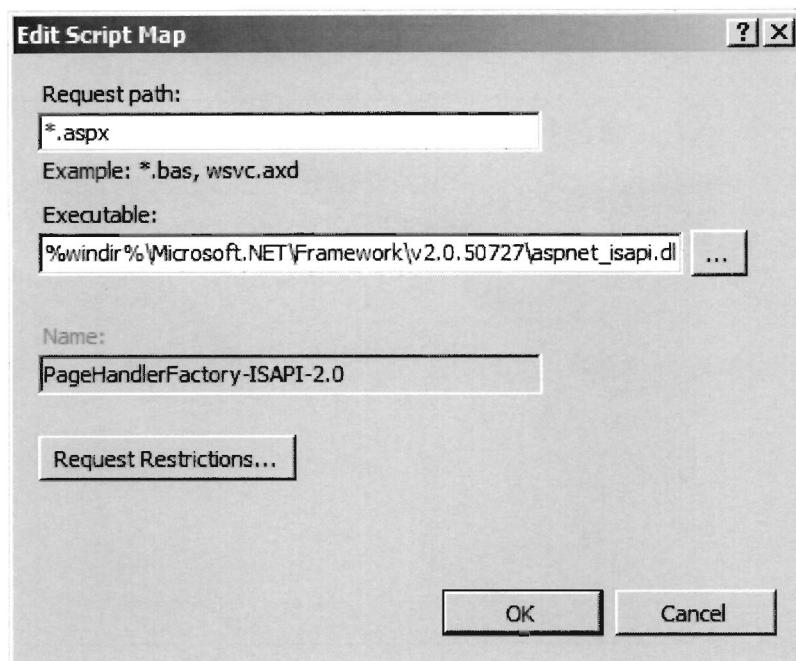
1. 開啓 Internet Information Services (IIS) Manager，選取欲設定的網站，在「Feature View」視窗中，選取 IIS 分類的「Handler Mappings」，如下圖所示：



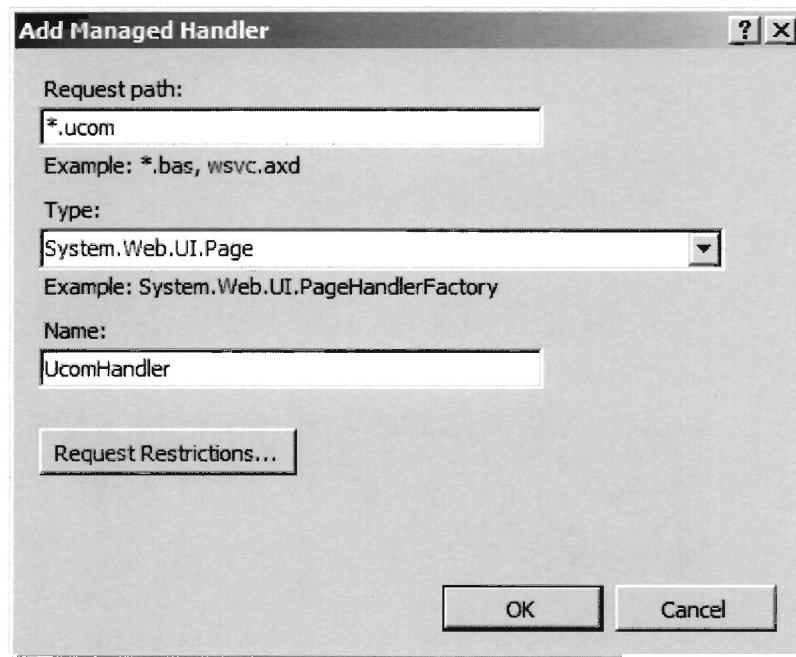
2. 對著「Handler Mappings」雙擊滑鼠，進入「Handler Mappings」編輯視窗。



3. 預設副檔名為.aspx、.ascx、.asmx 及.ashx 都已自動對應到 ASP.NET 的 ISAPI 擴充程式—aspnet\_isapi.dll。但若是這些以外的副檔名便需自行手動對應。
4. 若是要更改目前 IIS7 既有的設定，可直接使用滑鼠雙擊欲更改的 Handler，例如：PageHandlerFactory-ISAPI-2.0。



5. 若是要新增自訂的 Handler，點選右邊 Actions 視窗的「Add Managed Handler...」，並設定相關的資料即可。





## 練習 10.4：註冊及測試 HTTP Module

目的：

在 ASP.NET 網址的 Web.config 檔中註冊自訂的 HTTP Handler，並使用 JPG 圖檔進行測試。

功能描述：

延續練習 10.3 所建立的網站，設定其 Web 組態檔(Web.config)，將 JPGHandler 註冊為此網站處理\*.jpg 的 HTTP Handler。最後存取網頁時，直接存取圖檔確認 JPGHandler 是否有作用。

預估實作時間：10 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 請延續上一個練習專案 Mod10\_3，若是上一個練習未完成，請開啓「\U9544\Practices\VB 或 CS\Mod10\_4\Starter\Mod10\_4」網站。

3. 從「File」→「Open Web Site」→選取「ASP.NET Web Site」→選擇「File System」設定 Folder 選取「\U9544\Practices\VB 或 CS\Mod10\_4\Starter\Mod10\_4」目錄，點選「Open」。
4. 開啓 Web.config 檔，找到在其中的  
`<system.web>...</system.web>` 區段中的 httpModules 項目，加入以下內容：

```
<system.web>
...
<httpHandlers>
    <add verb="*" path="*.jpg" type="JPGHandler"/>
    ...
</httpHandlers>
...
</system.web>
```

5. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。
6. 在網址列，將副檔名改為.JPG，確認網頁中是否出現透過 httpHandlers 所處理的圖片。



## 總結

- 了解ASP.NET Request處理過程
- **ASP.NET**應用程式運作方式
- 學會設計 HTTP Module
- **ASP.NET**套用HTTP Module
- 學會設計 HTTP Handler
- **ASP.NET**套用HTTP Handler

