

教材編號：U9543a

微軟 MCTS/MCPD 認證系列
ASP.NET 3.5 網站開發基礎
(Fundamental Visual Studio 2008: ASP.NET 3.5)



恆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
高雄市中正三路 55 號 10 樓
TEL:07-2246222 FAX:07-2226262
<http://edu.uuu.com.tw>

【版權所有，未經恆逸書面授權請勿翻印或轉載】

微軟 MCTS/MCPD 認證系列
ASP.NET 3.5 網站開發基礎
Fundamental Visual Studio 2008: ASP.NET 3.5

SYSTEX
making it happen 精誠資訊

UCOM 恒逸資訊
教育訓練中心
Information Technology Education Center

作者／許薰尹、周季賢
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

版權所有，未經恒逸書面授權請勿翻印或轉載



ASP.NET 3.5 網站開發基礎 (Fundamental Visual Studio 2008: ASP.NET 3.5)

微軟 MCTS/MCPD 認證系列

教材編號：U9543a

教材大綱

第一章: ASP.NET 3.5 與 Visual Studio 2008 介紹與運用

什麼是 Microsoft.NET ?	4
.NET Framework 的成員	4
.NET Framework 的優勢	6
.NET 的開發工具 : Visual Studio 2008	7
ASP.NET 簡介	8
什麼是 ASP.NET	9
ASP.NET Web 應用程式	11
ASP.NET 網頁執行模型	12
.NET 專用開發工具 : Visual Studio 2008	13
Visual Studio 2008 能做什麼?	14
Visual Studio 2008 的專案範本	16
Visual Studio 2008 開發環境介紹	18
Web 應用程式的類型與建立方式	25
ASP.NET Web Application	26
ASP.NET Web Site	29
網站應用程式檔案	33

第二章: 設計 ASP.NET 網頁

Web Form 的組成	4
預設網頁模型(Page Model)	7
內嵌程式碼區塊(Embedded Code Blocks)	11
網頁間的互相導向	13
練習 2.1 : ASP.NET 的網頁導向	16
不同網站的網頁間傳遞值	20
同一網站網頁間傳遞值	22
練習 2.2 : 在 ASP.NET Web 網頁之間傳遞值	25
Postback	30
練習 2.3 : 了解 Postback	32

第三章: 使用網頁控制項

課程大綱	3
ASP.NET 的基本控制項類型	4
伺服器控制項與 Html 控制項	5
控制項的事件	7
各種常見的控制項	9
TextBox	10
Image	12
HiddenField	13
Button 、 ImageButton 與 LinkButton	14
練習 3.1 : 使用 Button 、 HiddenField 與 TextBox 控制項	16
CheckBox 與 RadioButton	19
清單系列控制項	21
練習 3.2 : 使用清單系列控制項	24
Panel	28
Calendar	29

練習 3.3 : 使用 Calendar 控制項.....	30
呼叫用戶端指令碼 <i>Java Script</i>	33
動態註冊用戶端指令碼 — 1	34
動態註冊用戶端指令碼 — 2	36
練習 3.4 : 動態註冊用戶端指令碼	37

第四章: 驗證使用者輸入

關於 ASP.NET 的驗證使用者輸入	4
ASP.NET 驗證控制項的功能	5
ASP.NET 驗證控制項的特性	7
ASP.NET 驗證控制項的共有屬性	9
驗證控制項使用方式介紹	11
RequiredFieldValidator 控制項	12
RangeValidator 控制項	13
CompareValidator 控制項	14
RegularExpressionValidator 控制項	15
練習 4.1 : 使用驗證控制項 – 1	16
CustomValidator 控制項	20
練習 4.2 : 使用 CustomValidator 控制項	22
錯誤訊息的顯示方式	25
設計 ASP.NET 網頁驗證的注意事項	28
練習 4.3 : 驗證控制項的錯誤訊息與分組	30

第五章: 追蹤與除錯

應用程式中常見的錯誤種類	4
Visual Studio 支援的除錯種類	6
伺服器端程式除錯	7
練習 5.1 : 設定中斷點	10
用戶端指令碼除錯	13
應用程式追蹤	15
.NET Framework 應用程式追蹤	17
練習 5.2 : 使用 Debug 物件	19
ASP.NET 網頁追蹤	21
網頁層級追蹤	23
應用程式層級追蹤	25
練習 5.3 : 執行頁面層級追蹤	27
自訂輸出	30

第六章: 設計主版頁面與網站導覽系統

什麼是 Master Page	4
Master Page 架構簡介	6
設計 Master Page	8
練習 6.2 : 設計 Master Page	10
Master Page 與 Content Page 之間的溝通	14
練習 6.2 : 在 Content Page 中控制 Master Page 的內容	18

動態變更對應的 Master Page	21
什麼是網站導覽控制項？	22
建立 Web.sitemap 檔案	24
練習 6.3：建立 Web.sitemap 檔案.....	26
使用網站導覽控制項.....	28
Menu 伺服器控制項	29
TreeView 伺服器控制項	31
SiteMapPath 伺服器控制項.....	33
練習 6.4：使用網站導覽控制項	35

第七章: 佈景主題與樣式表

佈景主題、面板與樣式表	4
佈景主題建立方式	6
面板設計方式	7
樣式表設計方式	9
設定網頁套用佈景主題	13
練習 7.1：設計佈景主題-利用面板檔	15
練習 7.2：設計佈景主題-利用樣式表	18
停用佈景主題設定	21
Theme 與 StylesheetTheme	23
動態變更網頁的樣式	25
練習 7.3：動態變更網頁的樣式	26

第八章: 網頁資料存取開發

ADO.NET 簡介	4
使用 Visual Studio 2008 工具管理資料庫.....	5
練習 8.1：利用 Visual Studio 2008 管理資料庫.....	7
ASP.NET 資料來源模型 (Data Source Model).....	11
資料繫結語法	15
GridView 控制項	16
練習 8.2 :使用 GridView 控制項	17
FormView 控制項	21
練習 8.3 :使用 FormView 控制項	22
DetailsView 控制項	25
DataList 控制項.....	26
Repeater 控制項	28
ListView 控制項.....	29
練習 8.4 :使用 ListView 控制項	31
其他相關控制項	35

第九章: AJAX 介紹與運用

什麼是 AJAX ?	4
ASP.NET AJAX 架構	6
使用 ASP.NET AJAX 伺服器控制項	9
ScriptManager 控制項.....	10

了解網頁部分更新	11
練習 9.1 : 使用 UpdatePanel 控制項設計非同步更新	13
使用 Timer 控制項的定期執行程式碼	17
練習 9.2 : 使用 Timer 控制項的定期執行程式碼	18
使用 UpdateProgress 控制項提供進度回饋	22
練習 9.3 : 使用 UpdateProgress 控制項提供進度回饋	23
ASP.NET AJAX Control Toolkit 簡介	27
使用 ASP.NET AJAX Control Toolkit	28
練習 9.4 : 使用 AJAX Control Toolkit 中的 CalendarExtender 控制項	30

第十章: 實用控制項運用

ASP.NET 實用控制項介紹	4
MultiView 與 View	5
FileUpload	7
練習 10.1 : 使用 MultiView 與 FileUpload 控制項	9
ImageMap	14
練習 10.2 : 使用 ImageMap	16
Wizard	19
Wizard 控制項使用簡介	21
WizardStep 設計方式	23
練習 6.3 : 建立 Web.sitemap 檔案	26
ASP.NET Mobile Web 應用程式簡介	30
Moblie Web Form	31
ASP.NET Mobile 控制項簡介	33
練習 10.4 : 建立 Mobile Web 應用程式	35
ObjectList	38

第十一章: 網站錯誤客製化

網頁常見錯誤	4
執行時期錯誤處理	5
預設錯誤頁面	7
啓用除錯設定	8
本機與遠端錯誤頁面	9
例外錯誤處理流程	11
頁面層級錯誤	12
練習 11.1 : 頁面層級錯誤處理	14
應用程式層級錯誤	18
練習 11.2: 應用程式層級錯誤處理	20
自訂錯誤頁面	23
練習 11.3 : 使用 ErrorPage 設定網頁自訂錯誤	25
練習 11.4 : 應用程式層級與自訂錯誤	29
通用 HTTP 錯誤處理	33

第十二章: 網站發行與部署

部署到 IIS 7	4
應用程式集區	7

網站組態檔階層	9
使用 IIS 管理工具修改 Web.config 檔	11
練習 12.1：使用 IIS 管理工具設定組態.....	15
ASP.NET Web Site Administration Tool.....	21
部署應用程式.....	22
使用 XCopy 部署網站.....	23
使用 Copy Web Site 部署專案	24
練習 12.2：使用 Copy Web Site 工具部署	26
網站先行編譯 (Precompilation)	30
使用 Publish Web Site 部署專案.....	31
練習 12.3：使用 Publish Web Site 工具部署專案	33
建立應用程式安裝檔	37
練習 12.4：建立應用程式安裝檔	39

第十三章：網站安全性規劃

安全性名詞	4
驗證 (Authentication).....	5
Windows 驗證	7
IIS 7 Class 模型.....	8
IIS 7 Integrated 模型	10
啓用或停用 Window 驗證.....	11
在 ASP.NET 網站設定 Windows 驗證	13
練習 13.1：使用 Windows 驗證	15
Forms 驗證	19
啓用 Forms 驗證	20
撰寫登入網頁	23
練習 13.2：使用 Forms 驗證	24
使用安全管理工具	28
練習 13.3：使用安全管理工具建立使用者帳號與角色.....	29
使用安全性控制項	32
練習 13.4：建立 Login 網頁	34
使用 Create User Wizard 控制項.....	38
練習 13.5：使用 Create User Wizard 新增使用者	39

關於本課程...

本課程為 MCTS 認證的必修課程，為 ASP.NET 3.5 網站開發的入門課程，以深入淺出的教學方式介紹關於 ASP.NET 3.5 網站開發的入門與實務開發上的應用。

對象

了解 HTML 或略熟悉 DHTML，以及具基本的程式設計概念者、欲了解如何應用 ASP.NET 3.5 技術架構、程式撰寫技巧者與欲取得 MCTS、MCPD 認證的程式開發。

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

參考課程

完成本課程之後，可參考以下相關課程：

- U9544VB : ASP.NET 3.5 網站開發開發進階 (Visual Basic 2008)
- U9544C# : ASP.NET 3.5 網站開發開發進階 (Visual C# 2008)
- 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：

- 使用 ASP.NET3.5 相關技術開發功能更強大的 Web 應用程式
- 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

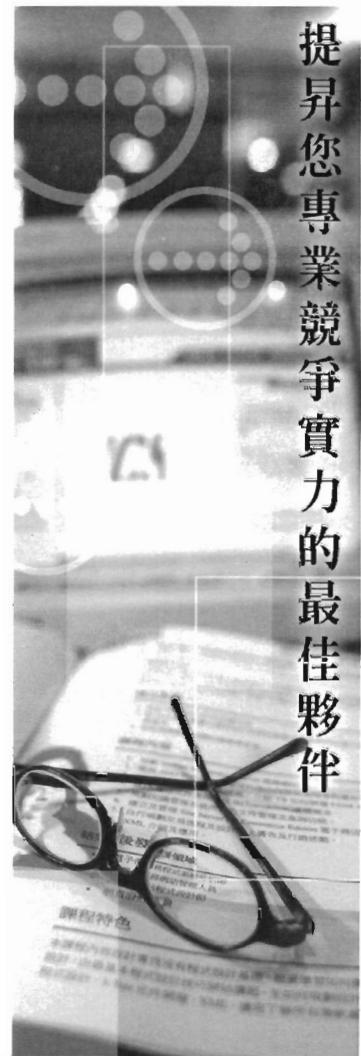
第一章: ASP.NET 3.5 與 Visual Studio 2008 介紹與運用

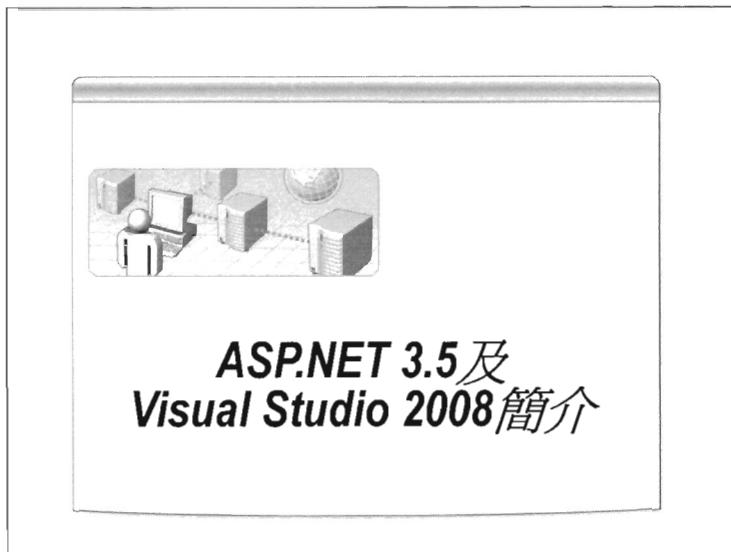
本章大綱

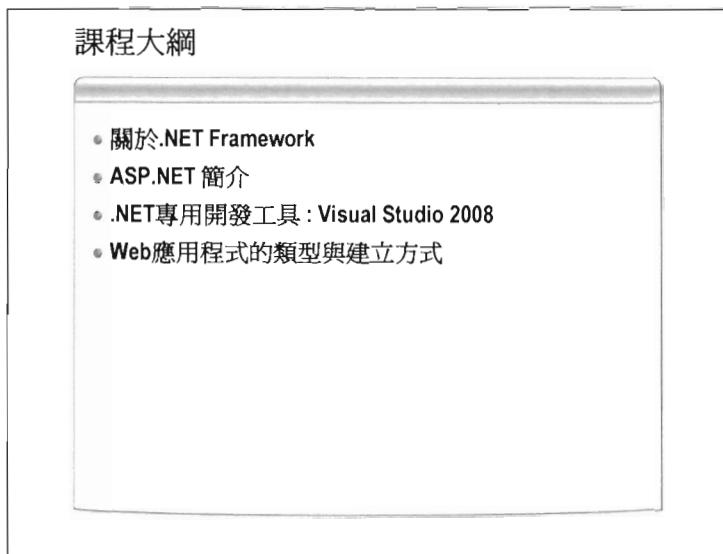
課程大綱.....	3
什麼是 Microsoft.NET ?	4
.NET Framework 的成員.....	4
.NET Framework 的優勢.....	6
.NET 的開發工具 : Visual Studio 2008	7
ASP.NET 簡介	8
什麼是 ASP.NET	9
ASP.NET Web 應用程式	11
ASP.NET 網頁執行模型.....	12
.NET 專用開發工具 : Visual Studio 2008.....	13
Visual Studio 2008 能做什麼?	14
Visual Studio 2008 的專案範本.....	16
Visual Studio 2008 開發環境介紹.....	18
Web 應用程式的類型與建立方式	25
ASP.NET Web Application	26
ASP.NET Web Site	29
網站應用程式檔案	33

作者：

周季賢





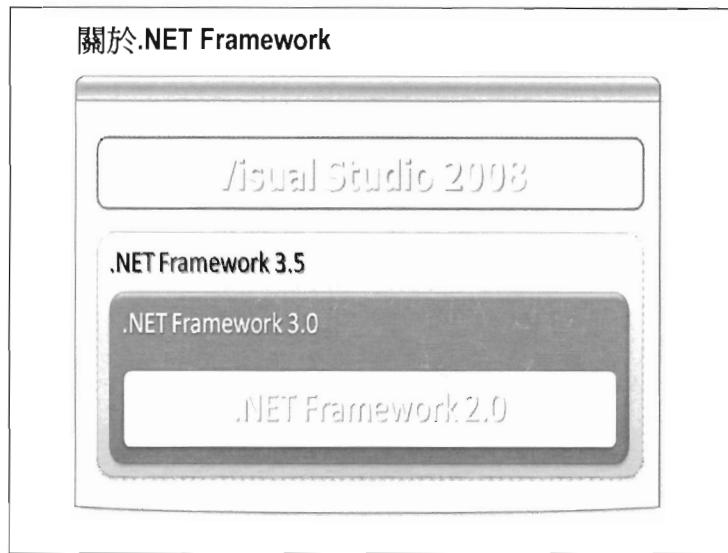


課程大綱

在這個章節中將介紹關於 ASP.NET 網站開發的基礎觀念。讓你在建立網站應用程式前，能夠對於.NET 應用程式開發與 ASP.NET 的運作架構有全面性的了解。屆時在建立 ASP.NET 網站時，便能輕鬆以對，做出正確的選擇。

本章介紹以下主題：

- 關於.NET Framework
- ASP.NET 簡介
- .NET 專用開發工具 : Visual Studio 2008
- Web 應用程式的類型與建立方式



什麼是 Microsoft.NET？

.NET，是 Microsoft 在近幾年來所推出的一個新興名詞，過去沒幾個人懂，到現在還是沒有多少人了解。微軟的促銷手法，總是以強而有力的辭彙，來吸引所有人的目光；.NET 也不例外，是微軟近幾年大力推廣的主力之一。它所代表的，其實並不是一個產品般簡單，而是一個平台的概念；在這個平台上，能夠讓各種應用程式透過 Internet 彼此通訊並共用資料，不論其作業系統或程式語言為何；而此平台的建立方式，其實僅需要安裝一個 Windows 元件即可，該元件名稱為「.NET Framework」。

.NET Framework 所能提供的是一個能夠建置，執行，與部署各式應用程式與 Web Services 的平台；它可以支援多種語言，提高產能，亦能夠整合舊有的軟體架構與新一代的應用程式與服務；同時簡化應用程式部署上的困難度以及提高 Web 應用程式操作的敏捷性。

.NET Framework 的成員

.NET Framework 兩個主要成員：Common Language Runtime 和 .NET Framework Class Library。

Common Language Runtime 是 .NET Framework 的核心元件。提供類似像記憶體管理、執行緒管理和遠端處理等核心服務，同時執行嚴格的型別安全 (Type Safety) 以及加強安全性。事實上，程式碼管理的概念是 Common Language Runtime 運作時的基本原則。以 Common Language Runtime 為目標的程式碼，被稱為 Managed 程式碼，而不以 Common Language Runtime 為目標的程式碼，則被稱為 Unmanaged 程式碼，如 COM+或 Win32 API。

.NET Framework 的另一個主要元件—.NET Framework Class Library，則是一成套應用範圍廣泛，使用物件導向開發出的大量的型別集合，您可用它來開發應用程式。而範圍從傳統命令、圖形使用者介面 (GUI) 應用程式與 ASP.NET 的最新的網頁設計為基礎的應用程式，例如 Web Form 和 XML Web Service，都包括在內。

.NET Framework 從.NET 技術推行到現在，經過了以下的版本改版：

- 1.0
- 1.1
- 2.0
- 3.0
- 3.5

乍看之下似乎分為五個版本，其實並不然。.NET Framework 1.0、1.1 和 2.0 版彼此之間是完全獨立的，而且所有版本都可出現在電腦上，而不管其他版本是否存在。當 1.0、1.1 和 2.0 版同處於一台電腦上時，每個版本都會擁有自己的 Common Language Runtime、.NET Framework Class Library、編譯器 (Compiler)、其他元件與專用的開發工具。我們在開發的時候，可以自行選擇目標平台的版本。

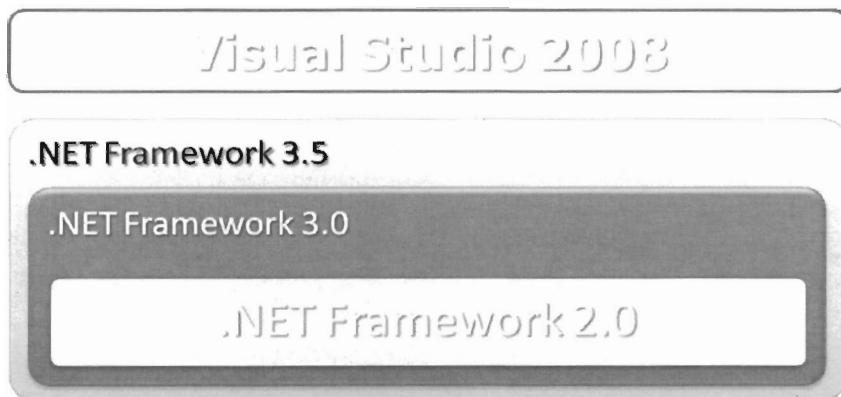
而不同於 1.0、1.1 和 2.0 的這三個版本，.NET Framework 3.5 版是以 2.0 和 3.0 版及其 Service Pack 為建置基礎；也就是說，3.0 與 3.5 並非獨立的版本，而是在以 2.0 版本為核心，再另外擴充的技術。

.NET Framework 的優勢

以下為.NET Framework 的優勢，亦為其設計目標：

- 提供可減少部署和版本控制衝突的程式碼執行環境。
- 提供程式碼執行環境的控制，加強程式碼執行時的安全性，包括未知或非完全信任之協力廠商所建立的程式碼。
- 開發各種應用程式時的一致性，例如 Windows 應用程式和 Web 應用程式。
- 資訊溝通部分，均採用業界標準，確保以.NET Framework 為基礎的程式能夠與其他程式整合。

.NET 的開發工具：Visual Studio 2008



就如同之前介紹，Visual Studio 是.NET 專用的開發工具。其開發的對象是針對.NET Framework 的版本；Visual Studio 2008 所針對的版本是比較特別的，它可以同時支援 2.0、3.0 與 3.5 三個版本的開發，至於該開發工具所能提供的有：

- 支援多種語言的開發

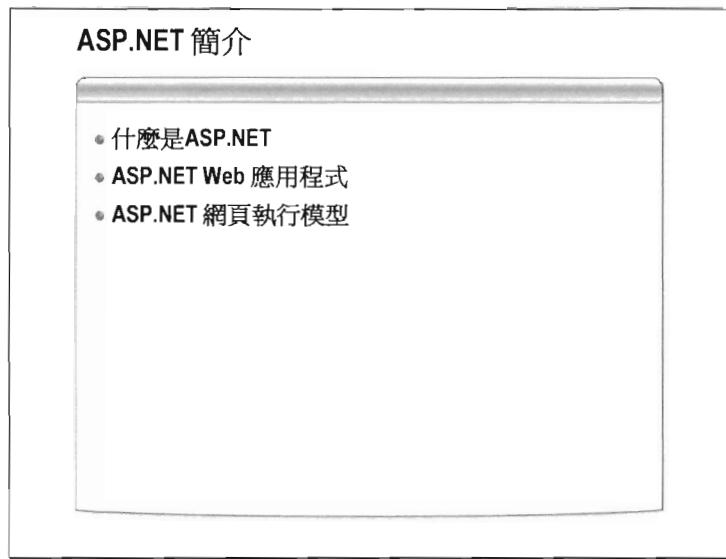
使用者可以選擇 C#、Visual Basic、C++ 和 J# 等支援.NET Framework 的類型作為開發專案的語言。