# 第十一章: Web Service 設計

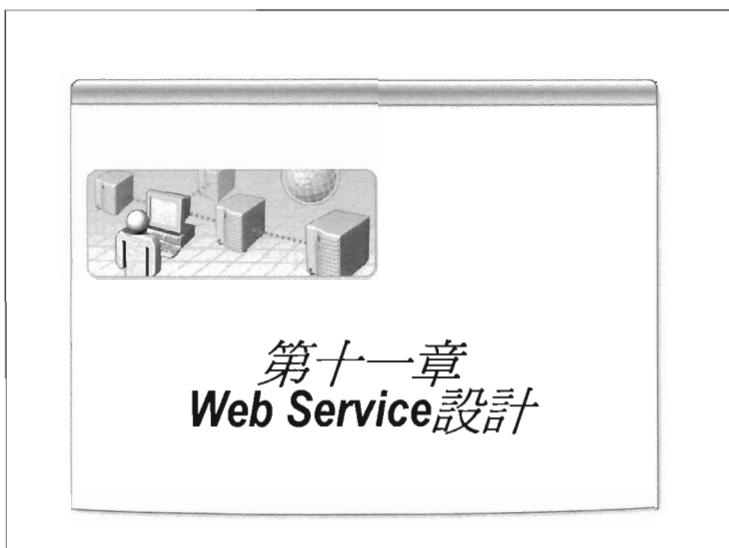
## 本章大綱

什麼是 Web Service .....	4
為什麼使用 Web Service .....	5
找尋並使用 Web Service 運作方式 .....	6
如何建立 Web Service .....	7
Web Service 程式架構 .....	9
練習 11.1: 設計一個 Web Service .....	11
存取 Web Service .....	14
Proxy 的概念 .....	16
Proxy 存取 Web Service .....	17
練習 11.2: 透過 Proxy 呼叫 Web Service .....	19
WCF 用戶端 .....	23
練習 11.3: 使用 WCF 叫用 Web Service .....	24



作者：

趙敏翔



---

## 大綱

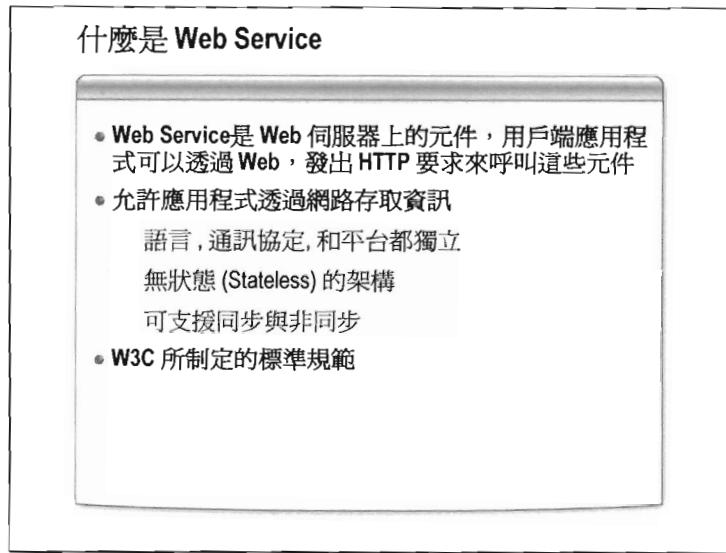
- 認識 Web Service
- 建立 Web Service
- 使用 ASP.NET 呼叫 Web Service
- 使用 WCF Client 呼叫 Web Service
- 總結

Internet 已經從提供網頁畫面的 Web Site 演進到下一代可以程式化的 Web Site，直接將各種組織、應用程式、服務及裝置連結起來，透過這些可程式化的 Web Site 讓 Internet 上的資源更加容易使用、存取，並可提供更聰明的服務。

本章將介紹 Web Service 以及使用 WCF 存取 Web Service 的概念，架構以及設計方式，藉由這些遠端存取的分散式設計技巧，讓各位能更了解 ASP.NET 整合上的強大功能。

在這一章中將學習到

- 認識 Web Service
- 建立 Web Service
- 使用 ASP.NET 呼叫 Web Service
- 使用 WCF Client 呼叫 Web Service



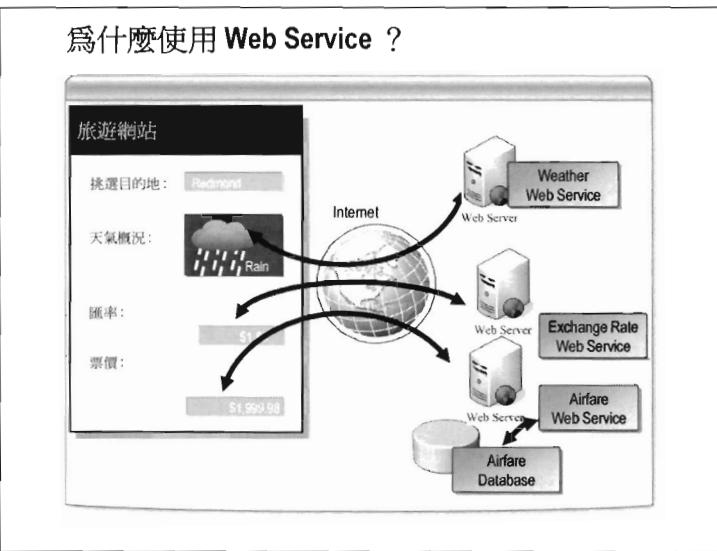
## 什麼是 Web Service

Web Service 提供一種簡單同時具備彈性的程式設計模型，並採用現行 Internet 上的標準，免除各組開發人員、組織在面臨不同平台、作業系統及程式語言時的困擾與紛爭。經由 Web Service 的包裝，系統開發與整合時不用再對每一種程式設計模型、程式語言、作業系統、平台吹毛求疵地進行選美比賽。

Web Service 是一種可經由標準 Internet 通訊協定存取的應用程式元件。開發人員可自 Internet 上的網站取得相關 Web Services 的描述及使用說明，使用這項 Web Service 服務的應用程式或是使用者可以 XML 送、收訊息，所有的功能都以開放的 Internet 通訊協定存取。