- 支援多種應用程式開發

使用者可以選擇各種類型的應用程式開發，包括了 Web 應用程式、Window 應用程式、Windows Presentation Foundation (WPF)、Windows Communications Foundation (WCF)、Windows Workflow Foundation (WF) 與 ASP.NET AJAX。

- 開發之外的支援

該工具除了能讓使用者開發各式語言的各種應用程式之外，還能提供除了開發之外的支援，如錯誤處理、除錯、說明文件查詢與應用程式的部署。



ASP.NET 簡介

在這個小節，您將認識 ASP.NET 是什麼、ASP.NET 的特性、ASP.NET 的運作方式以及 ASP.NET Web 應用程式架構。內容包括：

- 什麼是 ASP.NET
- ASP.NET Web 應用程式
- ASP.NET 網頁執行模型

什麼是ASP.NET

- .NET Framework中的Web應用程式技術
- 特性
 - 不限制瀏覽器
 - 不限制程式語言
- 技術
 - 動態網頁技術，可存取伺服器端資源
 - 網頁程式在伺服器端執行

什麼是 ASP.NET

ASP.NET 是一個整合的 Web 開發模型，是.NET Framework 的一部分，在撰寫 ASP.NET 應用程式時，您可以存取.NET Framework 中的類別。ASP.NET 也提供許多的基礎服務協助程式設計師建置企業級的網際網路應用程式。

特性

ASP.NET 具有以下特性：

- 不限制瀏覽器

ASP.NET 是一種可在 Web 伺服器上執行的程式設計架構，以動態產生和呈現 ASP.NET Web 網頁。ASP.NET 在將網頁結果呈現給提出要求的瀏覽器時，會根據各瀏覽器特性的不同，來呈現適當的網頁標籤。

- 不限制程式語言

ASP.NET 與程式語言是獨立不相干的。只要符合.NET 共通語言規範所設計出來的.NET 程式，都可以用來開發 ASP.NET 網站。這也就是說，由於底層.NET Framework 支援多種語言，因此基於.NET Framework 的 ASP.NET 也具備相同的特性，微軟及各軟體廠商針對.NET Framework 提供

許多語言編譯器，包括：Visual Basic、C#、C++、JScript、Cobol、Pascal、Perl 及 SmallTalk…等，這些編譯器都會將程式編譯為 MSIL。

ASP.NET 也強化了許多網站建置、網站管理、網站部署及安全性等議題的架構，讓開發者可以專心於規劃上，大大降低開發的時程，也能輕鬆維護網站運作。

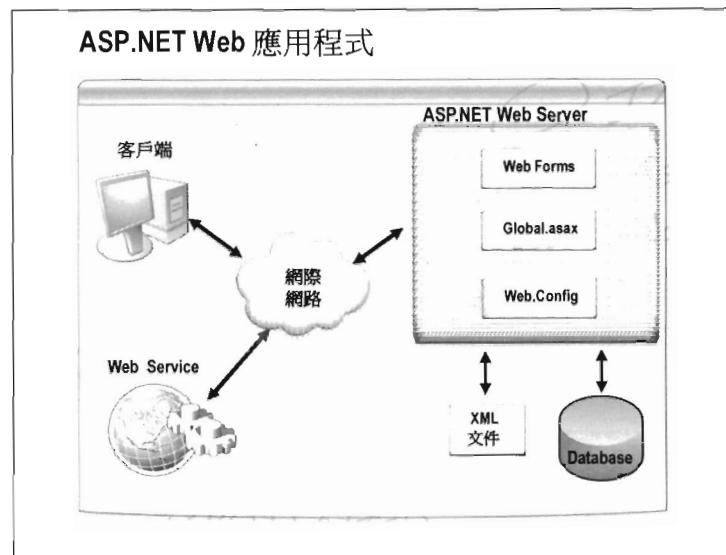
技術

- 動態網頁技術，可存取伺服器資源

ASP.NET 支援動態網頁的機制，可以輕鬆的存取 Web 伺服器上的資源，例如資料庫或是外部文件。經過編譯的程式能夠執行並產生 HTML、WML 或 XML 標籤。

- 網頁程式在伺服器端執行

由於 ASP.NET 支援動態網頁的機制，該機制其實是在每次使用者傳送資料回伺服器時，都動態的為該使用者執行編譯過的網頁程式，來產生新的網頁結果以回傳給使用者。所以所撰寫的程式語言(如 C#或 Visual Basic)，是在伺服器端執行的，與一般常見的 Java Script 並不相同。也就是因為如此，所以當網頁要與使用者互動來執行開發時撰寫的程式碼時，均需要透過控制項，將資料傳送回伺服器。



ASP.NET Web 應用程式

ASP.NET Web 應用程式包括許多不同的組成份子以及元件。要建立一個 ASP.NET Web 應用程式牽涉到使用以下的元素：

- **Web 表單(Web Forms)**

附檔名為 aspx 的網頁。這個網頁提供 Web 應用程式的使用者介面。讓客戶端從瀏覽器存取到您的網站。

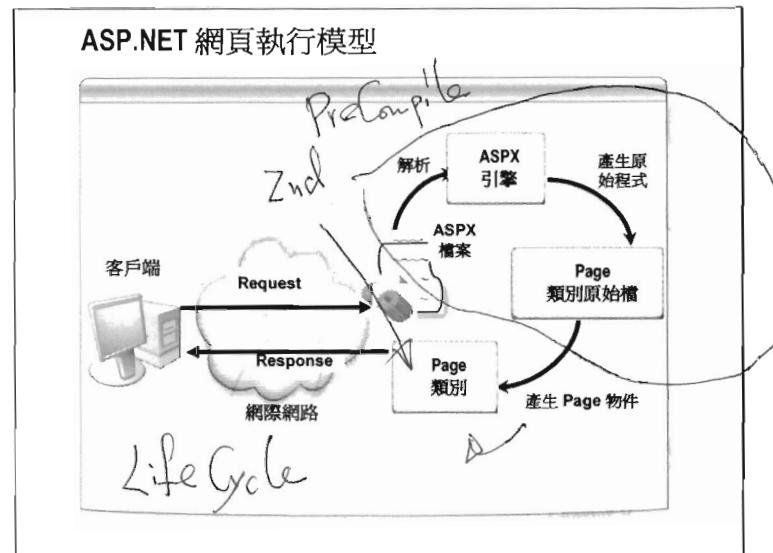
Visual Studio 2008 開發方式有兩種。第一種是將程式碼和網頁標籤都撰寫在*.aspx 檔案中；第二種是將程式碼另存到 *.aspx.cs (C#) 或 *.aspx.vb (Visual Basic) 的程式檔案 (僅以 C# 和 Visual Basic 為例)。利用宣告的語法關聯到 Web Forms，包含伺服端執行的程式碼。另外也支援 Visual Studio 2003 的開發方式。

- **組態檔(Web.config)**

XML 格式的組態檔案。定義 Web 應用程式，或 Web 伺服器…的設定。

- **Global.asax 檔案**

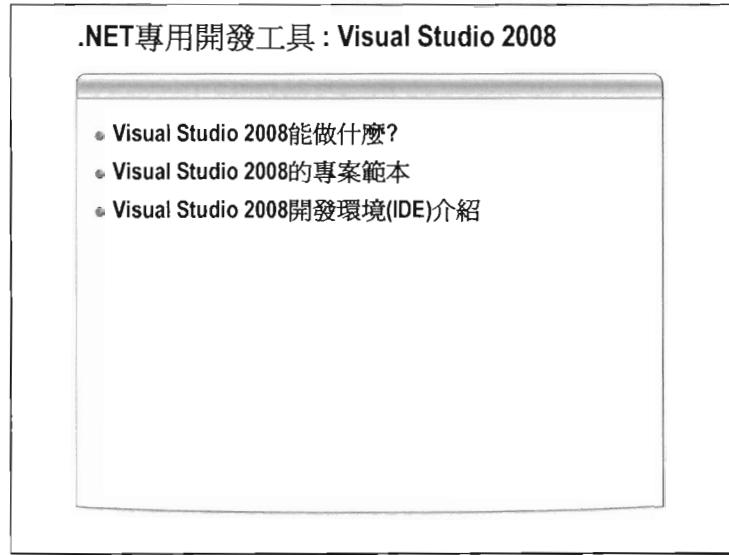
Global.asax 包含應用程式所需的事件處理程式碼。這些事件各有其觸發的時機，開發者必須先了解觸發時機才能妥善運用。



ASP.NET 網頁執行模型

ASP.NET 的網頁都須要編譯過才能夠執行。概念如下，當使用者發出一個 ASPX 網頁的要求後，ASP.NET 引擎會先將 ASPX 的網頁轉換成原始程式碼 (Source Code)，然後編譯成一個 DLL 檔案。接著將此網頁當成一個類別 (Class)，在記憶體中建立此物件的實體 (Object Instance)，並進行初始化作業。最後才將執行結果回傳到客戶端。

實際上 ASP.NET 在 2.0 版之後提出的網頁、網站編譯模型，可以允許你在開發工具事先進行網站程式的編譯動作 (Precompile)，而不必在第一次執行時再進行動態編譯。在此不詳述這些細節，若要得到更多資訊，請參考 MSDN 文件。



.NET 專用開發工具 : Visual Studio 2008

在這個小節，我們將會介紹關於 Visual Studio 2008 的相關內容；除了將 Visual Studio 2008 所能提供的功能詳盡介紹之外，還會介紹關於 Visual Studio 2008 的開發環境與專案範本，內容包括：

- Visual Studio 2008 能做什麼
- Visual Studio 2008 的專案範本
- Visual Studio 2008 開發環境(IDE)介紹

Visual Studio 2008能做什麼?

- 支援多種.NET程式語言與專案的開發
- 單一專案中能撰寫不同的程式語言
- 能夠指定.NET Framework版本來開發
- 內建網站瀏覽器與資料庫管理工具
- 具有除錯功能
- 能夠自訂開發界面

Visual Studio 2008 能做什麼?

Visual Studio 是用來建置 .NET Framework 的應用程式，如 ASP.NET Web 應用程式、XML Web Services、桌面應用程式及行動應用程式的一套完整開發工具。支援多種語言開發，並使用相同的開發環境 (IDE)，如此一來工具及視窗均可共用，並且可以簡化多重語言方案的建立。而 Visual Studio 2008 為目前最新版本的開發工具，可支援多種新型態的技術來開發應用程式。Visual Studio 2008 具有以下的功能：

- 支援多種.NET 程式語言與專案的開發

Visual Studio 2008 支援的 C#、Visual Basic 和 C++語言開發，亦支援 Web 應用程式、Window 應用程式與 ASP.NET AJAX 等應用程式的開發。

- 單一專案中能撰寫不同的程式語言

Visual Studio 能夠在一個專案中，使用不同的語言來作開發；例如 ASP.NET 的網站應用程式中，就可以在建立新的網頁(Web Form)時，挑選程式語言類型。如此便能達到不同的語言程式設計師共同開發的設計理念。

- 能夠指定.NET Framework 版本來開發

Visual Studio 前期，每個版本的 Visual Studio 都對應至某一版的.NET Framework，不過因為 2.0、3.0 與 3.5 版的特性，所以在 Visual Studio 2008 的版本中，使用者在建立或是開發應用程式時，都可自行選擇對應的.NET Framework 版本。

- **內建網站瀏覽器與資料庫管理工具**

Visual Studio 在開發 ASP.NET 網站應用程式或是資料庫應用程時之時，都可以利用 Visual Studio 內建的網站瀏覽器直接瀏覽 ASP.NET 網站應用程式最新的結果，或是使用內建的資料庫管理工具，來查詢資料庫內容或是做任何的資料異動。

- **具有除錯功能**

Visual Studio 支援在開發任何應用程式後，能夠進行單元除錯或程式除錯的功能，在 Visual Studio 2008 的版本中，也改進了以往的中斷點模組視窗、例外狀況設定等功能。

- **能夠自訂開發視窗**

Visual Studio 的功能基本上都視窗化了，使用者可自行篩選功能視窗，所有視窗都可以被移動或是關閉；如此便可讓使用者自行決定開發介面的樣式。



Visual Studio 2008 的專案範本

安裝 Visual Studio 2008 時，許多事先準備好的的專案和項目範本也會一併安裝。這些專案範本選擇視窗可以在「File」→「New」→「Project」找到，在選擇視窗中可以看到為各式語言所準備的各種應用程式範本。

這些範本為使用者提供一個開始建立新專案或擴充目前專案的起點。專案範本提供特定專案類型所需的基本檔案，包括了原始程式碼、內嵌資源、專案檔與基本的組件參考。

範本檔除了預設之外，亦可自訂增加，以下是 Visual Studio 預設的基本應用程式範本：

- **類別庫範本(Class Library Template)**

可以和其他專案共用的可重複使用的類別或元件。這種專案類型被視為無視窗，且不包含 Windows Form 類別。

- **主控台應用程式範本(Console Application Template)**

是一種在命令列執行的應用程式，並無使用者圖形介面，類似古早時期的 DOS 介面。

- **空專案範本(Empty Project Template)**

空白的專案。這個範本的任何的其他參考、檔案或元件都必須手動加入。

- **Web 控制項程式庫範本(Web Control Library Template)**

用於開發 ASP.NET Web 應用程式所需使用的控制項，類似類別庫範本。

- **Windows 應用程式範本(Windows Application Template)**

可在本機上獨立執行的 Windows 應用程式或是分散式應用程，具有使用者圖形介面。

- **Windows 控制項程式庫範本(Windows Control Template)**

用於開發 Windows Form 應用程式所需使用的控制項，類似類別庫範本。

- **Windows 服務範本(Windows Service Template)**

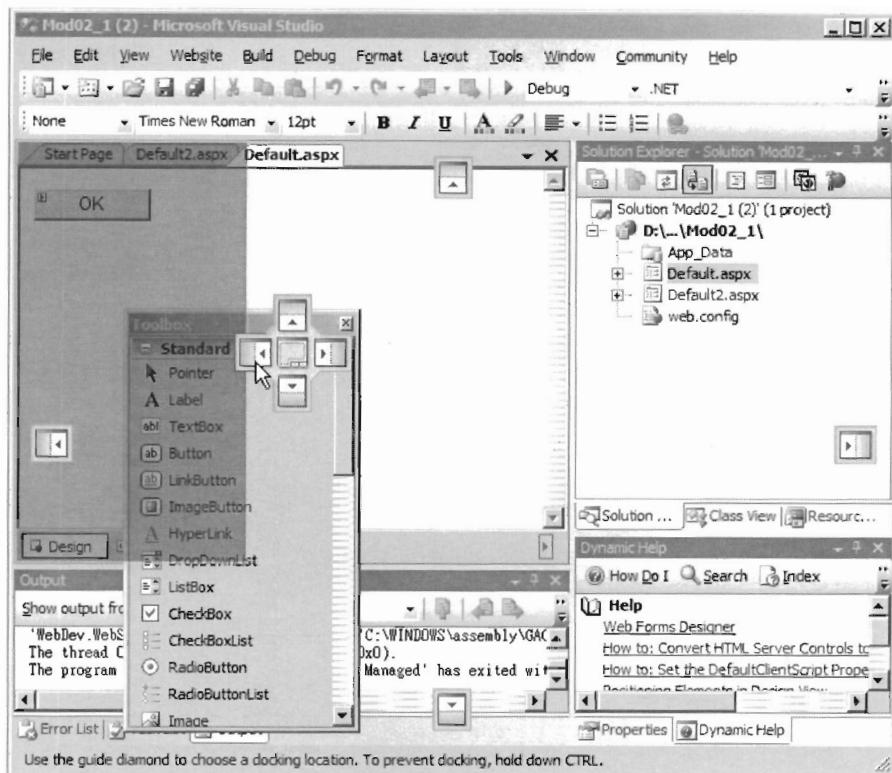
不具使用者介面的常駐型應用程式。可用來用來監視如系統效能。Windows 服務應用程式可以在電腦啓動時自動啓動，也可以暫停或重新啓動，都不會顯示任何使用者介面。這些功能非常適合在伺服器中使用，或每當需要不干擾使用同一部電腦之其他使用者的長期執行功能時使用。



Visual Studio 2008 開發環境介紹

Visual Studio 2008 整合開發環境包含了多種工具視窗以提供各類開發與設計的功能，而且每個視窗都可以切換成自動隱藏 (Auto Hide)、停駐 (Docking)、浮動視窗 (Floating)、或頁籤(Tabbed Document)的方式顯示。

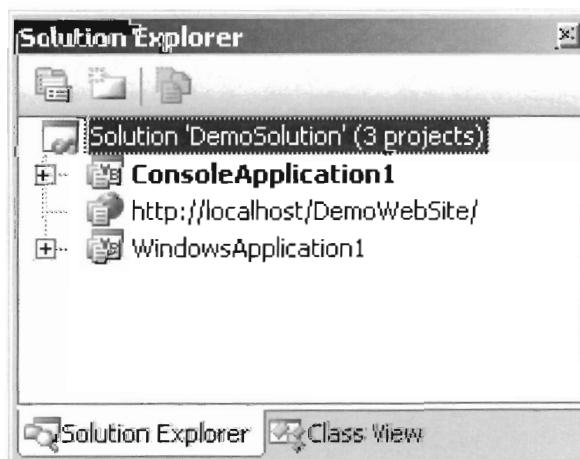
預設的情況下，各個工具視窗都是停駐在特定的位置，例如：靠在 IDE 視窗的左方或下方。當你用滑鼠左鍵按住某個視窗的標題，並進行拖曳時，便可以將視窗移動到別的位置，變成浮動視窗。你也可以將浮動視窗在拖曳回某個停駐的邊緣位置，在拖曳時，Visual Studio 2005 會顯示輔助的方向箭頭，方便你選擇要停駐的位置。只要把滑鼠移到代表該方向的箭頭，再將滑鼠左鍵放開，視窗就會停駐在指定方向的位置上。參考下圖：



以下將分別介紹整合開發環境中的幾個常用視窗，包括：方案總管 (Solution Explorer)、工具箱(Toolbox)、屬性視窗(Properties Window)與編輯視窗(Editor)等。

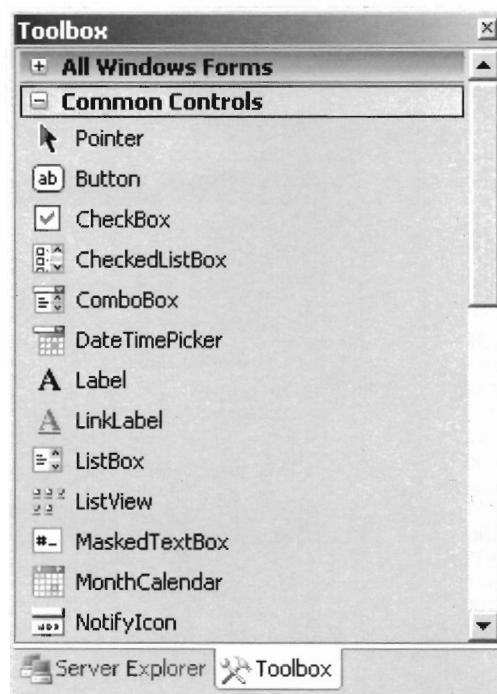
- 方案總管(Solution Explorer)

方案總管 (Solution Explorer)能夠讓您在方案或專案中檢視和執行項目管理工作。它能透過樹狀圖呈現 Solution 的內容，如 Project 與各 Project 的檔案清單。在這個視窗中，您可以對欲編輯的檔案新增、移除或刪除，也可以對 Project 新增或移除。



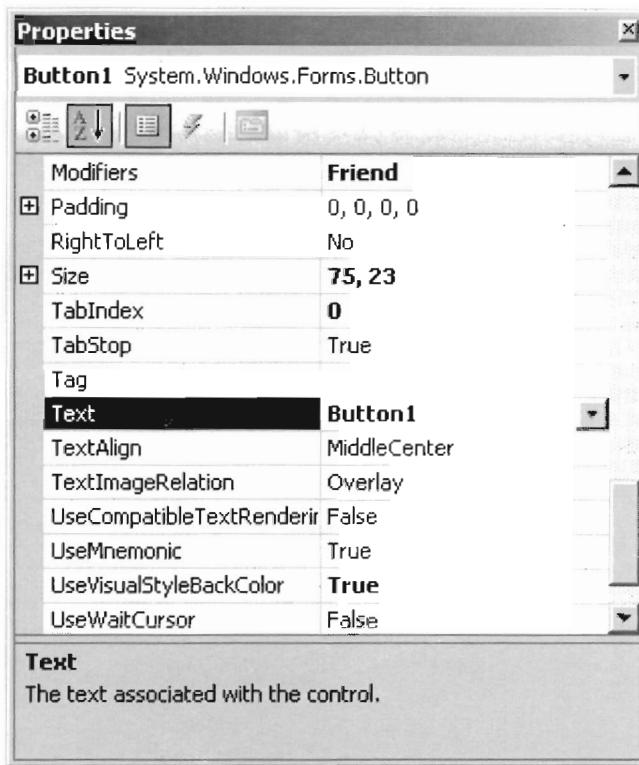
- 工具箱(Toolbox)

針對目前的設計內容，會呈現不同的顯示。在編輯使用者介面文件時，會呈現適合該文件的控制項；而編輯程式碼時，則可以當作剪貼簿。



- 屬性視窗(Properties Window)

在屬性視窗(Properties Window)，您可以檢視或變更控制項、Project 或 Solution 的屬性。特別是控制項，除了能夠調整其外觀之外，還能選擇並加入該控制項的事件處理常式。