所以 Web Service 是：

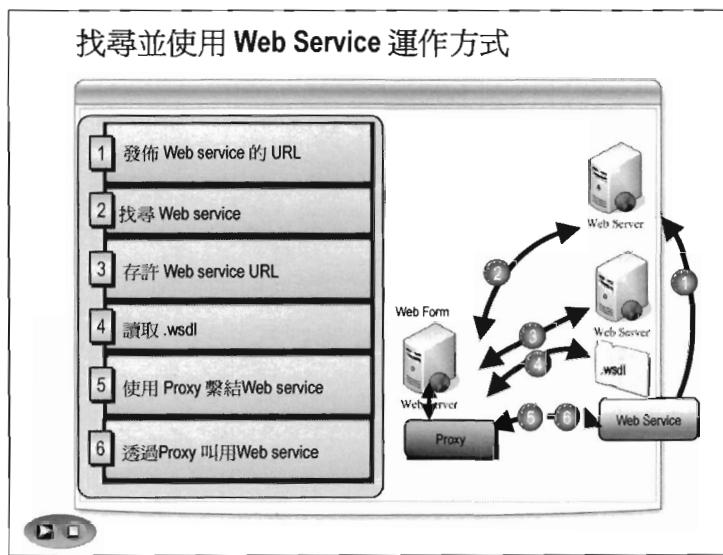
- 一個共通的標準可被用來描述資料，而且是中立於平台與語言的 - XML。
- 一種共通的連線形式的通訊協定，而且是中立於平台與語言的 - SOAP。
- 一個共通的合約語言 (Contract Language) 用來描述 web services 例如：WSDL(Service Description Language)。



## 為什麼使用 Web Service

基本上 Web Service 將元件及服務透過 HTTP、XML 包裝成一個可程式化的 URI(universal resource identifier)，程式設計時建立與使用一個物件就像瀏覽一個網址一樣容易，Web Service 的核心就是 SOAP(Simple Object Access Protocol)，一種精簡的通訊協定，用意在「Programming the Web」。

使用 Web Service 有很多好處，例如：各個不同的系統間存在相同的商業邏輯或是程式邏輯，透過 Web Service 將這些相同的商業邏輯或是程式邏輯封裝起來，開放在網路上提供功能，那麼這些不同的系統，只需要呼叫這個 Web Service 就會擁有相同的功能。



## 找尋並使用 Web Service 運作方式

當設計好 Web Service 之後，如果要讓其他系統來使用，那麼就要將它發佈到網路上，讓其他系統可以找得到。其他系統就可以根據 Web Service 的 WSDL 了解這個 Web Service 所提供的服務有哪些。

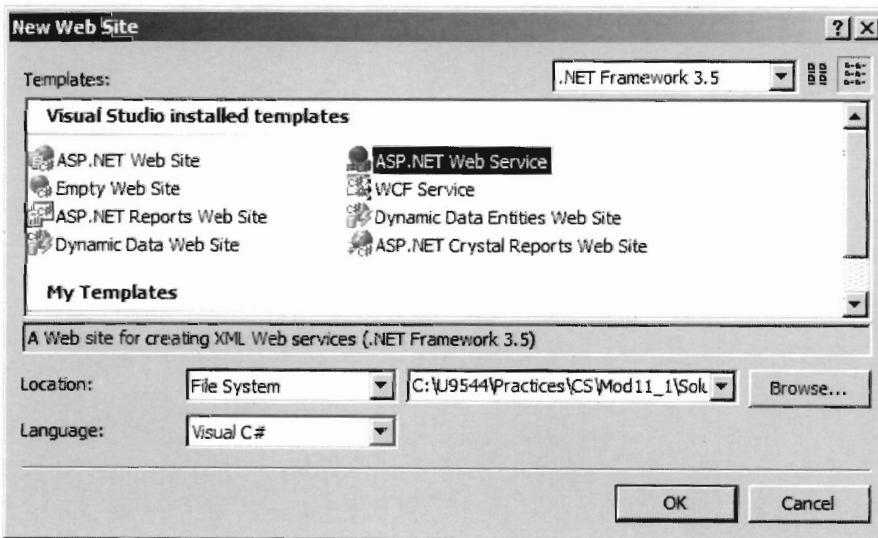
整個 Web Service 的 運作方式，在此分為以下幾個主要步驟：

1. 發佈 Web service 的 URL
2. 用戶端找尋 Web service
3. 用戶端存許 Web service URL
4. 用戶端讀取 .wsdl
5. 用戶端使用 Proxy 繫結 Web service
6. 用戶端透過 Proxy 叫用 Web service

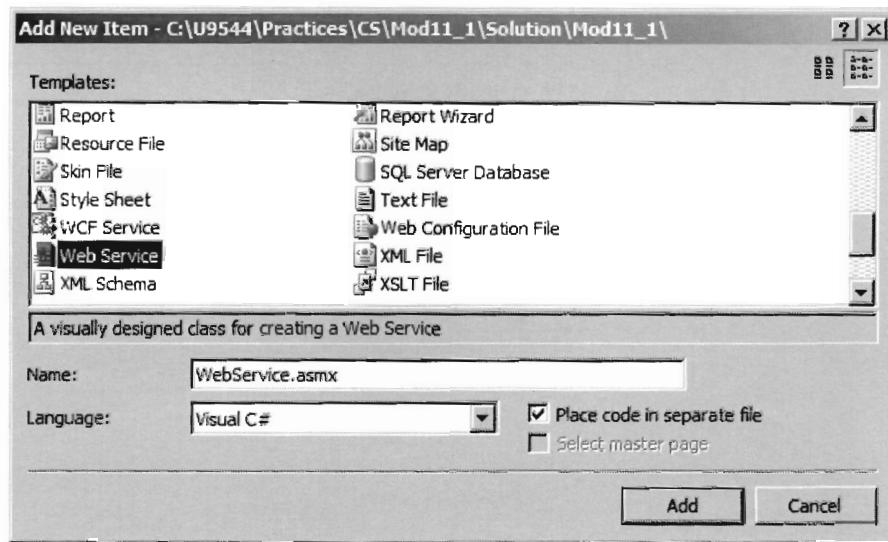


## 如何建立 Web Service

使用 Visual Studio 2008 建立 Web Service 之前，可以先建立一個專門用來提供 Web Service 的專案。

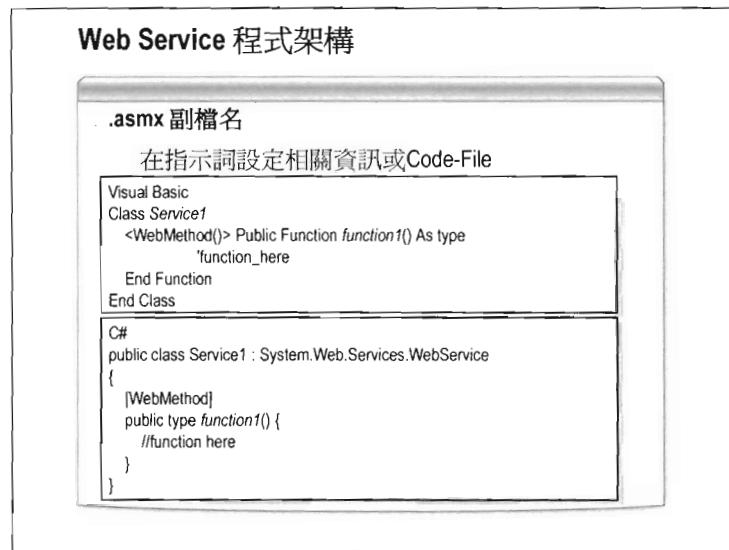


然後直接建立一個 Web Service 的項目就可以開始設計，當使用 Visual Studio 2008 新增一個 Web Service 時，預設會把這個 Web Service 所需要匯入的命名空間以及類別屬性做事先的定義。



Web Service 的指示詞會設定為：

```
<%@ WebService Language="C#或VB" Class="WebService" %>
```



## Web Service 程式架構

設計 Web Service 有幾個主要的特色：

- 副檔名為 asmx
- 類別繼承自 System.Web.Services.WebService
- 開放的功能要用 WebMethod 屬性設定。
- 如果要允許可讓 AJAX 叫用，則必須針對類別設定 System.Web.Script.Services.ScriptService 屬性

Web Service 的程式碼架構如下：

```

Visual Basic
<System.Web.Script.Services.ScriptService()> _
<WebService(Namespace:="http://tempuri.org/")> _
<WebServiceBinding(ConformsTo:=WsiProfiles.BasicProfile1_1)>
    <Global.Microsoft.VisualBasic.CompilerServices.DesignerGenerate
d()> _
    Class Service1
        <WebMethod()> Public Function function1() As type
            'function_here
        End Function
    End Class

```

```
C#
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
[System.Web.Script.Services.ScriptService]
public class Service1 : System.Web.Services.WebService
{
    [WebMethod]
    public type function1() {
        //function here
    }
}
```

### 練習 11.1: 設計一個 Web Service

在這個練習中，你將學習瞭解如何設計一個Web Service，並且設計一個可以接收參數，且回傳資訊的方法。

預估實作時間：10分鐘



### 練習 11.1：設計一個 Web Service

#### 目的：

在這個練習中，你將學習瞭解如何設計一個 Web Service，並且設計一個可以接收參數，且回傳資訊的方法。

#### 功能描述：

在這個練習中，你將設計一個 Web Service，並且設計一個方法，此方法可以接收一個字串參數，再回傳資料。

#### 預估實作時間：10分鐘

#### 實作步驟：

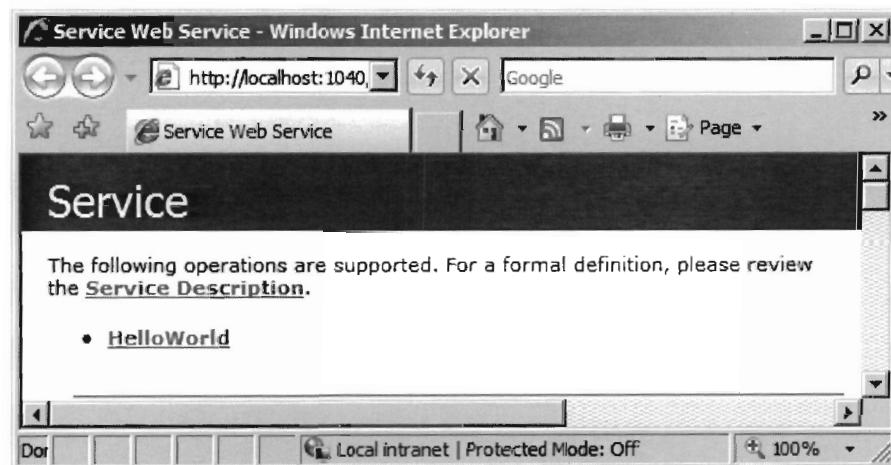
1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Service」→將「Location」設為「File System」並點選「Browse...」按鈕選取「U9544\Practices\VB 或 CS\Mod11\_1\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod11\_1」。

3. 開啓 App\_Code 資料夾下的 Service.cs。
4. 在 Service.cs 中修改預設的 HelloWorld 方法，變更為加入一個 userName 參數，並將此訊息回傳，程式碼如下：

```
Visual Basic
<WebMethod()>
Public Function HelloWorld(ByVal userName As String) As String
    Return "Hello World :" + userName
End Function
```

```
C#
[WebMethod]
public string HelloWorld(string userName)
{
    return "Hello World :" + userName;
}
```

5. 使用瀏覽器執行 Web Service 測試。在「Solution Explorer」→點選 Service.asmx → 按滑鼠右鍵 → 選「View In Browser」。