- **編輯視窗(Editor)**

在編輯視窗(Editor)，您可以編輯程式碼或是設計使用者介面，編輯視窗(Editor)使用頁籤式設計方法，使用者可以同時交叉編輯多種文件。針對 ASP.NET 網頁的使用者介面，Visual Studio 2008 更支援了視覺化、原始碼標籤以及分割視窗的三種編輯方式。



整合開發環境中還有許多開發時的必要或輔助式窗，由於用法需搭配設計情境，所以在此不多作介紹，在後續課程中將會一一提及。

練習1.1：使用 Visual Studio 2008 IDE

•這個練習主要在於認識 Visual Studio 2008 整合開發環境的基本操作。

•預估實作時間：10分鐘

練習 1.1：使用 Visual Studio 2008 IDE

目的：

學習 Visual Studio 2008 整合開發環境的基本操作，以及建立一個 ASP.NET 網站。

功能描述：

開啓 Visual Studio 2008，並且建立一個 ASP.NET 網站應用程式。
檢視各個視窗，包括：Solution Explorer、Toolbox、Properties、
Document 視窗的 Design View 與 Source View 等等。

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「\U9543\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. 檢視方案總管中的項目，並使用檔案總管檢視網站所在的目錄中有哪些檔案及目錄。
4. 切換「Toolbox」視窗的自動隱藏顯示模式，以及將視窗拖曳到別的位置成為浮動視窗，再拖曳回原來的位置停駐。
5. 將「Document」視窗切換至「Design View」，從「Toolbox」裡面將一個 Button 控制項拖曳到網頁上面。
6. 在 Button1 上面點一下滑鼠左鍵以選取此控制項，觀察屬性視窗，並設定按鈕控制項的 Text 屬性為「Ok」。注意修改過的屬性值會以粗黑體顯示。
7. 練習編輯器的 IntelliSense 功能。將「Document」視窗切換至 Source View，觀察網頁的原始碼。然後找到「asp:Button」標籤，在標籤中加入以下程式碼：

```
BackColor="Yellow"
```

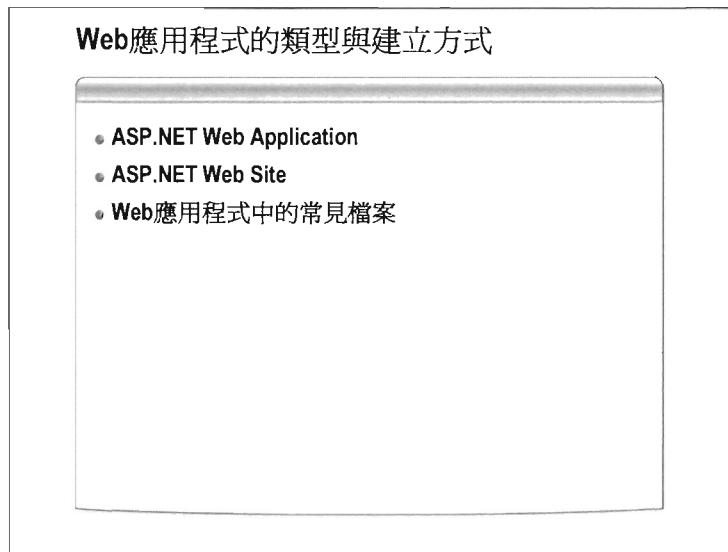
整段標籤看起來會像這樣：

```
<asp:Button ID="Button1" BackColor="Yellow" runat="server"
Text="Button" />
```

8. 在方案總管中的 Default.aspx 項目上點右鍵，選擇「View in Browser」，以檢視網頁實際執行的結果。參考下圖：



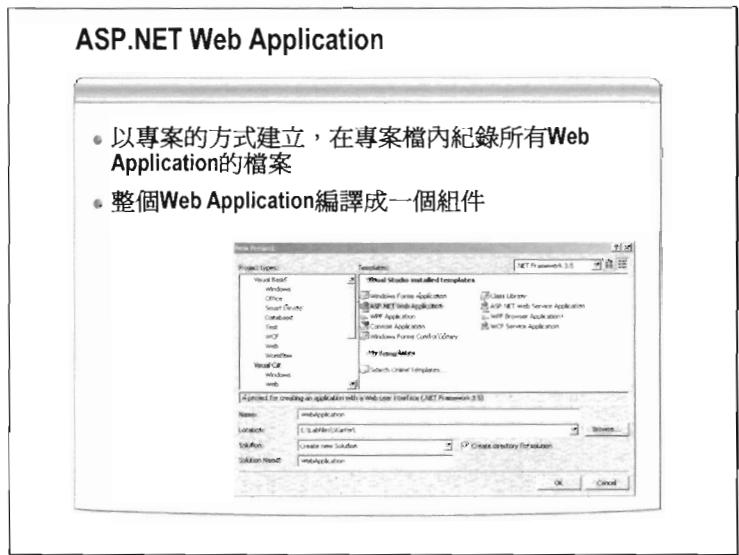
9. 儲存所有檔案。從主選單點選「File」→「Save All」。
10. 關閉方案。從主選單點選「File」→「Close Solution」。
11. 關閉 Visual Studio 2008。你可以從主選單點選「File」→「Exit」，或者直接點視窗右上角的關閉鈕。



Web 應用程式的類型與建立方式

在這個小節，我們將會介紹關於 ASP.NET 的 Web 應用程式建立方式，除了將 Web 應用程式的種類詳盡介紹之外，還會介紹關於 Web 應用程式建立後所產生的檔案，內容包括：

- ASP.NET Web Application
- ASP.NET Web Site
- Web 應用程式中的常見檔案



ASP.NET Web Application

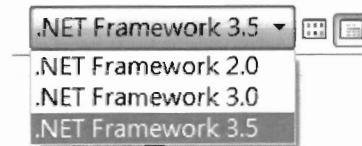
ASP.NET Web Application 為 Visual Studio 2008 替代的專案類型，ASP.NET Web Application 模型會必須明確地參考方案專案檔的檔案才是該專案的一部分。

ASP.NET Web Application 模型所使用的專案、組建和編譯語意與都與 Visual Studio .NET 2003 的 Web 專案相同，在下列的情況下，您可以使用 ASP.NET Web Application 模型開發網站：

- 將 Visual Studio .NET 2003 應用程式移轉到 Visual Studio 2008。
- 自行決定編譯後組件的名稱。
- 使用多個 Web 專案來建置 Web 應用程式。

欲建立 ASP.NET Web Application，必須要在 Visual Studio 2008 的選單內，選擇「File」→「New」→「Project」，在專案範本視窗內選擇 ASP.NET Web Application，挑選.NET Framework 目標版本，最後再決定專案資料夾位置即可。

特別需注意的是挑選.NET Framework 目標版本時，由於 Visual Studio 2008 支援 2.0、3.0 與 3.5 三種版本的.NET Framework，所以在建立專案或網站時，都會在範本視窗右上方見到如下圖的選項：



決定目標版本除了會影響屆時開發網頁所能提供的功能(如：元件或控制項)之外，也會影響 Web.config 檔案中的設定內容。決定之後，還能在專案或網站的屬性頁中再次調整。

練習1.2：建立 ASP.NET Web Application

•這個練習主要在於學習如何建立 ASP.NET Web Application 以及自行控制編譯後組件名稱。

•預估實作時間：10分鐘

練習 1.2：建立 ASP.NET Web Application 網站

目的：

學習建立 ASP.NET Web Application 網站以及自行控制編譯後組件名稱。

功能描述：

在本機建立一個 ASP.NET Web Application 網站，檢視網站內容並嘗試自行修改編譯後組件名稱。

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 Visual Studio 2008 的主選單點選「File」→「New」→「Project」，接著會開啓「New Project」對話窗。
3. 在對話窗裡進行以下設定：

- 在「Project Type」區塊中選擇 Visual C# 或 Visual Basic。
 - 在「Templates」區塊中選擇「ASP.NET Web Application」。
 - 在視窗右上方選擇.NET Framework 的目標版本為「.NET Framework 3.5」。
 - 「Name」部分選擇 WebApplication1。
 - 「Location」選擇「\U9543\Practices\VB 或 CS\Mod01_2\Starter」。
 - 「Solution Name」維持為 WebApplication1。
 - 按「OK」鈕。
4. 網站建立成功後，開啓「\U9543\Practices\VB 或 CS\Mod01_2\Starter\ WebApplication1」，應可在該資料夾內找到 ASP.NET Web Application 網站。
5. 接著從 Visual Studio 2008 的主選單點選「Build」→「Build WebApplication1」。
6. 建置成功之後，開啓「\U9543\Practices\VB 或 CS\Mod01_2\Starter\ WebApplication1\bin」，應可在該資料夾內找到 WebApplication1.dll 檔案。
7. 從「Solution Explorer」視窗中對 WebApplication1 網站根目錄按下滑鼠右鍵，選擇「Property」，開啓專案屬性頁。
8. 在專案屬性頁中選擇「Application」頁籤，修改頁籤內的「Assembly Name」為「myAssembly」。
9. 完成修改之後，在一次從 Visual Studio 2008 的主選單點選「Build」→「Build WebApplication1」。

10. 建置成功之後，開啓「\U9543\Practices\VB 或 CS\Mod01_2\Starter\ WebApplication\bin」，應可在該資料夾內發現 dll 檔案被更名為 myAssembly.dll。



ASP.NET Web Site

ASP.NET Web Site 為 Visual Studio 2008 預設的專案類型，ASP.NET Web Site 模型會使用檔案目錄結構來定義專案內容。這個模型中沒有專案檔，而目錄中的所有檔案都是專案的一部分

使用 ASP.NET Web Site 開發時，該專案模型會辨認特定類型內容的某些資料夾名稱，如公用程式類別的程式碼，就必須要放在根目錄下的 App_Code 目錄內。

而在編譯的部分，ASP.NET Web Site 網站模型可以將不同的程式碼檔案分別編譯成個別的組件。

欲建立 ASP.NET Web Site，必須要在 Visual Studio 2008 的選單內，選擇『File』→『New』→『Web Site』，在專案範本視窗內選擇 ASP.NET Web Site，挑選.NET Framework 目標版本，最後再決定網站建立方式即可。

而 ASP.NET Web Site 支援四種建立網站的方式：

- File System

將網站建立在任何目錄下，包括本機的目錄和任何遠端的分享磁碟機。當你使用這種方式建立網站時，本機不需要 IIS，在開發測試時使用的是 Visual Studio 內建的網站伺服器。

- Local IIS :

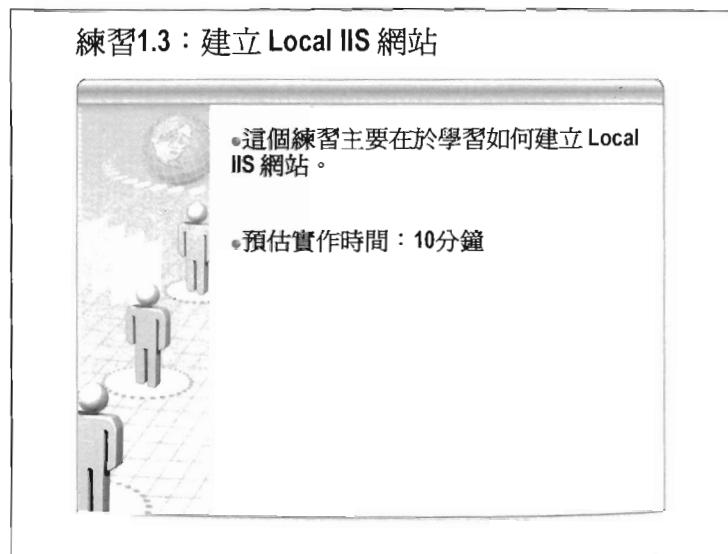
將網站建立在本機電腦的 IIS 網站目錄下。當你選擇以此位置建立 ASP.NET 網站時，網站的檔案會建立在 IIS 網站的資料夾底下，預設為 C:\Inetpub\wwwroot。

- FTP Site :

以 FTP 的方式建立及管理遠端的網站。這種方式通常用在當你的網站位於別的企業時，例如：將網站交給其他網際網路服務公司託管。

- Remote Site :

將網站建立在遠端機器的 IIS 虛擬目錄中。這種方式適用於將網站建立在企業內部網路的其他伺服器上，而且該伺服器必須安裝 IIS 與 FrontPage Server Extensions。



練習 1.3 : 建立 Local IIS 網站

目的：

學習以 Local IIS 的方式建立 ASP.NET 網站。

功能描述：

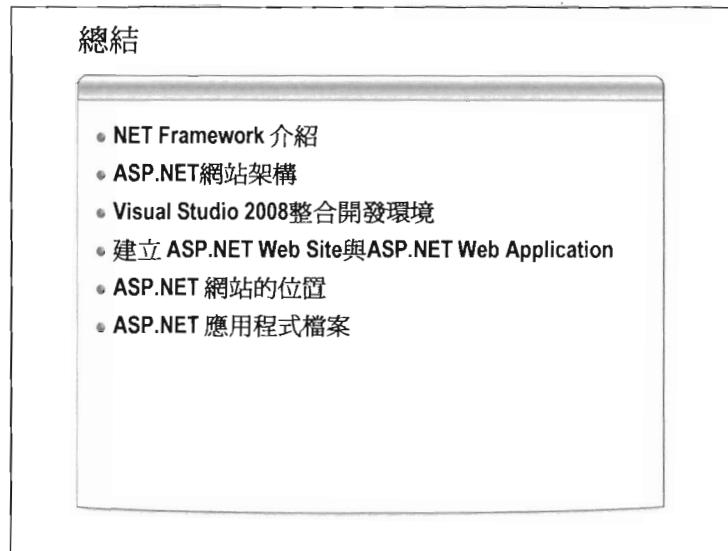
在本機的 IIS 中建立一個 ASP.NET 網站，並檢視 IIS 虛擬目錄。

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 Visual Studio 2008 的主選單點選「File」→「New」→「Web
Site」，接著會開啓「New Web Site」對話窗。
3. 在對話窗裡進行以下設定：
 - 在「Templates」區塊中選擇「ASP.NET Web Site」。

- 在視窗右上方選擇.NET Framework 的目標版本為「.NET Framework 3.5」。
 - 「Location」選擇「HTTP」，並且在右邊的文字盒輸入網址：<http://localhost/LocalWebSite>。
 - 「Language」選 Visual C# 或 Visual Basic。
 - 按「OK」鈕。
4. 網站建立成功後，開啓檔案總管檢視
C:\Inetpub\wwwroot\LocalWebSite 目錄。
5. 開啓 IIS 管理員檢視剛才建立的網站。做法為：點選「Start」→「Administrative Tools」→「Internet Information Services (IIS) Manager」，然後在左邊面版中展開樹狀節點：「local computer」→「Web Sites」→「Default Web Sites」，應該可以看到剛才建立的 LocalWebSite 網站。



本章節介紹了關於.NET Framework 與 Visual Studio2008 的相關內容。.NET Framework 2.0、3.0 與 3.5 各版本的主要技術如下：

以下是 2.0 主要技術：

- Common Language Runtime(CLR)和 Base Class Library(BCL)。
- 泛型型別和方法。
- C#、Visual Basic、C++ 和 J# 的編譯器。
- ADO.NET。
- ASP.NET。
- Windows Form。
- Web Services。

以下是 3.0 主要技術：

- Windows Presentation Foundation (WPF)。
- Windows Communications Foundation (WCF)。
- Windows Workflow Foundation (WF)。

以下是 3.5 主要技術：

- Language Integrated Query (LINQ)。
- C#、Visual Basic 和 C++ 的新編譯器。
- ASP.NET AJAX。

Visual Studio2008 所提供的支援的額外功能如下：

- 支援 Web Services、AJAX 與 LINQ

由於.NET Framework 以至 3.5 版，所以該版本之下的 ASP.NET 理所當然的能夠使用各式舊有的(Web Services)或是最新的技術(AJAX 與 LINQ)來開發網頁應用程式，使用 Web Services 可以輕鬆的達到資料溝通的目的，ASP.NET AJAX 能夠輕鬆的開發出互動性高的網頁應用程式，而 LINQ 則能對於各式資源，都能以一致化的方式存取。

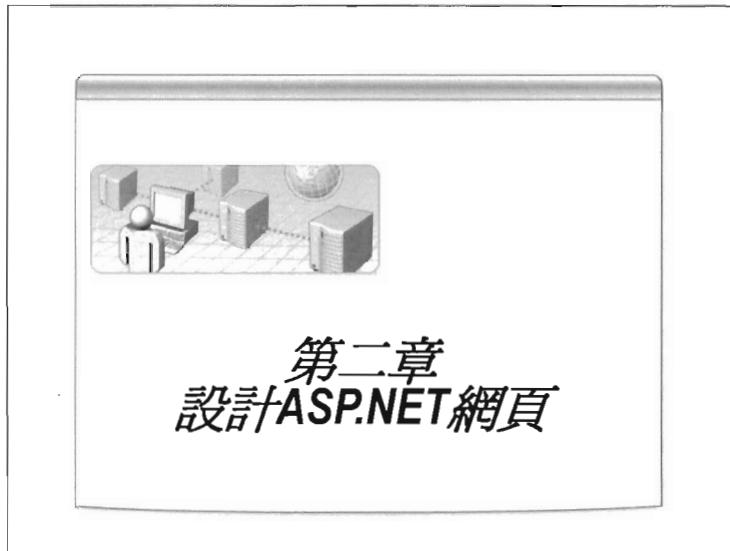
第二章: 設計 ASP.NET 網頁

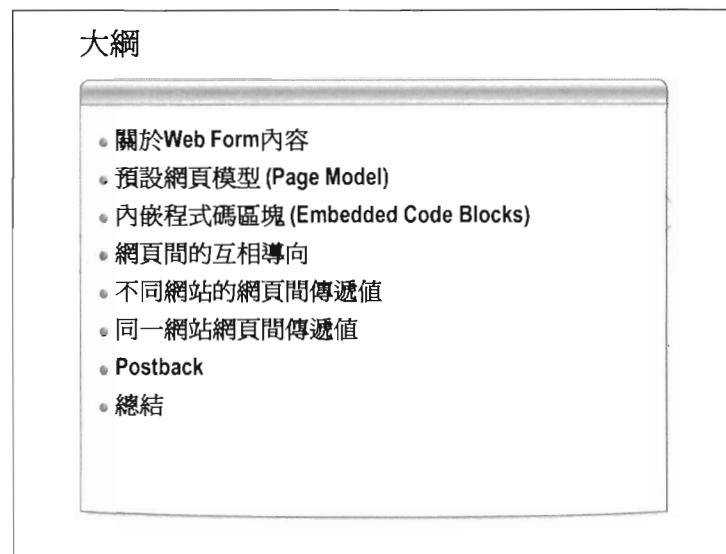
本章大綱

課程大綱.....	3
Web Form 的組成.....	4
預設網頁模型(Page Model)	7
內嵌程式碼區塊(Embedded Code Blocks).....	11
網頁間的互相導向	13
練習 2.1 : ASP.NET 的網頁導向	16
不同網站的網頁間傳遞值	20
同一網站網頁間傳遞值.....	22
練習 2.2 : 在 ASP.NET Web 網頁之間傳遞值.....	25
Postback.....	30
練習 2.3 : 了解 Postback.....	32

作者：
周季賢





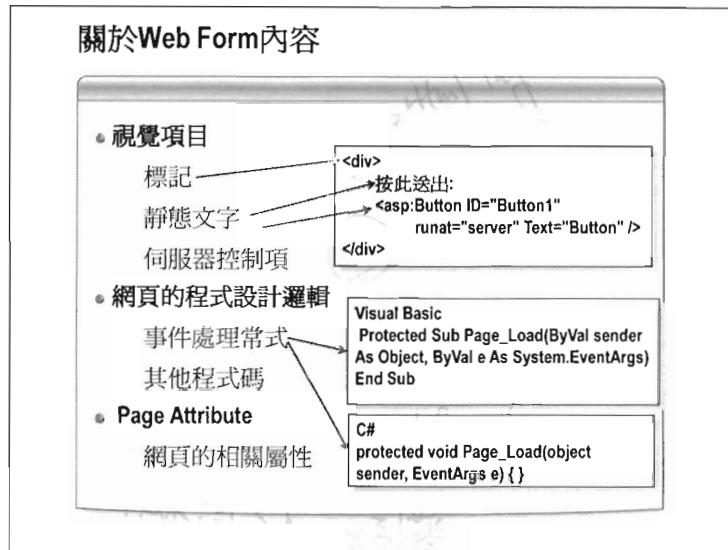


課程大綱

在這個章節中將介紹關於 ASP.NET Web 應用程式 Web Form 的相關內容。主要介紹撰寫方式，內容辨識與網頁之間的導向。

本章介紹以下主題：

- 關於 Web Form 內容
- 預設網頁模型 (Page Model)
- 內嵌程式碼區塊 (Embedded Code Blocks)
- 網頁間的互相導向
- 不同網站的網頁間傳遞值
- 同一網站網頁間傳遞值
- Postback



Web Form 的組成

一個 Web Form 指的就是一個 ASP.NET 的網頁，簡單的來說，在網站中，每一個附檔名為 ASPX 的檔案，就是一個 Web Form。Web Form (Web 表單) 被當成是一個容器，上面可以放置各式各樣展示使用者介面的控制項，包含了 HTML、程式碼和控制項等，而且執行在 Web 伺服器上。

視覺項目

ASP.NET Web Form 中，由視覺項目來決定使用者屆時看到的網頁外觀，而視覺項目包含了以下幾個種類：

- 標記

就是 Html 標籤，如`<html>`、`<head>`與`<body>`等標籤，這些標籤是靜態的，就是當撰寫完畢之後，就不會再改變其顯示方式。

- 靜態文字

即為網頁中的純文字，如果僅需要顯示文字內容給使用者，不需要增加任何顯示效果的話，在適當的位置直接鍵盤輸入文字即可。

- 伺服器控制項

也是一種標籤，如下所示：

```
<asp:Label id="Label1" runat="server" Text="Label" >
```

與靜態的 Html 標籤不同的是，該標籤在執行時期時，會動態的轉變成 Html 標籤來顯示，詳細內容將會在第三章介紹。

網頁的程式設計邏輯

ASP.NET Web Form 中，由程式設計邏輯來決定使用者與網頁互動時的效果與處理方式，程式設計邏輯基本上就是 C# 或 Visual Basic 程式碼，而程式設計邏輯包含了以下幾個種類：