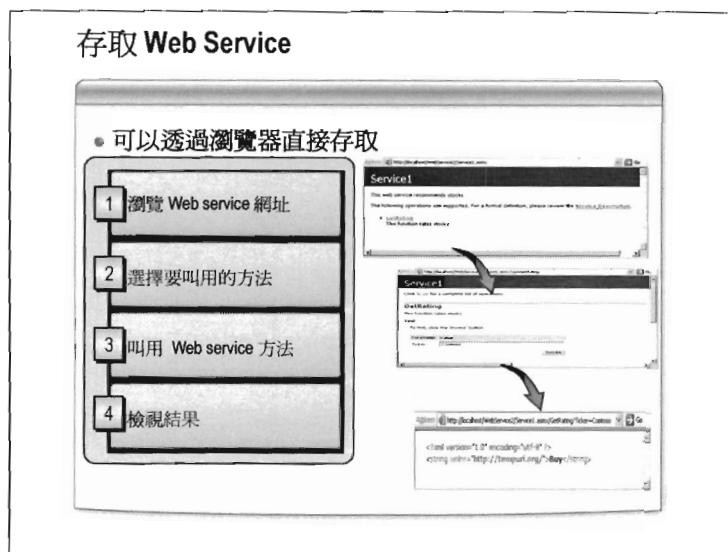


6. 按下開放出來的服務功能 HelloWorld，並且在參數的地方輸入一個使用者名稱，按下「Invoke」執行測試。



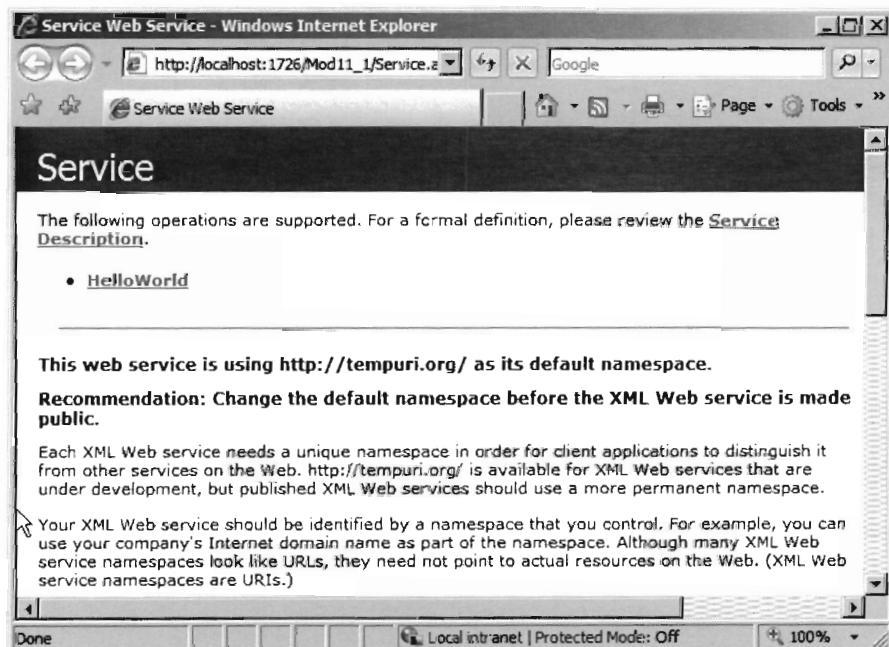
7. 叫用 Web Service 成功之後，將會將訊息回傳回來。





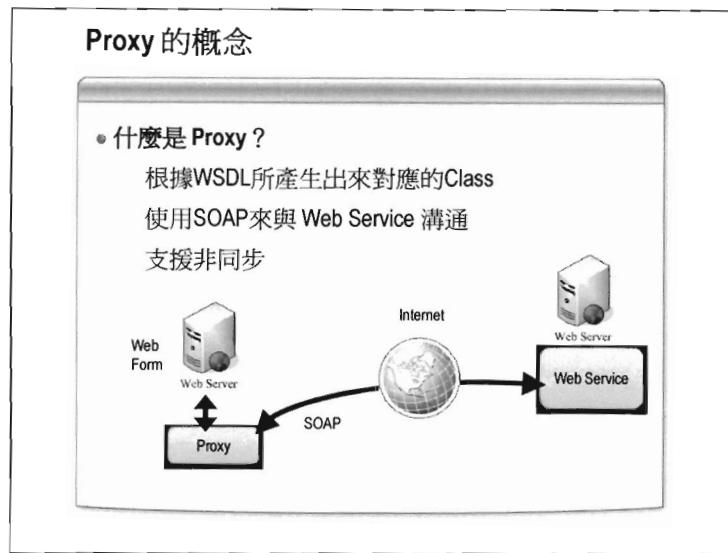
存取 Web Service

要測驗開發好的 Web Service 成不成功，是不是可以順利呼叫，最簡單的存取方式就是直接透過瀏覽器去瀏覽 asmx 副檔名的應用程式。



當使用瀏覽器打開 Web Service 之後，將可看到此 Web Service 所提供的方法有哪些？點進去方法中，可以直接用網頁執行 Web Service 的功能測試，測試後，Web Service 會將結果回傳顯示在網頁上。

如果在 URL 之後加入?WSDL，同樣的內容會以 XML 的格式、使用 Web Service Description Language(WSDL)的語法來展現，這份 WSDL 檔案十分重要，因為其它的應用程式便是參考這個檔案所提供的資訊來存取這項 Web Service。

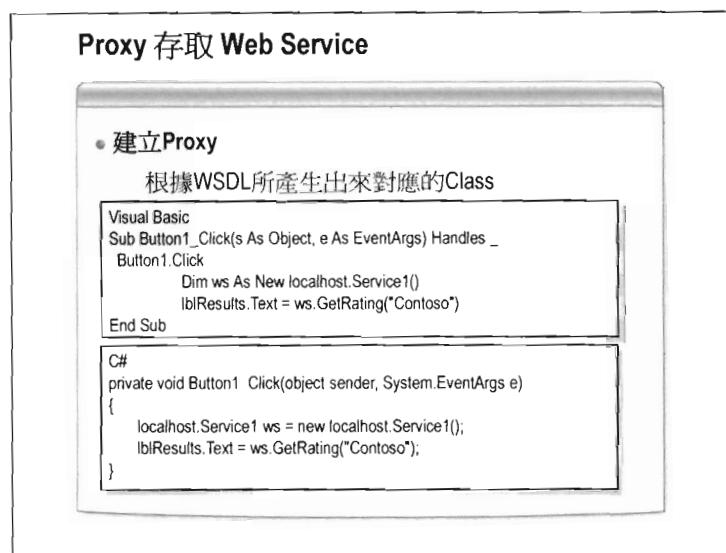


## Proxy 的概念

使用 Web Browser 存取 Web Service 雖然簡單，但畢竟這是給人查詢用的介面，總不能凡事都透過人進行，如果要真正善用所有的資訊服務，應用程式之間應該可以彼此直接交談、整合，這個時候便需透過 Web Service 的程式化介面。