- 事件處理常式

事件處理常式就是當需要針對某些控制項的事件來撰寫程式時，開發工具自動或使用者手動的建立出來的一個程式碼區塊，該程式碼區塊內的程式，會在控制項的事件發生當下執行。如下所示：

```
Visual Basic
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    End Sub
```

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

- 其他程式碼

其他程式碼指的是事件處理常式之外的程式，如宣告類別成員或是自訂函式等等。如下所示：

```
Visual Basic
Private item As Integer
Private Function sayHello() As String
    Return "HELLO"
End Function
```

```
C#
private int item;
private string sayHello()
{
    return "HELLO";
}
```

Page Attribute

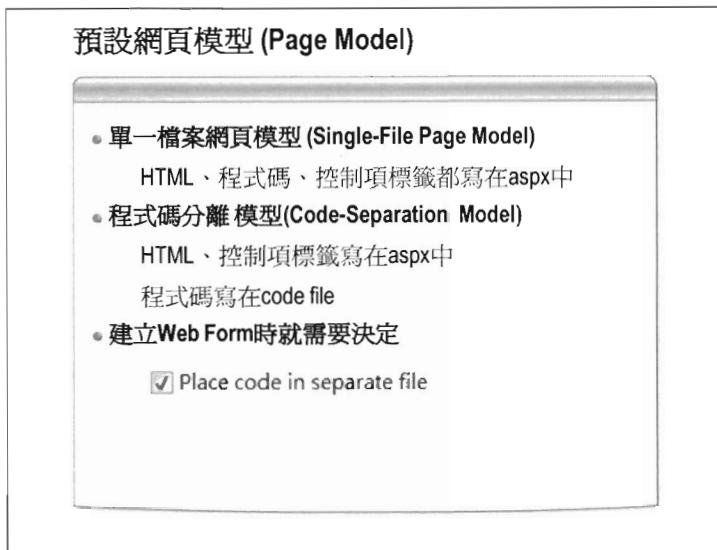
一份正常的 Web Form，其 aspx 文件的最上方一定會有 Page 指示詞，Web Form 的特性可以藉由 Page attribute 進行宣告。Page attribute 描述了 Page 等級專屬的特性 (Attribute) 以提供 ASP.NET 網頁的剖析器 (Parser) 和編譯器處理的時候使用。一個網頁之中只能有一份 Page attribute，以下是 Visual Basic 與 C# 的 Page attribute 標籤範例：

Visual Basic <%@ page language="VB" %>
--

C# <%@ page language="C#" %>
--

而 Page attribute 標籤內有各式各樣的屬性值，以下介紹幾個常見的屬性值：

- **Language**
描述網頁撰寫時所使用的程式語言。如設定為 vb 代表使用 Visual Basic；若為 C# 則可以使用 c# 或 cs。
- **AutoEventWireup**
表示網頁的事件是否依照預設命名自動辨識。預設為 true。
- **CodeFile**
描述 Code-Separation 檔案的檔名。提供 ASP.NET 3.5 版使用。ASP.NET 1.x 使用的 Codebehind Attribute 在 Visual Studio 2008 開發工具中已不再支援。
- **Inherits**
參考至 Code-Separation 檔案內的類別名稱



預設網頁模型 (Page Model)

ASP.NET 在 2.0 的版本之後，提供了 2 種預設的網頁設計模型，包含：

- 單一檔案網頁模型 (Single-File Page Model)
- 程式碼分離模型 (Code-Separation Model)

這兩種模型的功能是相同的，並且您可以在兩個模型上使用相同的控制項和程式碼。

單一檔案網頁模型 (Single-File Page Model)

前文提過，一份 Web Form 中，基本應該具備的就是「視覺化項目」與「程式設計邏輯」，單一檔案網頁模型 (Single-File Page Model)，指的是將「視覺化項目」與「程式設計邏輯」撰寫在同一個.aspx 檔案之中。在程式執行時期，.aspx 檔會繼承自 Page 類別，產生一個新的類別，以執行網頁中的程式碼。

Single-File Page Model 的「程式設計邏輯」是放在.aspx 檔案中的以下區段內：

```
<script runat="server">
...
</script>
```

而單一檔案網頁模型 (Single-File Page Model) 的 Page attribute 預設則是如下所示：

Visual Basic
<%@ page language="VB" %>

C#
<%@ page language="C#" %>

可以明顯的看出來，因為所有內容都集中在 aspx 檔案內，所以也不需要再另行註記 CodeFile 與合併類別的名稱。

程式碼分離模型 (Code-Separation Model)

Visual Studio 2008 開發工具預設的設計模型。能夠讓您將「視覺化項目」與「程式設計邏輯」分離在不同檔案，以分別進行維護，這種機制的設計方式如下：

- aspx 檔案中包含了「視覺化項目」與 Page attribute，而 aspx 中的 Page attribute 預設如下所示：

Visual Basic
<%@ page language="VB" CodeFile="MyCodeBehind.aspx.vb"
Inherits="MyCodeBehind" %>

C#
<%@ page language="C#" CodeFile="MyCodeBehind.aspx.cs" In
herits="MyCodeBehind" %>

與單一檔案網頁模型 (Single-File Page Model) 不同的是，該模型會使用「CodeFile」與「Inherits」標籤來註記 CodeFile 與合併類別的名稱。

- 「程式設計邏輯」則放在稱之為「CodeFile」的 .vb (Visual Basic) 或 .cs (C#) 檔案之中。檔案之中會有一個預設的類別，

而此類別名稱則需要對應到 Page attribute 的「Inherits」屬性。例如：

```
Visual Basic
Imports Microsoft.VisualBasic
Partial Class MyCodeBehind
Inherits System.Web.UI.Page
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Label1.Text = "Hello " & TextBox1.Text
End Sub
End Class
```

```
C#
using System;
public partial class MyCodeBehind : System.Web.UI.Page
{
    protected void Button1_Click(object sender, EventArgs e)
    {
        Label1.Text = "Hello " + TextBox1.Text;
    }
}
```

特別需注意的是該類別使用 `partial` 關鍵字宣告，表示類別並非完全包含在單一檔案中。而是在網頁執行時，編譯器會讀取 `.aspx` 檔案中的內容與此類別、將其組成單一類別，然後編譯為單一類別的單元。

哪種模型比較好？

單一檔案網頁模型和程式碼後置網頁模型的功能完全相同。在執行的時候，都以相同方式執行，沒有效能好壞的差異。你可以根據應用程式的結構，或想把介面和程式碼分離的喜好自由地選擇。

單一檔案網頁模型好處：

- 程式碼較易閱讀。
- 較易部署。
- 沒有相依檔，容易改名。
- 較易搭配原始程式碼控管系統使用。

程式碼後置網頁模型好處：

- 程式碼和標籤分離，較不會互相影響。

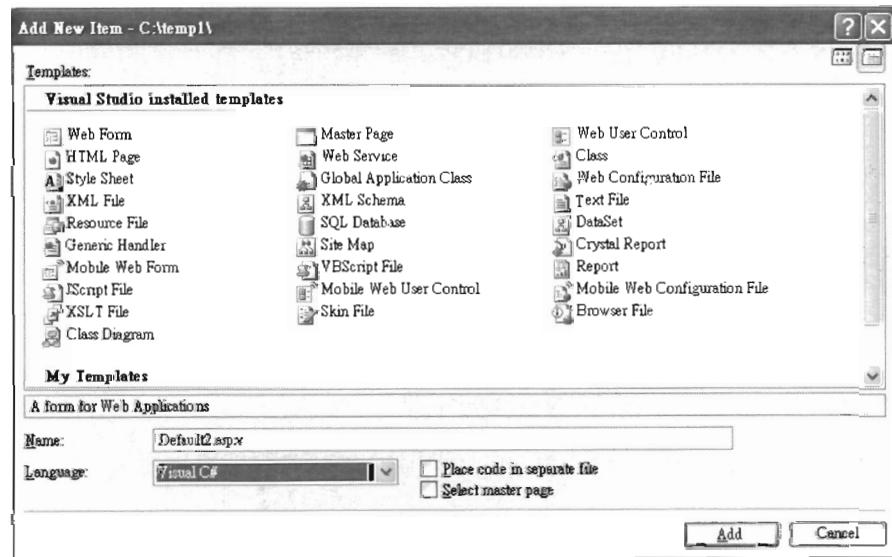
- 程式碼可以讓多個網頁重複使用。
- UI 部份可以交由美工進行設計，不會不小心刪除程式師所設計的程式碼，因為程式碼是獨立在另一個檔案之中。

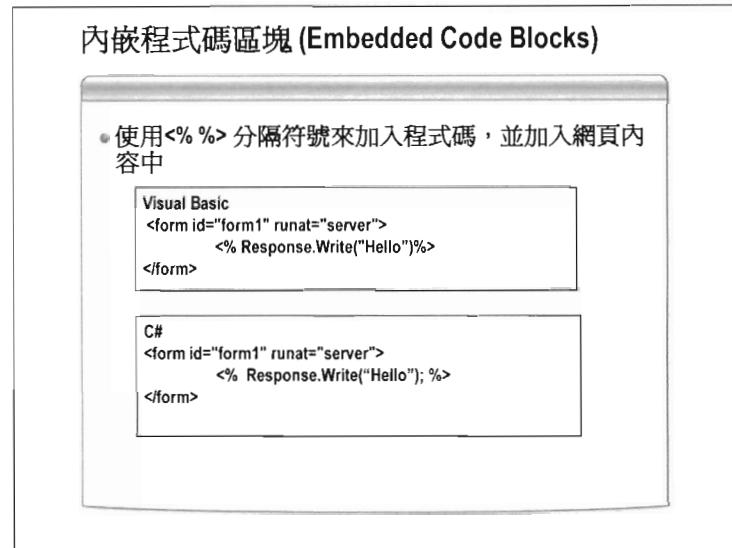
建立 Web Form 時就需要決定

那麼，決定好網頁模型之後，該如何去建立呢？其實決定設計模型的關鍵就在建立 Web Form 的對話視窗內。

新增一個 ASPX 網頁時，在「Add New Item」的對話盒中：

- 勾選「Place code in separate file」核取方塊，則建立程式碼分離模型 (Code-Separation Model)的網頁；
- 沒有勾選「Place code in separate file」核取方塊，則建立單一檔案網頁模型 (Single-File Page Model)的網頁。





內嵌程式碼區塊(Embedded Code Blocks)

除了將使用 ASP.NET 所提供的預設網頁模型，您還可以有另一個選擇，就是使用內嵌的程式碼區塊，直接將程式碼內嵌在網頁中。

該如何撰寫內嵌的程式碼區塊呢？其實就是利用<% %>的符號文字，在其內加入 Visual Basic 與 C#的程式碼，並放置到網頁內。以下介紹幾種方式：

- 直接撰寫程式碼，並使之執行：

這種方法與一般的程式碼撰寫法相同，不同的只有撰寫的位置變成<% %>標籤內，如下所示：

```
Visual Basic
<div>
    現在時間:<% Response.Write(DateTime.Now.ToString("t"))%>
</div>
```

```
C#
<div>
    現在時間:<% Response.Write(DateTime.Now.ToString("t")); %>
</div>
```

- 呼叫方法，直接取得其結果並輸出：

這種方法可以簡化呼叫方法後還需自行輸出的程式碼，也可呼叫自行撰寫的方法，如下所示：

```
Visual Basic
<script runat="server">
    Private Function sayHello() As String
        Return "HELLO!!!!"
    End Function
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <% =sayHello() %>
        </div>
    </form>
</body>
</html>
```

```
C#
<script runat="server">
    private string sayHello()
    { return "HELLO!!!!"; }
</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <% =sayHello() %>
        </div>
    </form>
</body>
</html>
```

雖然 ASP.NET 支援內嵌的程式碼區塊撰寫方式，但是其實是為了保留與舊版 ASP 技術的相容性。一般而言，當程式邏輯較為複雜時，使用內嵌的程式碼區塊的方式來撰寫程式並不是很恰當，因這種撰寫方式其實很難偵錯和維護。不過對於簡易的網頁設計來說，這種設計方式是相當直覺的。

網頁間的互相導向

- **超連結**
 使用超連結控制項
- **瀏覽器重新導向**
 Response.Redirect
- **跨網頁 POST**
 使用控制項PostBackURL屬性
- **伺服器重新導向**
 Server.Transfer

網頁間的互相導向

在開發 Web 應用程式時，經常需要因為使用者的行為，而必須讓使用者由原本的網頁轉而檢視另一張網頁內容。例如：在查詢頁決定條件之後，要將使用者導向至結果頁面來檢視查詢結果。而 ASP.NET 因應不同的需要，提供了以下四種導向方法：

超連結

超連結即為一般網頁內容上常見，由 HTML 的[標籤所建立出的靜態連結。超連結可在\[標籤內容中指定識別文字與導向的 URL；而使用者屆時在網頁上就會看到識別文字，點選該文字之後，瀏覽器便會導向至標籤中指定的 URL。標籤如下所示：\]\(#\)](#)

```
<a id="HyperLink1" href="Target.aspx">目標網頁</a>
```

建立超連結標籤亦可使用 ASP.NET 所提供的 HyperLink 控制項，該控制項能在執行時期產生[標籤的內容。](#)

瀏覽器重新導向(Response.Redirect)

這種方式是利用程式碼將命令傳送到使用者的瀏覽器，請瀏覽器導向至目的網頁。這種方式就好比是用程式設計方式按一下超連結。較為不同的是，因為是由程式碼控制，所以可以輕鬆的變動導向的 URL。

程式碼如下所示：

```
Visual Basic
Response.Redirect("~/Target.aspx")
```

```
C#
Response.Redirect("~/Target.aspx");
```

→ 並非程式或系統路徑的根目錄

跨網頁 POST

預設，ASP.NET 的按鈕控制項在按下之後，會做一個稱之為 PostBack 的行為，該行為會對按鈕所屬的網頁再做一次的 Post；而跨網頁 Post 可讓按鈕在按下之後，Post 到不同的網頁。

跨網頁 Post 也很類似超連結，但不同的是使用 HTTP POST 的方式來導向至目的網頁，使用 HTTP POST 的方式會將來源網頁上相關資訊送到目的網頁(如控制項的值)。

設計方式很簡單，只需要設定按鈕的 PostBackUrl 屬性，在該屬性填上目的網頁的網址及可，如下所示：

```
<asp:Button ID="Button2" runat="server" Text="Button"
PostBackUrl "~/Target.aspx" />
```

伺服器重新導向(Server.Transfer)

該方法與瀏覽器重新導向非常相似，但是最大的不同在於該方法的導向行為是在伺服器端發生的。所以在該行為之下，目的網頁也只能限制於相同的 Web 應用程式中。

由於導向行為是在伺服器端發生的，因此瀏覽器端是無法得知該變動，所以瀏覽器端會保留原始(來源)URL 的相關資訊。例如，Internet Explorer 中的「網址」方塊不會在傳輸後變更，而是會繼續顯示來

源網頁的 URL。因此，當需要隱藏特定網頁的 URL 時，使用伺服器導向(Server.Transfer) 方法是最好的策略。

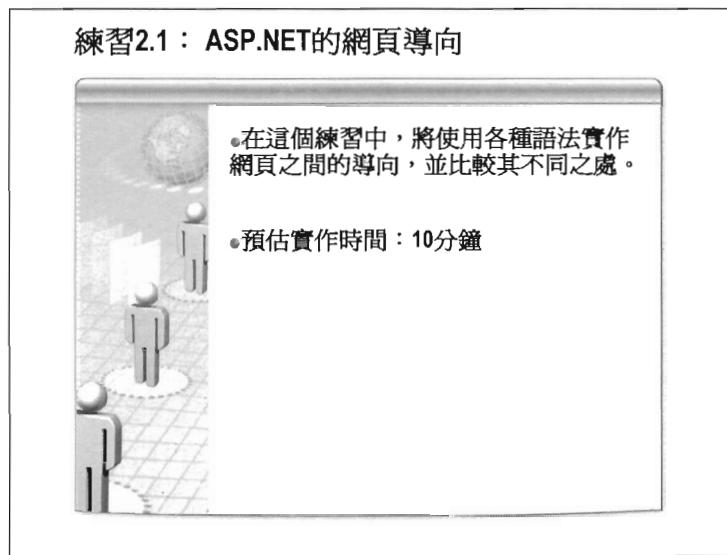
程式碼如下所示：

Visual Basic

```
Server.Transfer("~/Target.aspx")
```

C#

```
Server.Transfer("~/Target.aspx")
```



練習 2.1：ASP.NET 的網頁導向

目的：

這個練習主要是讓您熟悉 ASP.NET 的網頁導向，透過超連結、瀏覽器重新導向、跨網頁 POST 與伺服器重新導向等四種導向方式來練習網頁間互相導向的方式。

功能描述：

這個練習會在網頁中使用一個 HyperLink 與三個 Button 控制項，四個控制項分別對應到本練習的四種導向方式。

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」→「New Web Site」→選取「Empty Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選

取「\U9543\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」，檔名命名為「Source.aspx」，清除「Place code in separate file」核取方塊，建立來源網頁。
4. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，檔名命名為「Target.aspx」，清除「Place code in separate file」核取方塊，建立目的網頁。
5. 在「Source.aspx」網頁內，從「Toolbox」工具箱中拖拉一個 HyperLink 控制項，三個 Button 控制項。控制項的名稱屬性如下所示：

控制項	屬性	屬性值
HyperLink	ID	HyperLink1
	Text	測試超連結
	NavigateUrl	~/Target.aspx
Button	ID	Button1
	Text	瀏覽器重新導向
Button	ID	Button2
	Text	跨網頁 POST
	PostBackUrl	~/Target.aspx
Button	ID	Button3
	Text	伺服器重新導向

完成後樣式如下：



6. 用滑鼠點選 Button 控制項 Button1 兩次，Visual Studio 開發工具會建立 Button1 控制項的 Click 事件的處理常式，在程式碼區塊中加入下面的程式碼：

Visual Basic

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs)
    Response.Redirect("~/Target.aspx")
End Sub
```

C#

```
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Redirect("~/Target.aspx");
}
```

- 用滑鼠點選 Button 控制項 Button3 兩次，Visual Studio 開發工具會建立 Button3 控制項的 Click 事件的處理常式，在程式碼區塊中加入下面的程式碼：

Visual Basic

```
Protected Sub Button3_Click(ByVal sender As Object, ByVal e As
System.EventArgs)
    Server.Transfer("~/Target.aspx")
End Sub
```

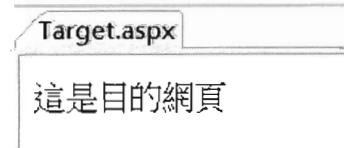
C#

```
protected void Button3_Click(object sender, EventArgs e)
{
    Server.Transfer("~/Target.aspx");
}
```

- 在「Target.aspx」網頁內，加入一個 Label 控制項，控制項的名稱屬性如下所示：

控制項	屬性	屬性值
Label	ID	Label1
	Text	這是目的網頁

完成後內容如下：



9. 執行網頁測試。在「Solution Explorer」→點選「Source.aspx」網頁→按滑鼠右鍵→選「View In Browser」。當網頁上的四個控制項被點選時，畫面將會被導向至「Target.aspx」網頁。

10. 注意 Button3 控制項的按鈕，雖然點選該按鈕之後，網頁內容會出現「Target.aspx」網頁結果，但是網址列卻還是顯示「Source.aspx」的 URL。



不同網站的網頁間傳遞值

前文中，介紹了如何使用 ASP.NET 之間的網頁導向方式。接著要更進一步，要介紹如何在網頁導向時傳遞資訊。首先介紹的兩種方式是可跨網站傳遞資料的方式：QueryString 與 Form：

QueryString

這種方式就是一般俗稱的查詢字串，設計方式分為兩部分，來源網頁與目的網頁

來源網頁的部分，可利用前文的任何一種網頁導向方式，不同的是在設定目的網頁 URL 的部分，更改成「URL?QueryString」而在QueryString 內依照「key=value」組成任何要傳遞的資訊；可設定多組，每組之間使用「&」符號連接。例如：

```
http://localhost/CS/QueryString_b.aspx?id=Tony&pw=1234
```

目的網頁的部分，則在程式內利用 Request.QueryString 的集合，依照QueryString 的 key 值來取得 value 值即可，如下所示：

```
Visual Basic
```

```
Response.Write(Request.QueryString("id"))
```

C#

```
Response.Write(Request.QueryString["id"]);
```

QueryString 的特性為

- 適用使用任何導向方式。
- 不同網站之間，也能使用該方法傳遞資訊。
- 由於傳遞的資訊是直接寫在網址上，所以此種方法的安全性是很低的，容易被竊取或是修改。

Form

這項方法是當來源網頁使用跨網頁 POST 或是伺服器重新導向的方式時，可利用該方式來取得來源網頁所 POST 的內容。

例如來源網頁上有按鈕(Button)與輸入欄位(TextBox)，當按下按鈕作跨網頁 POST 時，就可以在目的網頁的 Request.Form 集合內找到 key 值為輸入欄位(TextBox)id 的使用者輸入內容，如下所示：

Visual Basic

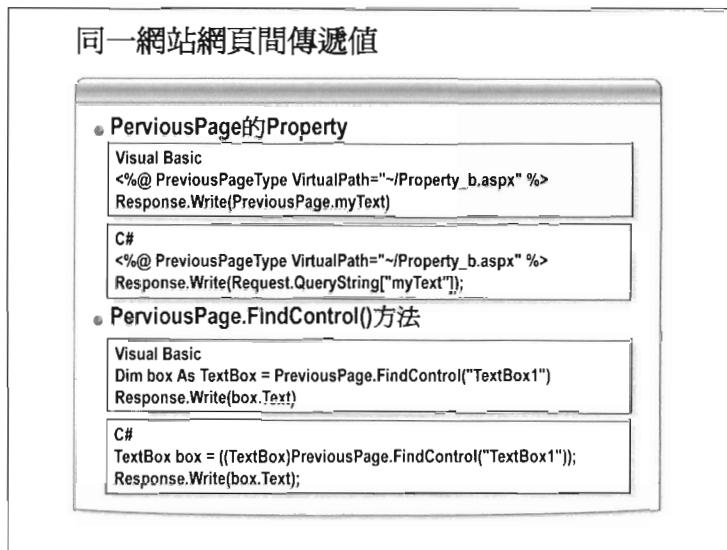
```
Response.Write(Request.Form("TextBox1").ToString())
```

C#

```
Response.Write(Request.Form["TextBox1"].ToString());
```

Form 的特性為

- 僅適用跨網頁 POST 或是伺服器重新導向的方式
- 取得來源網頁所 POST 的資料，而非所有控制項的值或屬性
- ASP.NET Web 網頁的 POST 的資料中會包含內部網頁處理所需要的資訊，例如 __VIEWSTATE、__EVENTTARGET 和 __EVENTARGUMENT。



同一網站網頁間傳遞值

前文中介紹兩種網頁間傳遞值的方式，是可以跨網站傳遞資料使用的。而本文中所介紹的方式，則只能在同一個 ASP.NET 的網站之前傳遞值時所能使用，雖然多了些限制，但是在設計上的便利性也提升不少。

PreviousPage 的 Property

ASP.NET 網頁中提供了一個屬性「`PreviousPage`」，可供目標網頁利用該屬性取得來源網頁的內容。而設定公用屬性(Property)的方式則是一種相當方便的資訊傳遞方法。設計方式分為兩部分，來源網頁與目的網頁。

來源網頁的部分，僅能使用跨網頁 POST 或是伺服器重新導向的方式來做網頁導向，另外在網頁內容的部分，還需要加入公用屬性(Property)宣告，公用屬性宣告內可設定要傳送的任何值，設定唯讀屬性即可，因為目的網頁也無法為來源網頁設定值～設定方式如下：

```
Visual Basic
Public ReadOnly Property myText() As String
Get
```

```
Return "Hello UCOM"
End Get
End Property
```

```
C#
public string myText
{
    get { return "Hello UCOM"; }
}
```

目的網頁的部分，其實使用跨網頁 POST 或是伺服器重新導向的方式來做網頁導向都可以取得 PreviousPage 物件，但是如果要能取得來源網頁的公用屬性(Property)，則需要使用 PreviousPageType 指示詞來指定來源網頁才可使用，指定方式如下：

```
<%@ PreviousPageType VirtualPath="~/Source.aspx" %>
```

指定完成之後，即可在程式碼內，使用強行別的方式來存取來源網頁內的公用屬性(Property)，如下所示：

```
Visual Basic
Response.Write(PreviousPage.myText)
```

```
C#
Response.Write(PreviousPage.myText);
```

PreviousPage 的 Property 的特性為：

- 僅適用跨網頁 POST 或是伺服器重新導向的方式
- 來源網頁與目的網頁必須要在同一個 ASP.NET 網站內

PreviousPage 的 FindControl 方法

該方法與 PreviousPage 的屬性類似，不同的是該方法只能取得來源網頁的控制項屬性，不過好處是不需要像上一方法要做許多前置設定。設計方式只需要在目的網頁中，利用 PreviousPage 的 FindControl 方法依來源網頁的控制項 id 取回控制項，再轉型回該控制項的型別即可，如下所示：

Visual Basic

```
Dim box As TextBox = PreviousPage.FindControl("TextBox1")
Response.Write(box.Text)
```

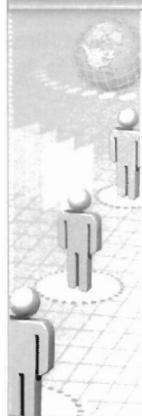
C#

```
TextBox box = ((TextBox)PreviousPage.FindControl("TextBox1"));
Response.Write(box.Text);
```

PreviousPage 的 FindControl 方法的特性為：

- 僅適用跨網頁 POST 或是伺服器重新導向的方式
- 來源網頁與目的網頁必須要在同一個 ASP.NET 網站內
- 僅能取得預設表單內容中的控制項，如欲取得的控制項在其它控制項容器內，則需要先找回該容器控制項。

練習2.2：在ASP.NET Web 網頁之間傳遞值



•在這個練習中，將使用各種語法來讓被導向網頁得到前一網頁的內容。

•預估實作時間：15分鐘

練習 2.2：在 ASP.NET Web 網頁之間傳遞值

目的：

這個練習主要是讓您熟悉 ASP.NET Web 網頁之間傳遞值的各種方法。

功能描述：

這個練習會在網頁中建立一張來源網頁，四張目的網頁，在來源網頁上放置四個 Button 與一個 TextBox 控制項，讓四個 Button 控制項在被點選之後，且在個別的來源網頁內使用各種方法來取得來源頁所傳遞過來的資訊。

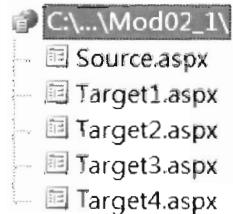
預估實作時間：15 分鐘

實作步驟：

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

取「\U9543\Practices\VB 或 CS\Mod02_2\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod02_2」。

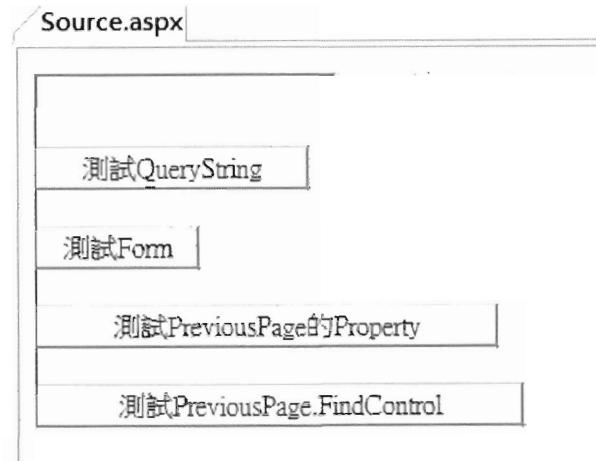
3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，檔名命名為「Source.aspx」，清除「Place code in separate file」核取方塊，建立來源網頁。
4. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，檔名命名為「Target1.aspx」，清除「Place code in separate file」核取方塊，建立目的網頁。並依照此方法依序建立「Target2.aspx」、「Target3.aspx」與「Target4.aspx」。完成後檔案結構應如下所示：



5. 在「Source.aspx」網頁內，從「Toolbox」工具箱中拖拉一個 TextBox 控制項，四個 Button 控制項。控制項的名稱屬性如下所示：

控制項	屬性	屬性值
TextBox	ID	TextBox1
Button	ID	Button1
	Text	測試 QueryString
Button	ID	Button2
	Text	測試 Form
	PostBackUrl	~/Target2.aspx
Button	ID	Button3
	Text	測試 PreviousPage 的 Property
	PostBackUrl	~/Target3.aspx
Button	ID	Button4
	Text	測試 PreviousPage. FindControl
	PostBackUrl	~/Target4.aspx

完成後應如下所示：



6. 滑鼠點選 Button 控制項 Button1 兩次，Visual Studio 開發工具會建立 Button1 控制項的 Click 事件的處理常式，在程式碼區塊中加入下面的程式碼：

```
Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Response.Redirect("~/Target1.aspx?myText=" & TextBox1.Text)
End Sub
```

```
C#
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Redirect("~/Target1.aspx?myText=" + TextBox1.Text);
}
```

7. 在程式碼區塊中加入以下公用屬性：

```
Visual Basic
Public ReadOnly Property myText() As String
    Get
        Return TextBox1.Text
    End Get
End Property
```

```
C#
public string myText
{
    get { return TextBox1.Text; }
}
```

8. 在 Target1.aspx 的 Page_Load 事件內加入以下程式碼：

```
Visual Basic
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Response.Write(Request.QueryString("myText"))
End Sub
```

```
C#
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write(Request.QueryString["myText"]);
}
```

9. 在 Target2.aspx 的 Page_Load 事件內加入以下程式碼：

```
Visual Basic
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Response.Write(Request.Form("TextBox1"))
End Sub
```

```
C#
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write(Request.Form["TextBox1"]);
}
```

10. 在 Target3.aspx 的 Page 指示詞下加入 PreviousPageType 指示詞：

```
<%@ PreviousPageType VirtualPath="~/Source.aspx" %>
```

11. 在 Target3.aspx 的 Page_Load 事件內加入以下程式碼：

```
Visual Basic
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Response.Write(PreviousPage.myText)
End Sub
```

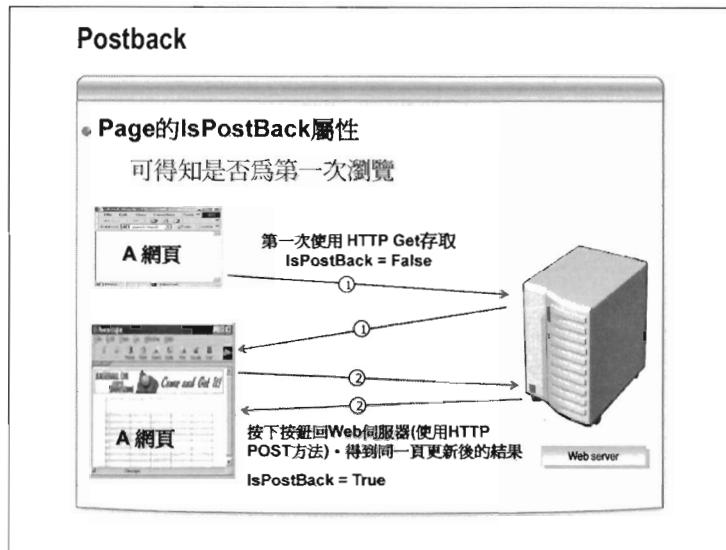
```
C#
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write(PreviousPage.myText);
}
```

12. 在 Target4.aspx 的 Page_Load 事件內加入以下程式碼：

```
Visual Basic  
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)  
    Dim tb As TextBox = PreviousPage.FindControl("TextBox1")  
    Response.Write(tb.Text)  
End Sub
```

```
C#  
protected void Page_Load(object sender, EventArgs e)  
{  
    TextBox tb = ((TextBox)PreviousPage.FindControl("TextBox1"));  
    Response.Write(tb.Text);  
}
```

13. 執行網頁測試。在「Solution Explorer」→點選「Source.aspx」網頁→按滑鼠右鍵→選「View In Browser」。在 TextBox 輸入任意值，接著點選網頁上的四個 Button 控制項，畫面將會被導向至個別網頁，並且顯示「Source.aspx」的 TextBox 內的文字。



Postback

使用者在瀏覽網頁時，如果點選了瀏覽器上的控制項，如按鈕，預設會將利用 POST 的方式導向至自己本身的網頁，以便再次執行其伺服端程式碼，然後將新的頁面結果呈現給使用者。這種行為我們就稱之為 Postback。

以下為瀏覽器第一次瀏覽網頁的流程：

1. 使用者提出網頁要求。(利用 HTTP GET)
2. 伺服器執行程式(Page_Load)，動態產生出標籤，並將標籤傳送到瀏覽器。

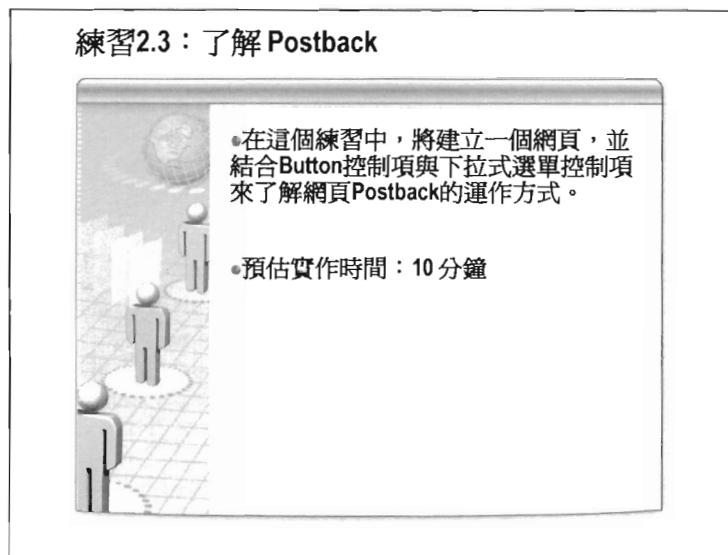
以下為使用者取得網頁後，按下按鈕的流程：

1. 使用者按下按鈕。
2. 瀏覽器使用 POST 的方式再度導向相同頁面 (Postback)。
3. 在伺服器上，網頁再次執行程式(Page_Load)。
4. 執行按鈕的 Click 事件內的程式碼。
5. 網頁產生出標籤，將標籤送回瀏覽器。

由上述流程可得推論得知，每一次使用者點選按鈕，就會再重覆上述 1 到 5 步驟，也就是每次都會進行一次與伺服器間的往返動作。

在這些流程中需特別注意的是，不管是網頁因為瀏覽器第一次瀏覽，或是因為利用網頁上控制項作 PostBack，Page_Load 事件內的程式碼都會主動執行，如果 Page_Load 內的程式碼只需要在瀏覽器第一次瀏覽時才執行(如控制項初始化)，那麼放在 Page_Load 內的程式碼就需要做判斷才行。

網頁的執行個體(page)中，提供了一個名為 IsPostBack 的布林值屬性，如果程式是因為 Postback 而再度執行，那麼該屬性就會顯示 true。換而言之，如果是第一次執行該網頁，則是 false。利用該屬性的判斷，就可以簡單的寫出只有第一次才能執行的程式碼區段了。



練習 2.3：了解 Postback

目的：

在這個練習中，將建立一個網頁，並結合 Button 控制項來了解網頁 Postback 的運作方式。

功能描述：

在練習中，你將建立一個網頁，加入一個 Label 與 Button，並在 Page_Load 事件撰寫伺服端的程式碼，以測試 Postback 的運作方式。

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「Empty Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9543\Practices\VB 或 CS\Mod02_3\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod02_3」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，檔名命名為「UsePostBack.aspx」，清除「Place code in separate file」核取方塊，建立來源網頁。
4. 從「Toolbox」→「Standard」頁籤拖曳一個 Label 控制項與一個 Button 控制項到網頁，如下所示：



5. 雙擊網頁設計畫面任何空白處產生 Page_Load 事件。在 Page_Load 事件中撰寫以下程式碼：

Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Label1.Text = Now.ToString()
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = System.DateTime.Now.ToString();
}
```

6. 在「Solution Explorer」視窗→點選 UsePostBack.aspx 網頁→按滑鼠右鍵→選「View In Browser」。

這個網頁一執行，就會在 Label 上顯示系統時間，每當你點選按鈕一下，又會重新讀取系統的最新時間並顯示。

7. 結束程式的執行。回到程式撰寫視窗。

8. 修改 Page_Load 事件程式碼：

Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
```

```
stem.EventArgs)
If Page.IsPostBack Then
    Label1.Text = Now.ToString()
Else
    Label1.Text = ""
End If
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Page.IsPostBack)
    {
        Label1.Text = System.DateTime.Now.ToString();
    }
    else
    {
        Label1.Text = "";
    }
}
```

在「Solution Explorer」視窗→點選 UsePostBack.aspx 網頁→按滑鼠右鍵→選「View In Browser」。你將發現第一次執行這個網頁時，Label 上的文字是空白的，待按下 Button 之後，才會顯示系統的時間。

9. 結束程式的執行。

總結

- Web Form內容
- 預設網頁模型 (Page Model)
- 網頁間的互相導向
- Postback
- 總結

第三章: 使用網頁控制項

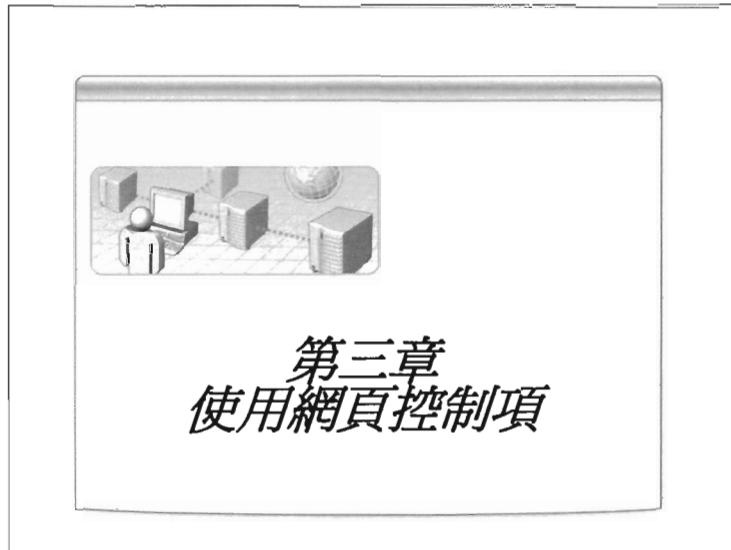
本章大綱

課程大綱	3
ASP.NET 的基本控制項類型	4
伺服器控制項與 HTML 控制項	5
控制項的事件	7
各種常見的控制項	9
TextBox	10
Image	12
HiddenField	13
Button、ImageButton 與 LinkButton	14
練習 3.1: 使用 Button、HiddenField 與 TextBox 控制項 ..	16
CheckBox 與 RadioButton	19
清單系列控制項	21
練習 3.2: 使用清單系列控制項	24
Panel	28
Calendar	29
練習 3.3: 使用 Calendar 控制項	30
呼叫用戶端指令碼	33
動態註冊用戶端指令碼 - 1	34
動態註冊用戶端指令碼 - 2	36
練習 3.4: 動態註冊用戶端指令碼	37

作者：

周季賢





大綱

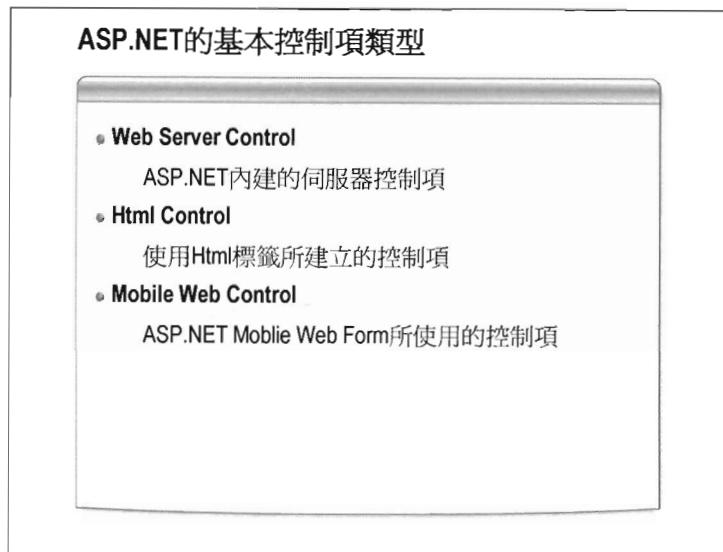
- ASP.NET的基本控制項類型
- 關於ASP.NET Web 伺服器控制項
- 控制項的事件
- 各種常見的控制項
- 用戶端指令碼

課程大綱

網頁表單控制項是建立網頁使用者介面的基本要件，熟悉各種控制項的常用屬性、事件及方法是開發網頁程式的第一步。

本章介紹以下主題：

- ASP.NET 的基本控制項類型
- 關於 ASP.NET Web 伺服器控制項
- 控制項的事件
- 各種常見的控制項
- 用戶端指令碼



ASP.NET 的基本控制項類型

當我們在開發 ASP.NET Web 應用程式時，可以使用下列控制項來做網頁介面的內容：

Web Server Control

ASP.NET 開發時的主力控制項，該種控制項是具有許多內建功能的控制項。像按鈕、文字方塊、月曆、功能表和樹狀檢視控制項等特殊用途的控制項。

Html Control

使用 Html 標籤(或是其他支援之標記中的項目，例如 XHTML)所建立的控制項，根據預設，伺服器無法使用 ASP.NET Web 網頁上的 HTML 控制項。

Mobile Web Control

ASP.NET 針對行動式裝置網頁開發的特殊控制項，也是一種伺服器控制項。標籤與一般的伺服器控制項不同，而且也僅能在 Mobile Web Form 內使用。



伺服器控制項與 Html 控制項

伺服器控制項與 Html 控制項由於在網頁上所見到的外觀相同，所以很容易因為不熟悉之間的差別，而造成選擇控制項時不知道該如何下手，以下說明這兩種控制項的差異：

伺服器控制項

伺服器控制項是 ASP.NET 所提供的內建控制項，該種控制項非常多樣化，而且也只限 ASP.NET 開發時所使用。以下是伺服器控制項的特性：

- 具備伺服器端的事件：

伺服器控制項都具有伺服器端的事件；這些事件通常會因使用者對控制項的各種行為而觸發，使用者可以在針對這些事件所產生的事件處理常式中加入程式碼，如此一來即可在事件觸發之時，確保程式正確執行。

- 自動保存狀態：

伺服器端控制項的另一項特色，就是能夠自動保存控制項的狀態。例如當下拉式選單選擇項目後，網頁重新載入時希望下拉式選單繼續停留在使用者所選擇的項目，以往開發者必

須撰寫程式才能做到，而使用伺服器端控制項便不需撰寫任何程式就能自動維護控制項狀態。

- 根據前端瀏覽器能力產生不同的 Html：

伺服器控制項的另一特點，就是開發時的標籤與傳送到使用者端的標籤並不相同，例如 TextBox 的標籤在開發時如下：

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

但是，最後使用者端所看到的標籤是如下所示：

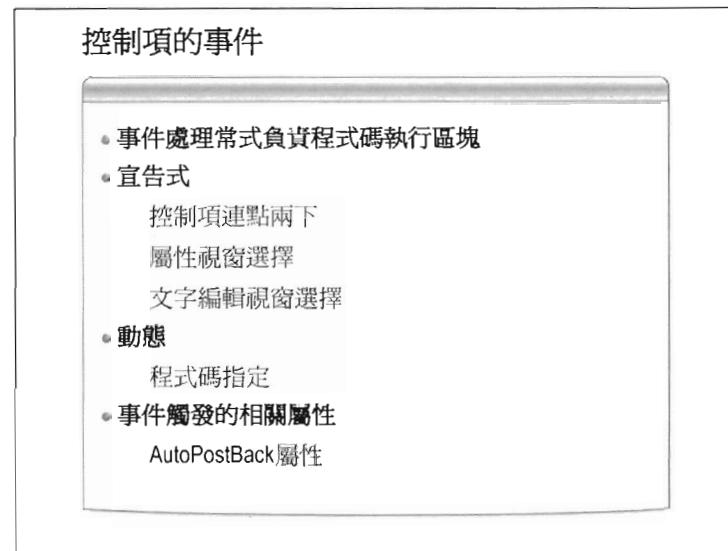
```
<input name="TextBox1" type="text" id="TextBox1" />
```

這是因為 ASP.NET 執行環境會自動抓取使用者瀏覽器的種類及版本，以決定伺服器控制項要產生哪種適合的 HTML 標籤。

Html 控制項

Html 控制項是使用一般的 Html 標籤來開發，該種控制項是標準的，所有的網頁開發程式與瀏覽器都能支援。而這種控制項在開時期與使用者最後看到的結果是相同的。

而預設該種控制項是無法與 ASP.NET 的伺服器溝通的，必須透過特殊的處理方式才能具有該功能。



控制項的事件

控制項透過事件處理常式，來提供使用者加入程式碼內容，並且在控制項發生該事件時，執行其內的程式碼。而預設，所有控制項的事件處理常式都不會出現，必須透過建立的方式才使用，以下介紹幾宣告與動態的建立方式：

宣告式

宣告式是在設計時期就需確定，明確的建立各控制項的事件處理常式。開發工具支援宣告式的事件處理常式建立，建立方式如有以下三種：

- **控制項連點兩下：**
每一個控制項都有一個預設的事件，該事件通常也是最常用的。在設計頁面直接對控制項連點兩下滑鼠右鍵，即可建立事件處理常式。
- **屬性視窗選擇：**
這種設計方式比較通用，可針對每一個控制項的所有事件來建立事件處理常式。在設計頁面挑選控制項，在對屬性視窗內的事件連點兩下滑鼠，即可建立。
- **文字編輯視窗選擇：**

該種設計方式必須要將編輯畫面切換到原始碼頁面，在原始碼編輯頁面上方的兩個下拉式選單選擇控制項與事件即可建立。

動態

動態建立事件處理常式的方式是在執行時期才決定，該種方式也必須要先將事件處理常式準備好，另外利用程式來指定控制項的事件與事件處理常式的細節關係。程式碼與控制項的關係如下所示：