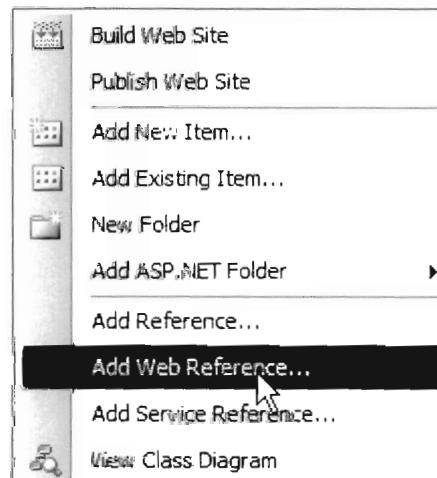
Web Service 程式化介面的底層使用 SOAP 來進行資料的傳送、編碼、解碼，將要呼叫的函式及程式中的參數及其資料內容對應、編碼成 XML 格式的 SOAP 封包傳送給 Web Service，接著再等待從 Web Service 傳回的 SOAP 封包、解析其中 XML 的內容、還原成程式可接收的變數及資料型別…。

為了節省程式開發人員寶貴的時間專注於程式功能的開發，.NET Framework 的 SDK 提供了一個工具程式「Wsdl」自動產生相關的 SOAP 封包處理程式碼，也就是所謂的 Proxy，概念上便是將遠端的 Web Service 包裝成一個物件，讓程式設計師可以用一般存取本機物件、呼叫其函式的相同做法。



## Proxy 存取 Web Service

使用 Visual Studio 開發工具建立的專案，如果想要存取遠端的 Web Service，最方便的設計方式就是在專案中加入 Web Reference。



當專案中加入 Web Reference，針對已開放的 Web Service 設定好之後，應用程式將會自動對應根據遠端的 WSDL 來產生對應的物件，接著只需要建立此 Proxy 物件的執行個體，並叫用他所提供的方法即可，範例程式如下：

```

Visual Basic
Sub Button1_Click(s As Object, e As EventArgs) Handles _

```

```
Button1.Click  
Dim ws As New localhost.Service1()  
lblResults.Text = ws.GetRating("Contoso")  
End Sub
```

```
C#  
private void Button1_Click(object sender, System.EventArgs e)  
{  
    localhost.Service1 ws = new localhost.Service1();  
    lblResults.Text = ws.GetRating("Contoso");  
}
```

### 練習 11.2: 透過 Proxy 呼叫 Web Service



在這個練習中，你將學習瞭解如何設計一個可存取Web Service的Proxy，並且在ASP.NET網頁上透過此Proxy來叫用 Web Service。

預估實作時間：10分鐘

### 練習 11.2 : 透過 Proxy 呼叫 Web Service

#### 目的：

在這個練習中，你將學習瞭解如何設計一個可存取 Web Service 的 Proxy，並且透過此 Proxy 來叫用 Web Service。

#### 功能描述：

在這個練習中，使用 ASP.NET 專案先將已經設計好的 Web Service 加入參考，藉以產生 Proxy，接著在網頁中，透過建立 Proxy 去存取遠端 Web Service。

#### 預估實作時間：10 分鐘

#### 實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9544\Practices\VB 或 CS\Mod11\_2\Starter」目錄，與使

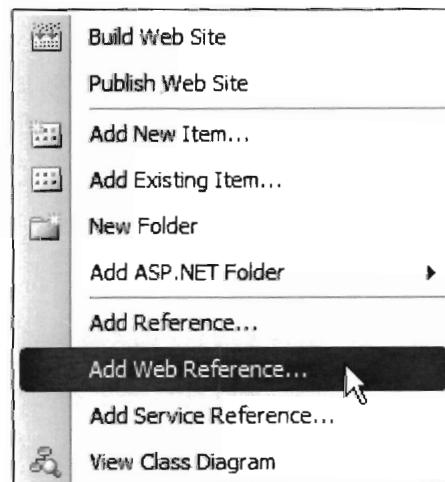
用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod011\_2」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 請延續開啓上一個練習 Mod11\_1 的 Web Service，或是開啓「\U9544\Practices\VB 或 CS\Mod11\_2\Starter」目錄下的 Mod11\_1 網站，並且使用瀏覽器瀏覽 Service.asmx，請將 Service.asmx 的網址先複製下來，如：

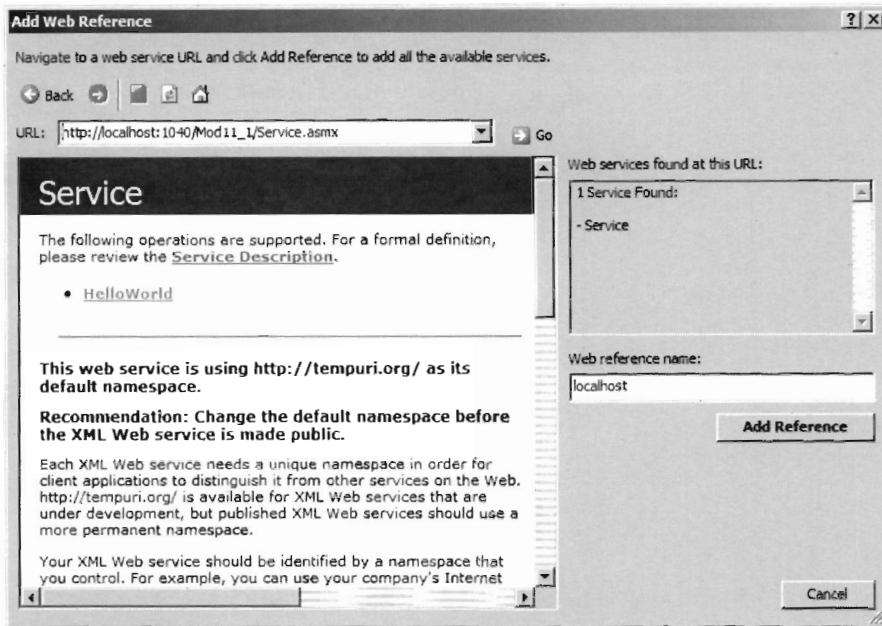
`http://localhost:1040/Mod11_1/Service.asmx`

(請注意：上列的 Port Number 可能會依據實際上的操作而有不同的 Port Number)