```
protected void Page_Load(object sender, EventArgs e)
{
    Button4.Click += new EventHandler(Button4_Click);
}
protected void Button4_Click(object sender, EventArgs e)
{
    Response.Write("AAA");
}
```

事件觸發的相關屬性

另外需注意的是，並非所有伺服器控制項在事件發生的當下都會主動回伺服器處理事件處理常式內的程式碼。部分控制項是屬於被動式的，如果要讓這類的控制項主動回伺服器執行，可將這類控制項的 AutoPostBack 屬性值設定為 true 即可。

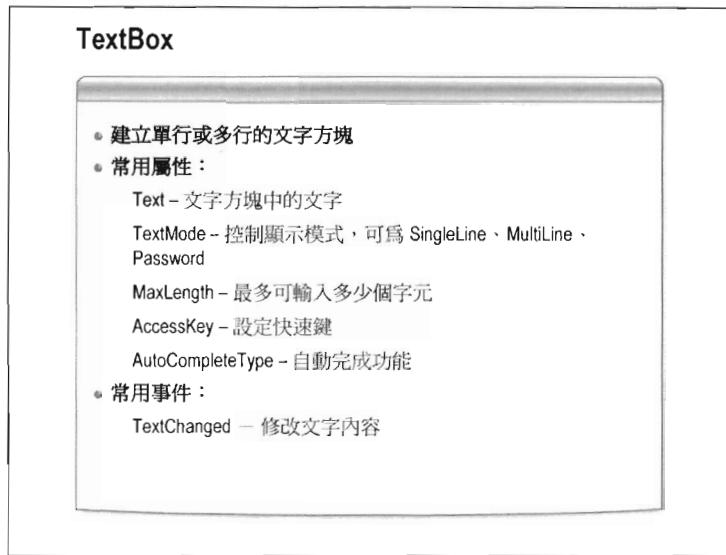
各種常見的控制項

- TextBox
- Image
- HiddenField
- Button、ImageButton與LinkButton
- CheckBox與RatioButton
- 清單系列控制項
- Panel
- Calendar

各種常見的控制項

本小節要介紹各種網頁開發時常見的控制項，這些控制項都位於 Toolbox 內的 Standard 頁籤內。控制項如下：

- TextBox
- Image
- HiddenField
- Button、ImageButton 與 LinkButton
- CheckBox 與 RatioButton
- 清單系列控制項
- Panel
- Calendar



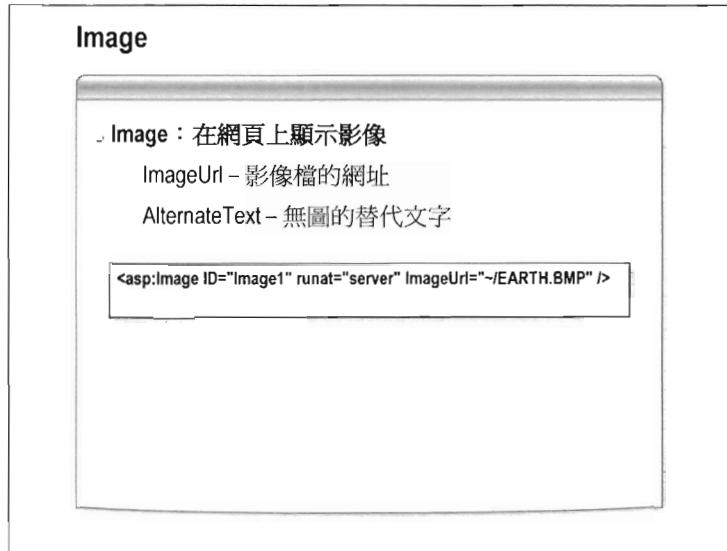
TextBox

TextBox 可以在執行時期建立各式各樣的文字方塊，是一個非常實用的控制項，它可以輕易建立單行、多行與密碼用的文字輸入方塊。它的常用屬性如下：

- Text：
文字方塊中的文字。
- TextMode：
控制顯示模式，可為 SingleLine、MultiLine、Password。預設為 SingleLine，也就是單行，如果要顯示為多行，則必須改為 MultiLine。如果要用來輸入密碼，則可以使用 Password，這樣的話，使用者輸入到文字方塊中的字元都會以「*」顯示。
- MaxLength：
最多可輸入多少個字元。預設為 0，也就是不限制長度。
- AccessKey：
設定快速鍵，可讓使用者在網頁利用鍵盤快速找到對應的 TextBox。
- AutoCompleteType：
選擇自動完成的類型，讓使用者選擇之前輸入過的值。

另外常用事件如下：

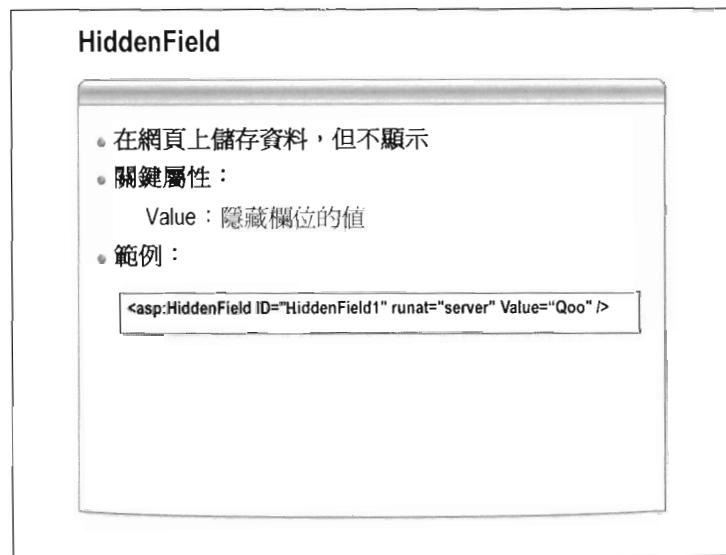
- **TextChanged**：
當使用者修改文字內容之後觸發。



Image

Image 控制項可以在網頁上顯示影像，可用程式控制或是設計時期指定影像的來源。常用屬性如下：

- **ImageUrl**
設定影像的來源網址。
- **AlternateText**
找不到影像時，該文字就會出現。



HiddenField

在開發網頁應用程式時，開發人員經常需要在網頁上使用`<input type="hidden">` 標籤來儲存一些隱藏的資料。HiddenField 控制項就是對應到 HTML 的隱藏欄位標籤，儲存在隱藏欄位的資料都不會顯示在瀏覽器上。

HiddenField 控制項的關鍵屬性只有一個：Value，也就是隱藏欄位的值。

範例：

```
<asp:HiddenField ID="HiddenFld" runat="server" Value="Qoo" />
```

此範例所產生的 HTML 標籤為：

```
<input type="hidden" name="HiddenFld" id="HiddenField1" value="Qoo" />
```



Button 、ImageButton 與 LinkButton

Button 、ImageButton 與 LinkButton 這三個控制項都是預設所提供的按鈕控制項；主要功能相同，僅外觀的部分不同而已。

- **Button**

即為最常見的控制項類型，會在網頁上出現按鈕圖示供使用者點選。

- **ImageButton**

提供影像按鈕，可供使用者自行提供自訂的按鈕圖片。與 Image 控制項類似，使用 ImageUrl 即可設定影像位置。

- **LinkButton**

外觀類似超連結，不過卻是按鈕，能夠讓使用者點選之後做按鈕該做的事情。

以下是 Button 、ImageButton 與 LinkButton 控制項共有的屬性與事件：

- **CommandName** :

與按鈕關聯的命令名稱，可供多個 Button 控制項共用事件時提供辨識的命令名稱。

- CommandArgument :

與按鈕關聯的命令參數，當辨識出命令名稱之後，可進一步依照 CommandArgument 來辨識命令所需的參數。

- Click :

按鈕按下的事件

- OnClientClick :

按鈕按下的前端事件，供呼叫用戶端指令碼時使用。

- Command :

按鈕按下的事件，可在該事件內取得該按鈕的 CommandName 與 CommandArgument 屬性，來執行適當的動作



練習 3.1 : 使用 Button、HiddenField 與 TextBox 控制項

目的：

這個練習主要是讓您熟悉使用 Button 控制項的 Click 事件，HiddenField 的 Value 屬性與 TextBox 控制項的 Maxlength 屬性，利用這些屬性來建立一個簡單的會員登入網頁所需的功能。

功能描述：

這個練習會在網頁中使用一個 Button、HiddenField 與兩個 TextBox 控制項，利用 HiddenField 讓私下記錄使用者登入網頁的次數，並在失敗三次之後阻止使用者再次使用登入功能。

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「Empty Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9543\Practices\VB 或 CS\Mod03_1\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod03_1」。
3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，檔名命名為「testLogin.aspx」，清除「Place code in separate file」核取方塊，建立來源網頁。
4. 在「testLogin.aspx」網頁內，從「Toolbox」工具箱中拖拉兩個 TextBox 控制項，一個 Button 控制項與一個 HiddenField 控制項。控制項的名稱屬性如下所示：

控制項	屬性	屬性值
TextBox	ID	txtUserName
TextBox	ID	txtPassword
Button	ID	Button1
	Text	登入
HiddenField	ID	HiddenField1
	Value	0

另外在適當位置加入純文字「帳號：」與「密碼：」，完成後設計頁面應如下所示：

The design view shows the following elements:

- A text input field labeled "帳號：" followed by an empty input field.
- A text input field labeled "密碼：" followed by an empty input field.
- A button labeled "登入" (Login).
- A hidden field labeled "HiddenField - HiddenField1".

5. 滑鼠點選 Button 控制項 Button1 兩次，Visual Studio 開發工具會建立 Button1 控制項的 Click 事件的處理常式，在程式碼區塊中加入下面的程式碼：

```
Visual Basic
If txtUserName.Text = "UCOM" AndAlso txtPassword.Text = "111"
Then
    Response.Write("帳號密碼正確")
Else
    Dim count As Integer = Convert.ToInt32(HiddenField1.Value)
```

```

count = count + 1
If count > 3 Then
    Response.Write("錯誤超過三次,拒絕登入")
    Button1.Enabled = False
Else
    Response.Write("帳號密碼不正確 , 錯誤次數:" & count.ToString())
End If
HiddenField1.Value = count.ToString()
End If

```

```

C#
if (txtUserName.Text == "UCOM" && txtPassword.Text == "111")
{
    Response.Write("帳號密碼正確");
}
else
{
    int count = Convert.ToInt32(HiddenField1.Value);
    count++;
    if (count > 3)
    {
        Response.Write("錯誤超過三次,拒絕登入");
        Button1.Enabled = false;
    }
    else
    {
        Response.Write("帳號密碼不正確 , 錯誤次數:" + count.ToString());
    }
    HiddenField1.Value = count.ToString();
}

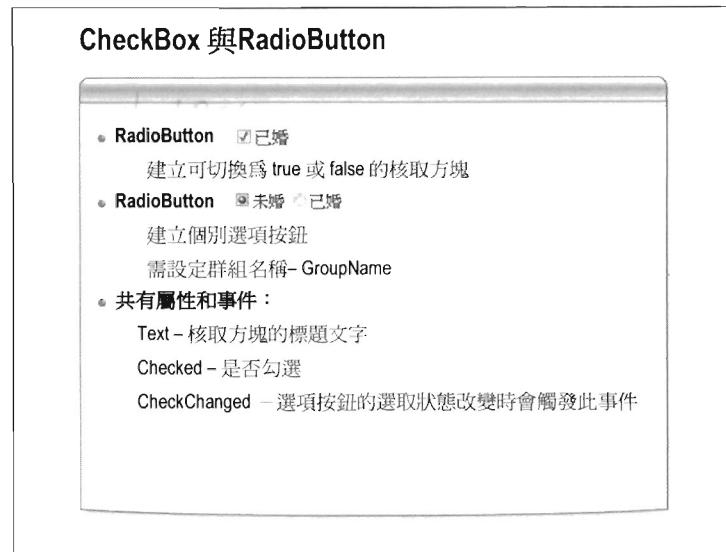
```

6. 執行網頁測試。在「Solution Explorer」→點選「testLogin.aspx」網頁→按滑鼠右鍵→選「View In Browser」。在帳號與密碼欄位輸入不正確的數值，接著點選網頁的 Button1 控制項，畫面最上方會顯示錯誤與不正確的次數；次數超過三次，會出現拒絕登入的訊息，並且 Button1 控制項也會被停止使用。如下所示：

錯誤超過三次,拒絕登入

帳號：

密碼：



CheckBox 與 RadioButton

CheckBox 與 RadioButton 控制項可在網頁上顯示核取方塊，讓使用者選擇。使用方式大致相同，不過這兩種控制項對於使用者選擇的方式是不同的。

- **CheckBox**

顯示方形的核取方塊，可讓使用者在 true 和 false 兩種情況中二選一，當被選取後，該方塊會以打勾表示。一群 CheckBox 控制項也可用來當作可多選的問題選項，如興趣選項等。

- **RadioButton**

顯示圓形的核取方塊，通常使用一群 RadioButton 控制項，不同的是同一群組 RadioButton 控制項內，僅能選擇一個，通常用來當作單選的問題選項，如婚姻狀況選項等。

RadioButton 有一個名為 GroupName 屬性，該屬性內的值相同，就會視為同一群組。

CheckBox 與 RadioButton 相同的屬性與事件如下：

- **Text**

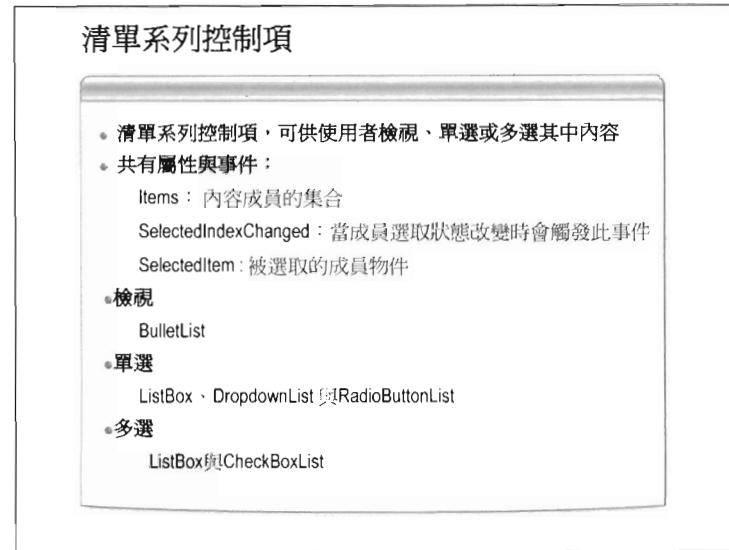
核取方塊的標題文字

- **Checked**

是否被勾選。如果被勾選，值為 true，反之為 false。

- **CheckChanged**

當選取狀態改變時會觸發此事件，可在該事件當下針對狀態改變的行為做適當的處理。



清單系列控制項

ASP.NET 提供為數眾多的清單系列控制項，可建立供使用者檢視、單選或多選內容的各式視覺化畫面。在此介紹五種清單系列控制項。分別如下：

- BulletList :

BulletList 控制項可用來建立一個帶有項目符號或項目編號的清單。該控制項提供名為 BulletStyle 的屬性可供挑選每個項目前面要顯示什麼符號或者數字編號。另外針對數字編號部分，還可以使用 FirstBulletNumber 屬性來決定第一個項目的編號。

- ListBox

ListBox 控制項可以包含多個項目，並且在網頁上顯示一個清單方塊。該控制項提供名為 SelectionMode 的屬性，能夠決定讓使用者選擇其中的一個或多個項目。

- DropDownList

DropDownList 控制項可以包含多個項目，並且在網頁上顯示一個下拉清單，讓使用者能夠從下拉式清單中選取某個項目。

- RadioButtonList

RadioButtonList 控制項可在網頁上一次建立多個 RadioButton，並且建立出來的 RadioButton 均屬於同一個群組。

- CheckBoxList

CheckBoxList 控制項可在網頁上顯示多個 CheckBox，當你需要在網頁上建立多個核取方塊時，使用 CheckBoxList 會比 CheckBox 更加方便。

清單系列共用的屬性與事件如下：

- Items

所有內容成員的集合，每一個成員都具有 Text 與 Value 屬性，Text 屬性的內容可供使用者檢視，而 Value 屬性是隱藏的，可用來儲存一些程式運作所需的關鍵字。

- SelectedIndexChanged :

當 Items 中的任何一個成員被選擇的狀態改變時會觸發此事件。需注意的是由於 BulletedList 僅能用於檢視，所以並無此事件。

- SelectedItem :

當成員被選取之後，可由該屬性取得成員物件。

在清單系列控制項設定成員的部分，有下列四種方式：

- 可從各控制項的屬性視窗直接開啓設定畫面來設定。
- 設定資料來源控制項當作資料來源。
- 撰寫程式，利用 Items 屬性的 Add 方法來動態載入成員。
- 撰寫程式，利用控制項的 DataSource 屬性來指定集合或陣列等當作資料來源

在清單系列控制項取得選取成員的部分，有下列兩種方式：

- 單選系列控制項，如 ListBox、DropdownList 與 RadioButtonList；可直接使用 SelectedItem 屬性取得所選成員內容。
- 多選系列控制項，如 ListBox 與 CheckBoxList；需要利用迴圈一一檢視成員內容，如下所示：

```
Protected Sub Button1_Click(ByVal sender As Object,
    ByVal e As System.EventArgs)

    Dim i As Integer
    For i = 0 To CheckBoxList1.Items.Count - 1
        If CheckBoxList1.Items(i).Selected Then
            Response.Write(CheckBoxList1.Items(i).Text & "<BR>")
        End If
    Next
End Sub
```

C#

```
protected void Button1_Click(object sender, EventArgs e)
{
    for (int i = 0; i < CheckBoxList1.Items.Count; i++) {
        if (CheckBoxList1.Items[i].Selected) {
            Response.Write(CheckBoxList1.Items(i).Text + "<BR>");
        }
    }
}
```



練習 3.2 : 使用清單系列控制項

目的：

這個練習主要是讓您熟悉使用清單系列控制項中的 ListBox 的使用方式，包含了初始化 ListBox 的內容成員，以及取得控制項內被選取的成員。

功能描述：

這個練習會在網頁中使用兩個 ListBox、一個 Button 控制項，利用 ListBox 的多選特性，讓使用者在其中一個 ListBox 挑選內容，並起在挑選完之後利用 Button 的 Click 事件的程式移動 ListBox 的成員至另一 ListBox 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「Empty Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9543\Practices\VB 或 CSM\Mod03_2\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod03_2」。
3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，檔名命名為「testListBox.aspx」，清除「Place code in separate file」核取方塊，建立來源網頁。
4. 在「testListBox.aspx」網頁內，從「Toolbox」工具箱中拖拉兩個 ListBox 控制項，一個 Button 控制項。控制項的名稱屬性如下所示：

控制項	屬性	屬性值
ListBox	ID	lbSource
	SelectionMode	Multiple
ListBox	ID	lbTarget
Button	ID	Button1
	Text	>>

另外適當設定 ListBox 的大小與控制項之間的位置，完成後設計頁面應如下所示：



5. 在設計頁面對空白處連點滑鼠右鍵兩下，進入 Page_Load 事件處理常式，在程式碼區塊中加入下面的程式碼：

```

Visual Basic
If Page.IsPostBack = False Then
    Dim ary() As String = {"成員一", "成員二", "成員三", "成員四", "成員五", "成員六"}
    lbSource.DataSource = ary
    lbSource.DataBind()
End If
End If
HiddenField1.Value = count.ToString()
End If

```