5. 回到 Mod11\_2 網站，在 Visual Studio 2008 中點選專案→滑鼠右鍵→點選「Add Web Reference」。



6. 在「Add Web Reference」視窗中的 Url 位置將第 4 步驟所複製的 Web Service 網址貼上，按下「GO」，找到服務之後→接著按下「Add Reference」。



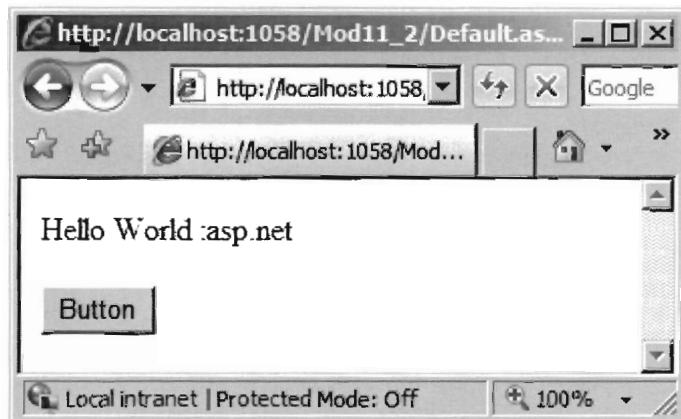
7. 回到網頁設計畫面，從工具箱中拖拉一個 Button 按鈕到畫面中，並且在 Button 的 Click 事件中撰寫程式建立 Proxy 物件叫用 Web Service。

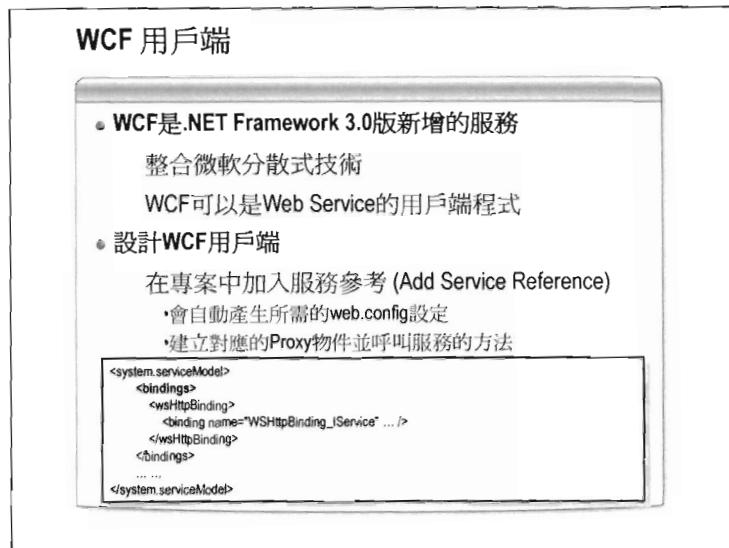
```
Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim ws As New localhost.Service
    Response.Write(ws.HelloWorld("asp.net"))
End Sub
```

8.

```
C#
protected void Button1_Click(object sender, EventArgs e)
{
    localhost.Service ws = new localhost.Service();
    Response.Write (ws.HelloWorld("asp.net"));
}
```

9. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。按下按鈕叫用 Web Service 檢視訊息是否成功回傳。



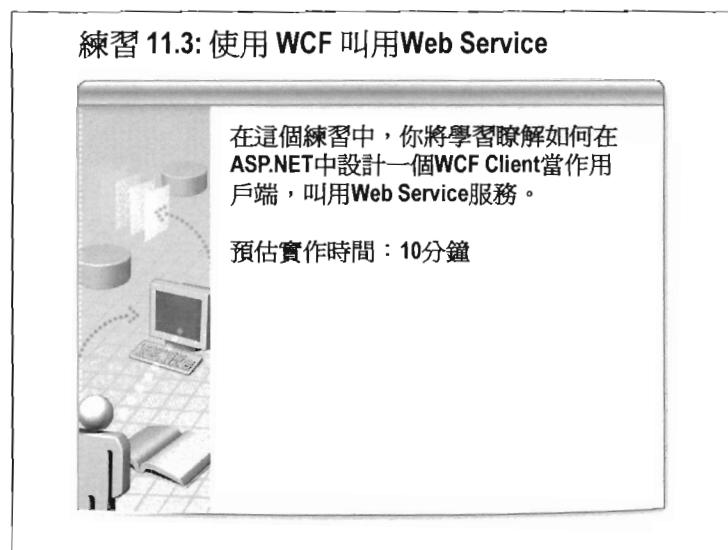


## WCF 用戶端

Windows Communication Foundation (WCF) 是分散式運算的一種機制，所談的都是有關服務，諸如：建立服務、裝載服務、使用服務以及服務的安全性考量等等。微軟提出 WCF 試著集遠端存取技術於一身，也就是說將來透過 WCF 來實作分散式架構的程式系統，既可以走 HTTP 也可以走 TCPIP，可說是將來 SOA 的新標準。

使用 WCF 用戶端來存取 Web Service，最主要是要在組態檔中定義好溝通的方式，而透過 Visual Studio 2008 在專案中使用『Add Service Reference』，便會自動將所需要的溝通設定全部自動設定完成，這也是使用 Visual Studio 2008 開發 WCF 用戶端叫用 Web Service 的最簡易方式。

```
<system.serviceModel>
<bindings>
<wsHttpBinding>
<binding name="WSHttpBinding_IService" ... />
</wsHttpBinding>
</bindings>
...
</system.serviceModel>
```



### 練習 11.3 : 使用 WCF 叫用 Web Service

目的：

在這個練習中，你將學習瞭解如何在 ASP.NET 中設計一個 WCF Client 當作用戶端，叫用 Web Service 服務

功能描述：

在這個練習中，將在 ASP.NET 網站中先加入 Web Service 的 Service Reference，以產生 WCF Client 的 Proxy，在建立這個 Proxy 物件去叫用 Web Service。

預估實作時間：10 分鐘

實作步驟：

1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
2. 從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選