```
C#
if (Page.IsPostBack == false)
{
    string[] ary = new string[] { "成員一", "成員二", "成員三", "成員四",
        "成員五", "成員六" };

    lbSource.DataSource = ary;
    lbSource.DataBind();
}
```

6. 滑鼠點選 Button 控制項 Button1 兩次，Visual Studio 開發工具會建立 Button1 控制項的 Click 事件的處理常式，在程式碼區塊中加入下面的程式碼：

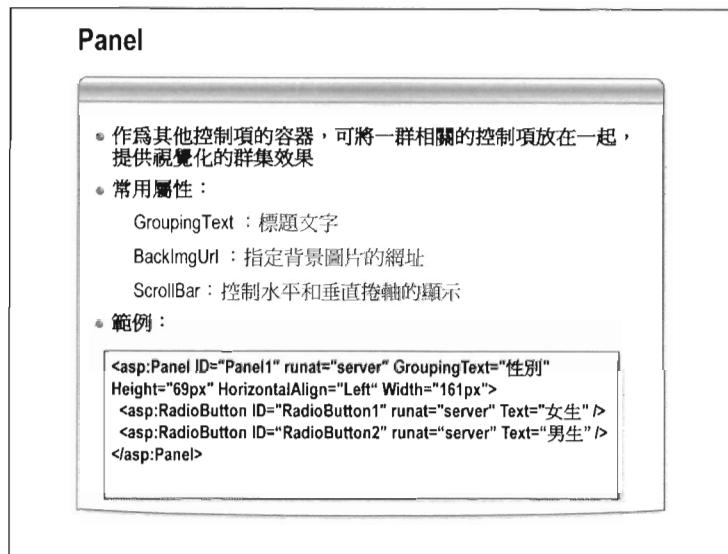
```
Visual Basic
For Each item As ListItem In lbSource.Items
    If item.Selected Then
        item.Selected = False
        lbTarget.Items.Add(item)
    End If
Next
For Each item As ListItem In lbTarget.Items
    If lbSource.Items.Contains(item) Then
        lbSource.Items.Remove(item)
    End If
Next
```

```
C#
foreach (ListItem item in lbSource.Items)
{
    if (item.Selected)
    {
        item.Selected = false;
        lbTarget.Items.Add(item);
    }
}
foreach (ListItem item in lbTarget.Items)
{
    if (lbSource.Items.Contains(item))
    {
        lbSource.Items.Remove(item);
    }
}
```

7. 執行網頁測試。在「Solution Explorer」→點選「testListBox.aspx」網頁→按滑鼠右鍵→選「View In Browser」。挑選左邊的 ListBox 的內容並按下 Button1 控制項

之後，被選取的內容應該會被搬移至右邊的 ListBox 控制項內。如下所示：



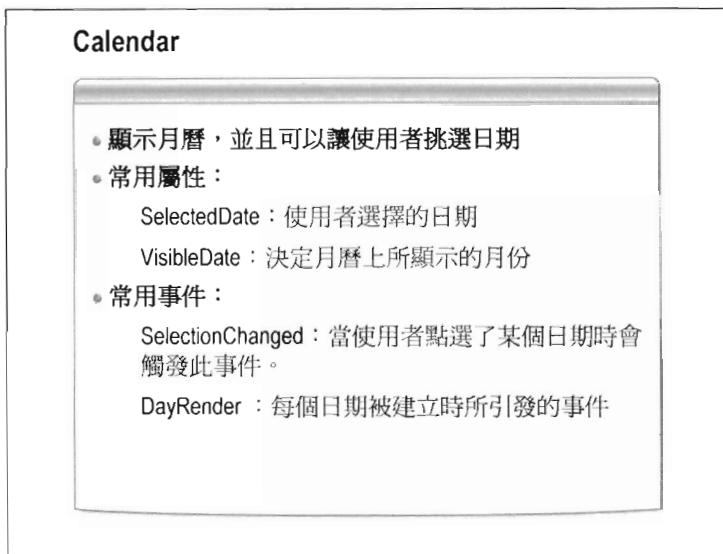


Panel

Panel 指控項可作為其他指控項的容器，將一群相關的指控項放在一個 Panel 裡面，可以達到視覺化的群組效果，或者方便利用程式碼控制一群指控項是否要顯示或者隱藏。

以下是 Panel 指控項的常用屬性：

- GroupingText：標題文字。
- BackImageUrl：指定背景圖片的網址。
- ScrollBar：控制水平和垂直捲軸的顯示。預設為 None，可指定 Horizontal、Vertical、或者 Both，以控制只要顯示水平捲軸、垂直捲軸、或者兩者都要顯示。若設定成 Auto 則指控項會自動判斷包含的指控項是否超出 Panel 的顯示區域，以決定要不要顯示捲軸。



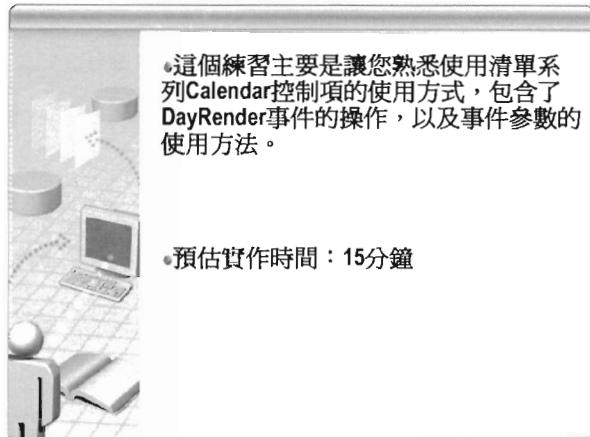
Calendar

Calendar 控制項可以在網頁上顯示月曆，並且讓使用者挑選日期。常用的屬性與事件如下：

- SelectedDate：
使用者選擇的日期。
- VisibleDate：
決定月曆上所顯示的月份，如僅設定了 SelectedDate 而沒有設定 VisibleDate，控制項並不會顯示 SelectedDate 所在的月份。
- SelectionChanged：
當使用者點選了某個日期時會觸發此事件。
- DayRender：
所顯示月份的每個日期被建立出來時，都會觸發該事件，可針對該事件來對欲客製化的日期著手。在該事件發生時，事件處理常式會提供對應的參數來提供使用者判斷或設定。

你可以用 Visual Studio 2008 提供的 Auto Format 功能，將月曆元件設定成預先定義的樣式。做法為：在 Calendar 控制項上點右鍵，再點選「Auto Format」，接著挑選一個樣式即可。

練習 3.3 :使用Calendar控制項



練習 3.3 :使用 Calendar 控制項

目的：

這個練習主要是讓您熟悉使用清單系列 Calendar 控制項的使用方式，包含了 DayRender 事件的操作，以及事件參數的使用方法。

功能描述：

這個練習會在網頁中使用一個 Calendar 控制項，利用其 DayRender 事件來替特定日期加入文字以及關閉選取功能。

預估實作時間：15 分鐘

實作步驟：

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

9. 從「File」→「New Web Site」→選取「Empty Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9543\Practices\VB 或 CS\Mod03_3\Starter」目錄，與使

用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod03_3」。

10. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，檔名命名為「testCalendar.aspx」，清除「Place code in separate file」核取方塊，建立來源網頁。
11. 在「testCalendar.aspx」網頁內，從「Toolbox」工具箱中拖拉一個 Calendar 控制項。自行調整其大小與 Auto Format 設定，再將 VisibleDate 屬性設定為 2008/11/13。
12. 建立 Calendar 的 DayRender 事件處理常式，在程式碼區塊中加入下面的程式碼：

```
Visual Basic
e.Day.IsSelectable = Not e.Day.IsWeekend

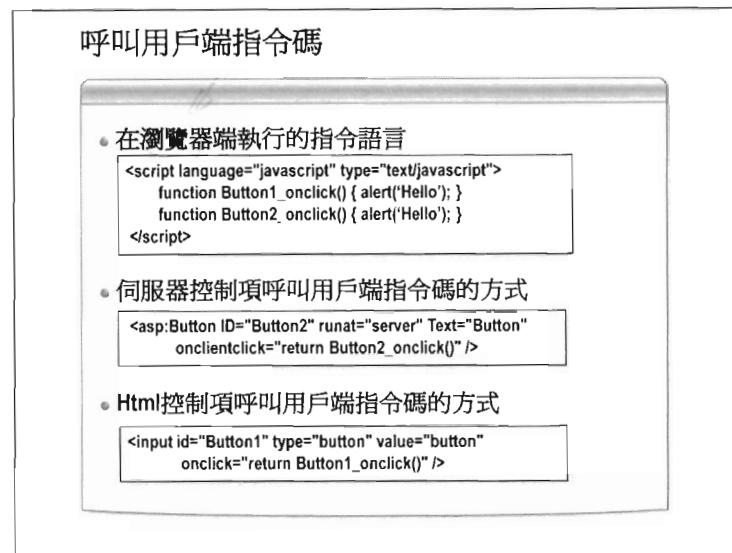
If e.Day.Date = New DateTime(2008, 11, 15) Then
    Dim lb As New Label()
    lb.Text = "<br/> 生日"
    lb.ForeColor = System.Drawing.Color.Red
    e.Cell.Controls.Add(lb)
End If
```

```
C#
e.Day.IsSelectable = !e.Day.IsWeekend;

if (e.Day.Date == new DateTime(2008, 11, 15))
{
    Label lb = new Label();
    lb.Text = "<br/> 生日";
    lb.ForeColor = System.Drawing.Color.Red;
    e.Cell.Controls.Add(lb);
}
```

13. 執行網頁測試。在「Solution Explorer」→點選「testCalendar.aspx」網頁→按滑鼠右鍵→選「View In Browser」。應該會發現日期是顯示於 2008 年 11 月份，並且在 2008 年 11 月 5 日的格子內有一紅字顯示「生日」，並且周末的部分均無法選取。如下所示：

Oct	November 2008						Dec
Sun	Mon	Tue	Wed	Thu	Fri	Sat	
	<u>27</u>	<u>28</u>	<u>29</u>	<u>30</u>	<u>31</u>	1	
2	<u>3</u>	<u>4</u>	<u>5</u>	<u>6</u>	<u>7</u>	8	
9	<u>10</u>	<u>11</u>	<u>12</u>	<u>13</u>	<u>14</u>	15 生日	
16	<u>17</u>	<u>18</u>	<u>19</u>	<u>20</u>	<u>21</u>	22	
23	<u>24</u>	<u>25</u>	<u>26</u>	<u>27</u>	<u>28</u>	29	
30	<u>1</u>	<u>2</u>	<u>3</u>	<u>4</u>	<u>5</u>	6	



呼叫用戶端指令碼

在開發 ASP.NET 網頁時，也可以呼叫用戶端的指令碼，用戶端指令碼與伺服器端指令碼（C#或 Visual Basic）不相同，是一種在用戶端（瀏覽器）執行的程式碼，常見的如 javascript 等。

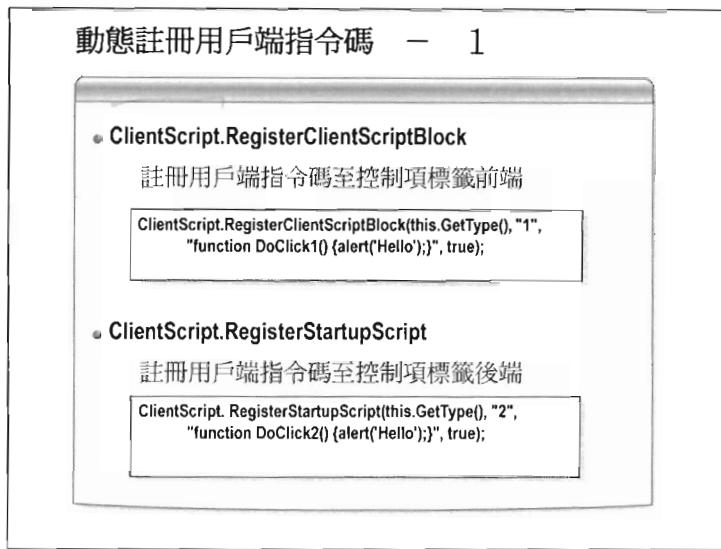
用戶端指令通常撰寫在程式碼頁面，利用特定的標籤宣告，如下所示：

```
<script language="javascript" type="text/javascript">
  function Button1_onclick() { alert('Hello'); }
  function Button2_onclick() { alert('Hello'); }
</script>
```

另外也可以撰寫在檔案中，使用特定標籤來繫結，如下所示：

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

而控制項呼叫方式也不相同，例如同樣是 Button 控制項，伺服器控制項就必須要使用名為 OnClientClick 事件來指定用戶端指令碼的函式。而 HTML 控制項則使用 OnClick 事件。



動態註冊用戶端指令碼 — 1

除了直接撰寫用戶端指令碼之外，ASP.NET 也支援利用程式動態產生所需的用戶端指令碼，以下介紹其中兩種方式：

- **ClientScript.RegisterClientScriptBlock**

註冊用戶端指令碼至 form 標籤前端。使用該方法需加入的參數有註冊的用戶端指令碼型別、指令碼索引鍵、指令碼內容與指令碼的標籤建立與否等四種。如下所示：

```
Visual Basic
ClientScript.RegisterClientScriptBlock(this.GetType(), "1",
    "function DoClick1() {alert('Hello');}", true)
```

```
C#
ClientScript.RegisterClientScriptBlock(this.GetType(), "1",
    "function DoClick1() {alert('Hello');}", true);
```

- **ClientScript.RegisterStartupScript**

註冊用戶端指令碼至 form 標籤後端，該方法的參數型別與數量與 ClientScript.RegisterClientScriptBlock 相同。如下所示：

```
Visual Basic
ClientScript.RegisterStartupScript (this.GetType(), "2",
```

```
"function DoClick2() {alert('Hello');}", true)
```

C#

```
ClientScript. RegisterStartupScript (this.GetType(), "2",
    "function DoClick2() {alert('Hello');}", true);
```

動態註冊用戶端指令碼 — 2

• ClientScript.RegisterClientScriptInclude

註冊用戶端指令碼檔案

```
ClientScript.RegisterClientScriptInclude(this.GetType(), "3",
    "JScript.js");
```

動態註冊用戶端指令碼 — 2

程式動態產生用戶端指令碼，除了直接產生在網頁結果內之外，ASP.NET 也支援動態指定用戶端指令碼檔案，使用 ClientScript.RegisterClientScriptInclude 方法即可。使用該方法需加入的參數有註冊的用戶端指令碼型別、指令碼索引鍵、指令碼檔位址等三種。如下所示：

Visual Basic

```
ClientScript.RegisterClientScriptInclude(this.GetType(), "3",
    "JScript.js")
```

C#

```
ClientScript.RegisterClientScriptInclude(this.GetType(), "3",
    "JScript.js");
```

練習 3.4 : 動態註冊用戶端指令碼

•這個練習主要是讓您熟悉使用 ClientScriptManager 的 RegisterClientScriptBlock 、 RegisterStartupScript 與 RegisterClientScriptInclude 三種指令的用法，並且了解其不同之處。

•預估實作時間：15分鐘

練習 3.4 : 動態註冊用戶端指令碼

目的：

這個練習主要是讓您熟悉使用 ClientScriptManager 的 RegisterClientScriptBlock 、 RegisterStartupScript 與 RegisterClientScriptInclude 三種指令的用法，並且了解其不同之處。

功能描述：

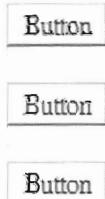
這個練習會在網頁中使用三個 Button 控制項，並且利用動態註冊用戶端指令碼的方式來供三個 Button 控制項的 OnClientClick 事件來呼叫，並檢視用戶端原始碼的差異之處。

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」，啟動 Visual Studio 2008 開發環境。

2. 從「File」→「New Web Site」→選取「Empty Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9543\Practices\VB 或 CS\Mod03_4\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod03_4」。
3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，檔名命名為「testClientScript.aspx」，清除「Place code in separate file」核取方塊，建立來源網頁。
4. 在「testClientScript.aspx」網頁內，從「Toolbox」工具箱中拖拉三個 Button 控制項，並調整適當位置，完成後應如下所示：



5. 自主選單「Web Site」下「Add New Item...」，選取「Jscript File」，檔名命名為「Jscript1.js」。
6. 在「Jscript1.js」內加入以下程式碼：

```
function function3() {
    alert('這是RegisterClientScriptInclude所建立的用戶端指令碼');
}
```

7. 回到「testClientScript.aspx」當案，在設計頁面對空白處連點滑鼠右鍵兩下，進入 Page_Load 事件處理常式，在程式碼區塊中加入下面的程式碼：

<p>Visual Basic</p> <pre>Button1.OnClientClick = "function1()" Button2.OnClientClick = "function2()" Button3.OnClientClick = "function3() ClientScript.RegisterClientScriptBlock(Me.GetType(), "1", "function function1() {alert('這是RegisterClientScriptBlock所建立的 用戶端指令碼'); return false;}", True) ClientScript.RegisterStartupScript(Me.GetType(), "2", "function function2() {alert('這是RegisterStartupScript所建立的 用戶端指令碼');}", True)</pre>
--

```
ClientScript.RegisterClientScriptInclude(Me.GetType(), "3",
    "JScript.js")
```

```
C#
Button1.OnClientClick = "function1();";
Button2.OnClientClick = "function2();";
Button3.OnClientClick = "function3();"

ClientScript.RegisterClientScriptBlock(this.GetType(), "1",
    "function function1() {alert('這是RegisterClientScriptBlock所建立的用戶端指令碼'); return false;}", true);

ClientScript.RegisterStartupScript(this.GetType(), "2",
    "function function2() {alert('這是RegisterStartupScript所建立的用戶端指令碼');}", true);

ClientScript.RegisterClientScriptInclude(this.GetType(), "3",
    "JScript.js");
```

8. 執行網頁測試。在「Solution Explorer」→點選「testCalendar.aspx」網頁→按滑鼠右鍵→選「View In Browser」。應該會發現三個 Button 在 PostBack 之前都會先跳出一個對話視窗，而每個視窗都會顯示其用戶端指令碼是由哪一個方法所建立出來的，如下所示：



9. 完成之後，可對網頁按下滑鼠右鍵選擇「View Source」來檢視原始碼，應該可以發現 RegisterStartupScript 與 RegisterClientScriptInclude 所產生的用戶端指令碼標籤宣告在 form 前端，而 RegisterClientScriptBlock 所產生的用戶端指令碼標籤宣告在 form 後端，如下所示：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>
```

```
</title></head>
<body>
    <form name="form1" method="post" action="testClientScript.
aspx" id="form1">
        <div>
            <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKLTcxMjQwMTA5NQ9kFgICBA9kFgYCAQ8PFgIeD
U9uQ2xpZW50Q2xpY2sFC2Z1bmN0aW9uMSgpZGQCAw8PFgIfAAUL
ZnVuY3Rpb24yKCIkZAIFDw8WAh8ABQtmW5jdGlvbjMoKWRkZBmK
izMFU2hKI0dwaNMo2o/B4iME" />
        </div>

        <script type="text/javascript">
//<![CDATA[
function function1() {alert('這是RegisterClientScriptBlock所建立的
用戶端指令碼'); return false;}//]]
        </script>

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

            <input type="hidden" name="__EVENTVALIDATION" id="__
EVENTVALIDATION" value="/wEWBAL5yOWGDAKM54rGBgK7q7GG
CALWIM+bArc5nAD5bqWrpmMEbyvW2QrEfap9" />
        </div>
        <div>

            <input type="submit" name="Button1" value="Button" onclick="function1();"
id="Button1" />
            <br />
            <br />

            <input type="submit" name="Button2" value="Button" onclick="function2();"
id="Button2" />
            <br />
            <br />
            <input type="submit" name="Button3" value="Button" onclick="function3();"
id="Button3" />

        </div>

        <script type="text/javascript">
//<![CDATA[
function function2() {alert('這是RegisterStartupScript所建立的用戶
端指令碼');}//]]
        </script>
    </form>
```

總結

- ASP.NET Web 伺服器控制項的特性
- 控制項的事件運用方式
- 各種常見的控制項使用方法
- ClientScript的註冊方式

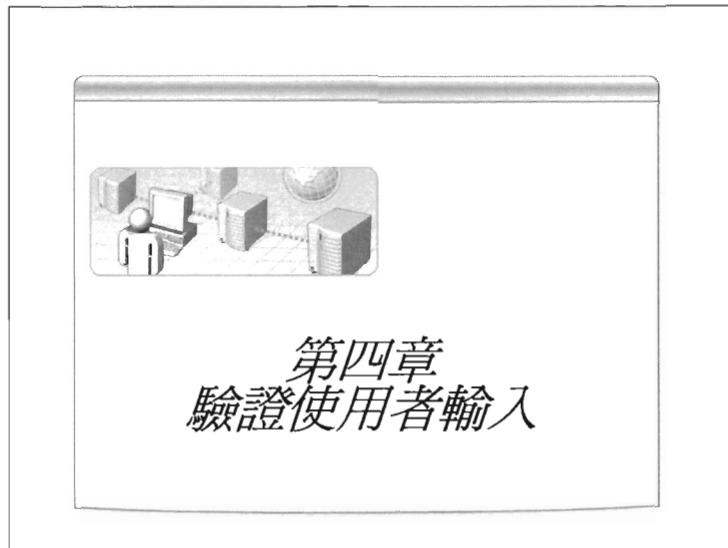
第四章: 驗證使用者輸入

本章大綱

課程大綱	3
關於 ASP.NET 的驗證使用者輸入	4
ASP.NET 驗證控制項的功能	5
ASP.NET 驗證控制項的特性	7
ASP.NET 驗證控制項的共有屬性	9
驗證控制項使用方式介紹	11
RequiredFieldValidator 控制項	12
RangeValidator 控制項	13
CompareValidator 控制項	14
RegularExpressionValidator 控制項	15
練習 4.1 : 使用驗證控制項 – 1	16
CustomValidator 控制項	20
練習 4.2 : 使用 CustomValidator 控制項	22
錯誤訊息的顯示方式	25
設計 ASP.NET 網頁驗證的注意事項	28
練習 4.3 : 驗證控制項的錯誤訊息與分組	30

作者：
周季賢





大綱

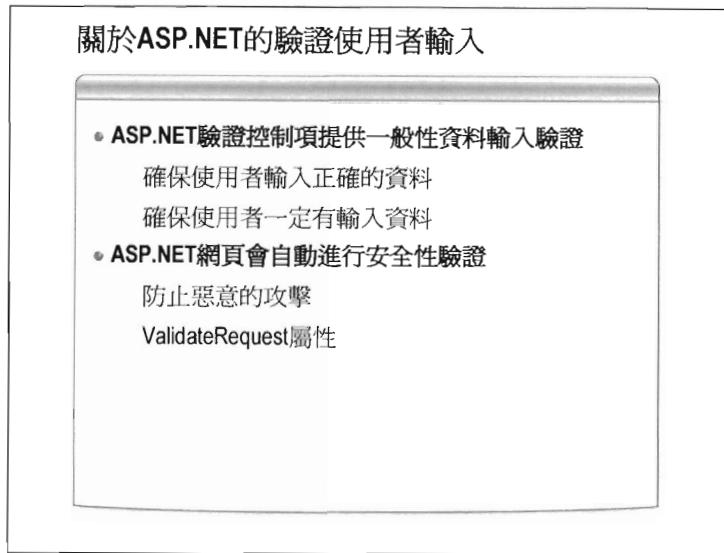
- 關於ASP.NET的驗證
- ASP.NET驗證控制項的功能
- ASP.NET驗證控制項的特性
- ASP.NET驗證控制項的共有屬性
- 驗證控制項使用方式
- 錯誤訊息顯示方式
- 設計ASP.NET網頁驗證的注意事項