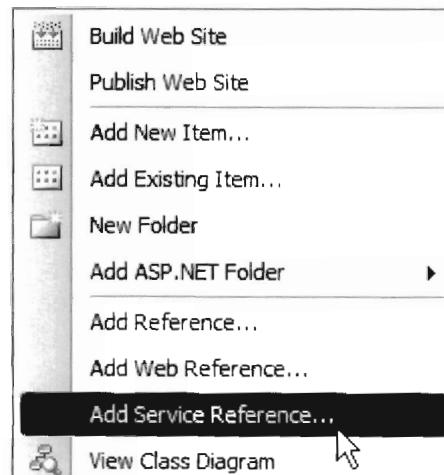
取「\U9544\Practices\VB 或 CS\Mod11\_3\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod011\_3」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，清除「Place code in separate file」核取方塊，新增一個網頁，使用預設的檔名命名。
4. 請延續開啓第一個練習 Mod11\_1 的 Web Service，或是開啓「\U9544\Practices\VB 或 CS\Mod11\_3\Starter」目錄下的 Mod11\_1 網站，並且使用瀏覽器瀏覽 Service.asmx，請將 Service.asmx 的網址先複製下來，如：

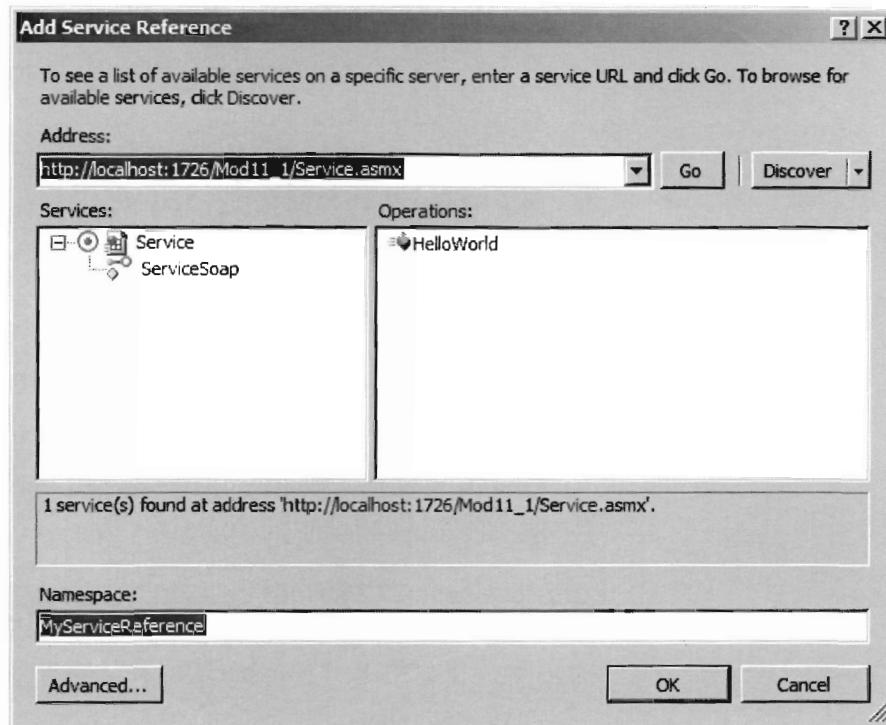
`http://localhost:1726/Mod11_1/Service.asmx`

(請注意：上列的 Port Number 可能會依據實際上的操作而有不同的 Port Number)

5. 回到 Mod11\_3 網站，在 Visual Studio 2008 中點選專案→滑鼠右鍵→點選「Add Service Reference...」。



6. 在「Add Service Reference」視窗中的 Url 位置將第 4 步驟所複製的 Web Service 網址貼上，按下「GO」。
7. 找到服務之後，修改 Namespace 為 MyServiceReference，接著按下「OK」。



8. 當在 WCF Client 加入 Web Service 參考後，會自動在組態檔將所需要的端點條件設定好，其中最主要的就是<client>項目 endpoint 中的 address、binding 和 contract：

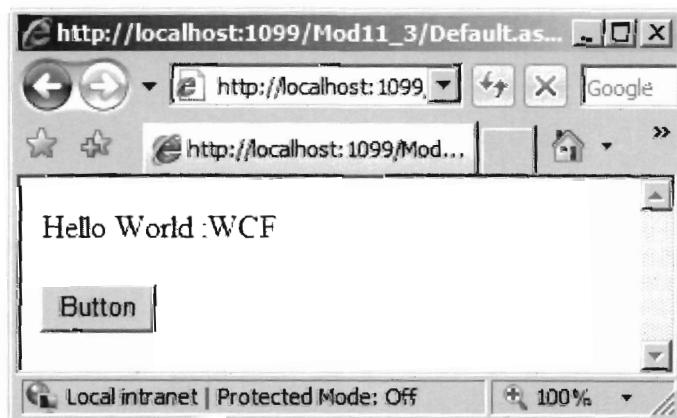
```
<system.serviceModel>
  <bindings>
    <basicHttpBinding>
      <binding name="ServiceSoap" ... ... />
      <security mode="None">
        <transport clientCredentialType="None" proxyCredentialType="None"
          realm="" />
        <message clientCredentialType="UserName" algorithmSuite="Default" />
      </security>
    </binding>
  </basicHttpBinding>
</bindings>
<client>
  <endpoint address="http://localhost:1726/Mod11_1/Service.asmx"
    binding="basicHttpBinding" bindingConfiguration="ServiceSoap"
    contract="MyServiceReference.ServiceSoap" name="ServiceSoap" />
</client>
</system.serviceModel>
```

9. 回到網頁設計畫面，從工具箱中拖拉一個 Button 按鈕到畫面中，並且在 Button 的 Click 事件中撰寫程式建立 WCF Client 物件叫用 Web Service。

```
Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim wcfClient As New MyServiceReference.ServiceSoapClient
    Response.Write(wcfClient.HelloWorld("WCF"))
End Sub
```

```
C#
protected void Button1_Click(object sender, EventArgs e)
{
    MyServiceReference.ServiceSoapClient wcfClient =
        new MyServiceReference.ServiceSoapClient();
    Response.Write(wcfClient.HelloWorld("WCF"));
}
```

10. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。按下按鈕叫用 Web Service 檢視訊息是否成功回傳。



## 總結

- 認識 Web Service
- 建立 Web Service
- 使用 ASP.NET 呼叫 Web Service
- 使用 WCF Client 呼叫 Web Service