課程大綱

在這個章節中將介紹 ASP.NET Web 應用程式驗證使用者輸入的方式。除了介紹各種 ASP.NET 所提供的驗證控制項使用方式之外，還會介紹當在網頁上實作輸入驗證時，所需注意的各種細節

本章介紹以下主題：

- 關於 ASP.NET 的驗證使用者輸入
- ASP.NET 驗證控制項的功能
- ASP.NET 驗證控制項的特性
- ASP.NET 驗證控制項的共有屬性
- 驗證控制項使用方式
- 錯誤訊息顯示方式
- 設計 ASP.NET 網頁驗證的注意事項



關於 ASP.NET 的驗證使用者輸入

在 ASP.NET 的驗證使用者輸入部分，分為兩個範疇：

ASP.NET 驗證控制項提供一般性資料輸入驗證

對於一般性的資料輸入，例如使用者的個人資料註冊或是登入時的帳號密碼等，ASP.NET 提供了一系列的驗證控制項來應對各種所需的情境。這種一般性的資料輸入的驗證，通常是為了確保使用者輸入正確的資料或是確保使用者一定有輸入資料行為，而這也是本章所欲介紹的內容。

ASP.NET 網頁會自動進行安全性驗證

使用者除了一般性的資料輸入之外，也有可能會利用資料輸入時的漏洞，而惡意輸入攻擊性的資料來破壞網站的安全。而 ASP.NET Web 網頁預設即會自動檢查潛在的惡意輸入，當 ASP.NET 偵測到使用者輸入可能的危險字串時(如指令碼)，就會自動產生安全性錯誤訊息。

如果要關閉此預設功能，可對網頁 Page 指示詞的 ValidateRequest 屬性設定為 false。

ASP.NET驗證控制項的功能

- 必要欄位驗證
RequiredFieldValidator
- 數值範圍限制
RangeValidator
- 輸入內容比較
CompareValidator
- 檢查格式
RegularExpressionValidator
- 自行定義驗證內容
CustomValidator

ASP.NET 驗證控制項的功能

ASP.NET 提供了五個驗證控制項，分別針對了五種使用者輸入的驗證行為：

- 必要欄位驗證

RequiredFieldValidator 提供了使用者是否有輸入必要欄位的檢查，強制使用者在送出資料時，必須在必要欄位輸入值。如註冊網頁的使用者帳號與密碼欄位。

- 數值範圍限制

RangeValidator 可以檢查使用者的輸入是否介於指定的上下限之間。可檢查的對象有數字、英文字母字元和日期內的範圍。如使用者年齡輸入的限制。

- 輸入內容比較

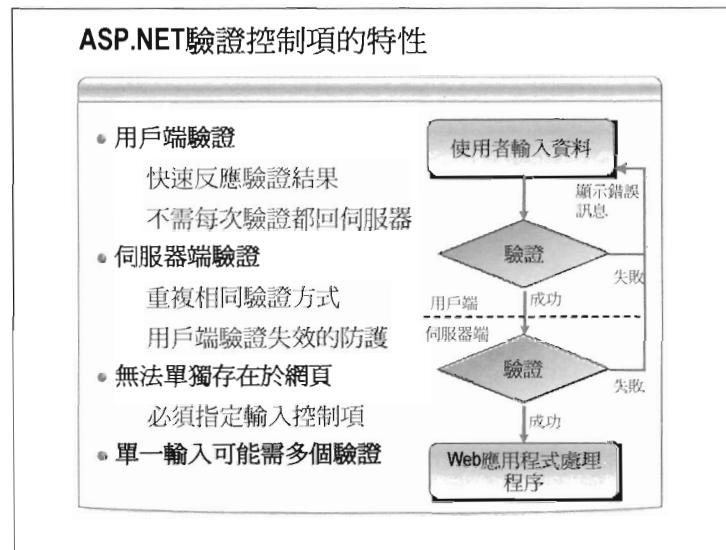
CompareValidator 可以將使用者輸入值與指定的常數值或是另一個控制項的值來做比較，比較雙方之間是否小於、等於或大於，進而決定驗證成功與否。如註冊網頁的密碼與確認密碼欄位。

- 檢查格式

RegularExpressionValidator 使用規則運算式 (Regular Expression) 來定義使用者輸入的資料樣式。可讓您檢查特殊格式的資料輸入，如電子郵件地址。

- **自行定義驗證內容**

CustomValidator 提供了使用者自行定義驗證規則的便利性，在上述驗證控制項都無法做到所需的驗證需求時，可使用該控制項來自訂驗證規則。如輸入資料必須與資料庫比對。



ASP.NET 驗證控制項的特性

ASP.NET 所提供的驗證控制項，均同時具有用戶端與伺服器端的驗證，利用用戶端與伺服器端的驗證，來達到不同的驗證需求：

用戶端驗證

如果用戶端有啓用指令碼的使用，那麼 ASP.NET 驗證控制項就可以使用用戶端指令碼執行驗證，這麼一來控制項可立即回應驗證成功與否而不需 PostBack，使用者將可感受到網頁執行效率的改善。

另外，也因為在前端立即驗證而不需在伺服器間來回，所以可以大幅的減少與伺服器的溝通次數，如此便可減輕伺服器的負擔。

伺服器端驗證

ASP.NET 驗證控制項的伺服器端驗證會提供與用戶端驗證邏輯相同的驗證行為；這種乍看之下似乎是畫蛇添足的伺服器端驗證，其實是可以在用戶端不支援或關閉指令碼而造成用戶端驗證失效的狀態下，提供第二層的防護。

無法單獨存在於網頁

ASP.NET 驗證控制項的存在目的就是為了驗證使用者輸入；所以當使用 ASP.NET 驗證控制項時，網頁上勢必至少要有一個輸入控制項來作為驗證對象的指定，如果網頁上只有 ASP.NET 驗證控制項或是加入 ASP.NET 驗證控制項卻沒有為它指定輸入控制項，屆時網頁執行便會出現產生例外錯誤。

單一輸入可能需要多個驗證

每一個 ASP.NET 驗證控制項的驗證功能都是單一的，所以當輸入欄位所需要的驗證目的不只一種，那麼就可能需要使用多個驗證控制項來對應，如電子郵件地址欄位也是必要欄位，就必須要同時使用 RequiredFieldValidator 與 RegularExpressionValidator 來對該欄位做驗證。



ASP.NET 驗證控制項的共有屬性

在使用 ASP.NET 驗證控制項之前，當然要對其屬性設定做個了解。ASP.NET 驗證控制項各自的功能，都提供了獨有的屬性，另外，共有屬性的部分，也都是在設計時不可不知的，以下分別說明共有屬性。

- **ControlToValidate**

要檢查的輸入控制項 ID，也是必要屬性，如加入驗證控制項而沒有設定此屬性，網頁在執行時便會產生例外錯誤。

- **ErrorMessage**

當驗證失敗時，要提供給使用者的錯誤訊息內容。

- **Display**

驗證控制項所在位置的顯示行為，此屬性有三種選擇 None、Static 與 Dynamic。選擇 None 則不顯示，Static 會為驗證控制項在網頁上保留空間，不管驗證成功與否，而 Dynamic 則是動態配置，如果驗證成功則不保留空間。

- **EnableClientScript**

是否要啓動用戶端驗證，該設定可自行決定是否關閉驗證控制項的用戶端驗證。

- SetFocusOnError

該屬性可決定當驗證失敗時，使用者的游標是否要停回驗證控制項所指定的輸入欄位內。

驗證控制項使用方式介紹

- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator

驗證控制項使用方式介紹

本小節將要為您介紹 ASP.NET 驗證控制項的使用方式，包括了下列五種驗證控制項。

- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator



RequiredFieldValidator 控制項

如果想要限制使用者一定要輸入資料，在 ASP.NET，可以使用 RequiredFieldValidator 控制項來檢查使用者是否輸入資料。例如：在登入網頁內，「帳號」和「密碼」欄位是一定要輸入的欄位，這兩個欄位最好就要加上 RequiredFieldValidator 控制項來限制使用者的輸入。

RequiredFieldValidator 的特殊屬性如下：

- InitialValue :

所指定的驗證控制項不允許的值，預設為空白。當被驗證的控制項具備預設值(例如：「未設定」)，你希望使用者輸入預設值以外的內容時，可將 InitialValue 的值設定為預設值，如此使用者就必須要挑選預設值以外的內容才可通過驗證。

需特別注意的是，預設只有 RequiredFieldValidator 控制項能檢查使用者有無對該欄位輸入值；RequiredFieldValidator 之外的其它控制項，預設在所檢查的欄位並無輸入時，是不會啓動驗證的。

RangeValidator 控制項

- 用來檢查使用者輸入的資料的是否在範圍之內

- 屬性

MinimumValue：最小值

MaximumValue：最大值

Type：資料型別

```
<asp:RangeValidator  
    ControlToValidate="TextBox2"  
    ErrorMessage="必須介於10-80歲"  
    MaximumValue="80"  
    MinimumValue="10"  
    Type="Integer">  
</asp:RangeValidator>
```

RangeValidator 控制項

RangeValidator 控制項可用來檢查使用者輸入的值是否在系統可以接受的範圍。例如：檢查使用者輸入工作進度百分比，你可以透過 RangeValidator 設定最小值是 1，最大值是 100，Type 設定為 Integer。

RangeValidator 的特殊屬性如下：

- MinimumValue：

設定範圍最小值。

- MaximumValue：

設定範圍最大值。

- Type：

設定值的資料型別。例如要檢查輸入數量資料，則必須設定為 Integer，若為預設的 String 型別，就可能造成判斷錯誤。



CompareValidator 控制項

CompareValidator 用來比較使用者輸入的資料。例如：使用者建立新帳號時，帳號的「密碼」欄位和「確認密碼」欄位所輸入的內容是否一致。

CompareValidator 的特殊屬性如下：

- ValueToCompare：

設定要比較的值。

- ControlToCompare：

設定要比較的控制項。

- Type：

設定值的資料型別 ~

- Operator：

設定比較子，比較子的內容以英文呈現，有 Equal、
NoEqual、GreaterThan，GreaterThanOrEqual、DataTypeCheck…
等。

RegularExpressionValidator 控制項

- 限制使用者輸入字元格式的控制項
- 利用 RegularExpression 限制格式
- 屬性：

ValidationExpression：設定 RegularExpression 語法

```
<asp:RegularExpressionValidator  
    ID="RegularExpressionValidator1"  
    runat="server"  
    ControlToValidate="TextBox5"  
    ErrorMessage="格式不正確"  
    ValidationExpression="\w+([.-]\w+)*@\w+([.-]  
.]\w+)*.\w+([.-]\w+)*">  
</asp:RegularExpressionValidator>
```

RegularExpressionValidator 控制項

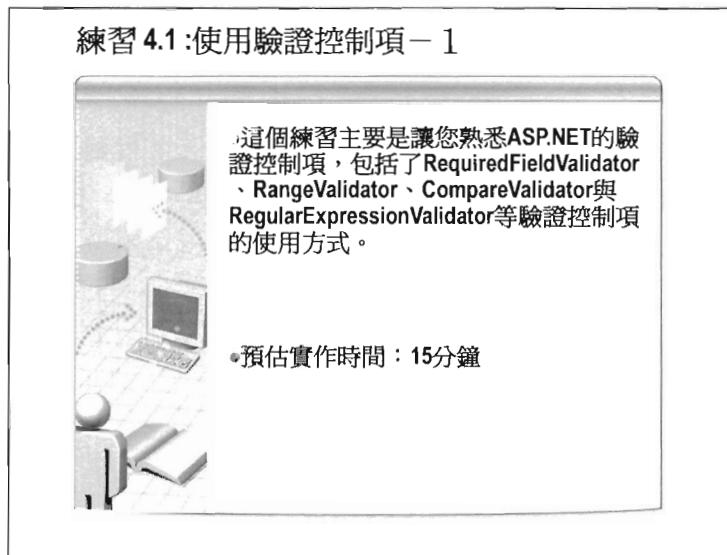
在 ASP.NET，可以利用 RegularExpressionValidator 控制項來檢查使用者的輸入是否符合預先定義的樣式。RegularExpressionValidator 使用規則運算式 (Regular Expression)來定義使用者輸入的資料樣式。

RegularExpressionValidator 的特殊屬性如下：

- ValidationExpression：

用來設定 RegularExpression 的規則語法。

規則運算式 (Regular Expression)的語法是相當複雜的，但是再開發工具內有提供一些常用的規則語法供開發者選擇。



練習 4.1 : 使用驗證控制項 – 1

目的：

這個練習主要是讓您熟悉 ASP.NET 的驗證控制項，包括了 RequiredFieldValidator 、 RangeValidator 、 CompareValidator 與 RegularExpressionValidator 等驗證控制項的使用方式

功能描述：

這個練習會在網頁中 RequiredFieldValidator 、 RangeValidator 、 CompareValidator 與 RegularExpressionValidator 等驗證控制項來針對使用者註冊時所填寫的資料來做檢查。

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「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod04_1\Starter」目錄，完成後，按下「Open」按鈕即可開啟網站。
3. 在「testValidator.aspx」網頁內，從「Toolbox」工具箱中拖拉兩個 RequiredFieldValidator 檢查項至 TextBox1 與 DropDownList1 檢查項右邊
4. 從「Toolbox」工具箱中拖拉一個 RangeValidator 檢查項至 TextBox2 右邊
5. 從「Toolbox」工具箱中拖拉一個 CompareValidator 檢查項至 TextBox4 右邊
6. 從「Toolbox」工具箱中拖拉一個 RegularExpressionValidator 檢查項至 TextBox5 右邊。
7. 接著設定驗證檢查項的屬性，檢查項的名稱屬性如下所示：

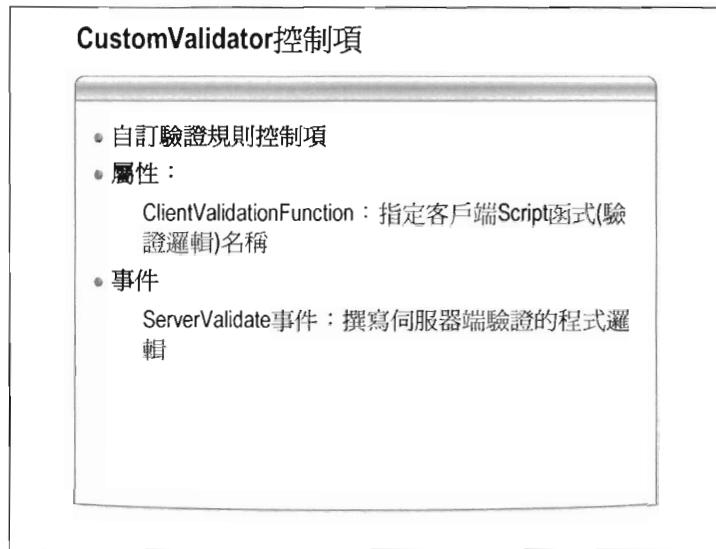
控制項	屬性	屬性值
RequiredField Validator	ID	RequiredFieldValidator1
	ControlToValidate	TextBox1
	ErrorMessage	帳號必須輸入
RequiredField Validator	ID	RequiredFieldValidator2
	ControlToValidate	瀏覽器重新導向
	ErrorMessage	居住地必須選擇
RangeValidator	ID	RangeValidator1
	ControlToValidate	TextBox2
	ErrorMessage	必須介於 10-80 歲
	MinimumValue	10
	MaximumValue	80
	Type	Integer
CompareValidator	ID	CompareValidator1
	ControlToValidate	TextBox4
	ErrorMessage	密碼與確認密碼需一致
	ControlToCompare	TextBox3
RegularExpressionValidator	ID	RegularExpressionValidator1
	ControlToValidate	TextBox5
	ErrorMessage	格式不正確
	ValidationExpression	Internet e-mail address

完成後樣式如下：

使用者帳號：	帳號必須輸入
居住地：	未選擇 ▼ 居住地必須選擇
年齡：	必須介於10-80歲
密碼：	
確認密碼：	密碼與確認密碼需一致
信箱：	格式不正確
	<input type="button" value="註冊"/>

8. 執行網頁測試。在「Solution Explorer」→點選「testValidator.aspx」網頁→按滑鼠右鍵→選「View In Browser」。在網頁上故意輸入會引發驗證失敗的資料，按下註冊按鈕時，應可見到所有驗證失敗的訊息。如下所示：

使用者帳號:	帳號必須輸入	
居住地:	未選擇	居住地必須選擇
年齡:	1	必須介於10-80歲
密碼:	1	
確認密碼:	3	密碼與確認密碼需一致
信箱:	123	格式不正確
<input type="button" value="註冊"/>		



CustomValidator 控制項

當所有的驗證控制項都無法滿足需求時，最後可以使用的就是 CustomValidator 控制項，使用 CustomValidator 控制項時，必須自行定義用戶端和伺服器端的程式處理邏輯。伺服器端使用 CustomValidator 控制項的 ServerValidate 事件觸發驗證的動作。用戶端的資料檢查，則需要先撰寫前端 Script 函式，再將 ClientValidationFunction 屬性設成該函式名稱。

RegularExpressionValidator 的特殊屬性如下：

- ClientValidationFunction :

指定客戶端 Script 函式(驗證邏輯)名稱。使用者必須要先自行先利用戶端程式碼撰寫對應的函式，完成之後再將函式的名稱設定至該屬性即可。用戶端程式碼對應的函式如下所示：

```
<script type="text/javascript">
function Client_Validate(sender,e)
{
    if (e.Value % 2 ==0)
        e.IsValid=true;
    else
        e.IsValid=false;
}
</script>
```

RegularExpressionValidator 的特殊事件如下：

- ServerValidate 事件：

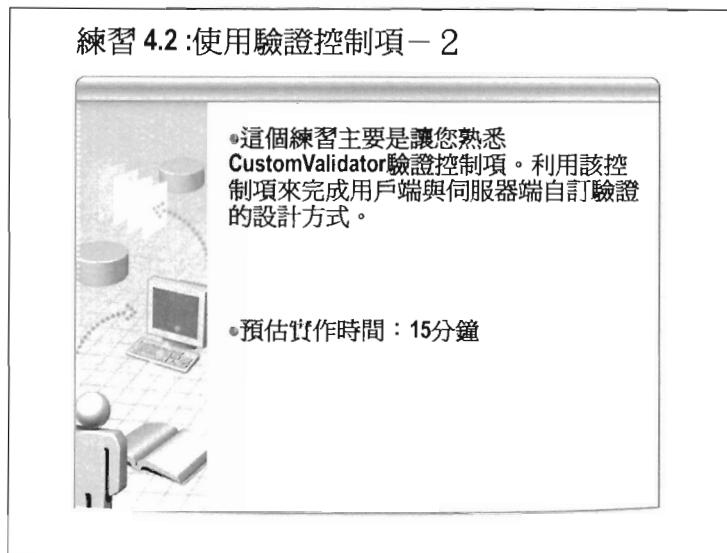
撰寫伺服器端驗證的程式邏輯，在對應的事件處理常式內，會有一個 System.Web.UI.WebControls.ServerValidateEventArgs 型別的參數，該參數提供兩個屬性：Value 與 IsValid；Value 為驗證的輸入控制項內的值，IsValid 則可決定是否驗證成功。如下所示：

Visual Basic

```
Protected Sub CustomValidator1_ServerValidate(ByVal source As Object, ByVal args As System.Web.UI.WebControls.ServerValidateEventArgs)
    If Integer.Parse(args.Value) Mod 2 = 0 Then
        args.IsValid = True
    Else
        args.IsValid = False
    End If
End Sub
```

C#

```
protected void CustomValidator1_ServerValidate(object sender, EventArgs args)
{
    if (int.Parse(args.Value) % 2 == 0)
        args.IsValid = true;
    else
        args.IsValid = false;
}
```



練習 4.2：使用 **CustomValidator** 控制項

目的：

這個練習主要是讓您熟悉 CustomValidator 驗證控制項。利用該控制項來完成用戶端與伺服器端自訂驗證的設計方式。

功能描述：

在這個練習中，將使用 CustomValidator 驗證控制項。利用該控制項的用戶端與伺服器端自訂驗證的設計方式，來檢查使用者輸入值是否為偶數。

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『Empty Web Site』→將『Location』設為『File System』並點選『Browse...』按鈕選取『\U9543\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」，檔名命名為「testCustomValidator.aspx」，清除「Place code in separate file」核取方塊，建立來源網頁。
4. 在「testCustomValidator.aspx」網頁內，從「Toolbox」工具箱中拖拉一個 TextBox 控制項、一個 CustomValidator 控制項與一個 Button 控制項。控制項的名稱屬性如下所示：

控制項	屬性	屬性值
TextBox	ID	TextBox1
CustomValidator	ID	CustomValidator1
	ClientValidationFunction	Client_Validate
	ControlToValidate	TextBox1
	ErrorMessage	必須為偶數
Button	ID	Button1
	Text	檢查

完成後設計頁面應如下所示：



5. 撰寫用戶端程式碼。請在<head>到</head>標籤內，撰寫以下的 Javascript 程式碼。

```
<script type="text/javascript">
    function Client_Validate(sender,e)
    {
        if (e.Value % 2 ==0)
            e.IsValid=true;
        else
            e.IsValid=false;
    }
</script>
```

6. 測試用戶端驗證。在「Solution Explorer」→點選「testCustomValidator.aspx」網頁→按滑鼠右鍵→選「View In Browser」。分別輸入 3 和 28。當輸入 3 的時候，由於違反驗證規則，所以，CustomValidator 控制項會出現錯誤訊息，如下圖：



7. 對 CustomValidator1 控制項連點滑鼠右鍵兩下，進入 CustomValidator1_ServerValidate 事件處理常式，在程式碼區塊中加入下面的程式碼：

Visual Basic

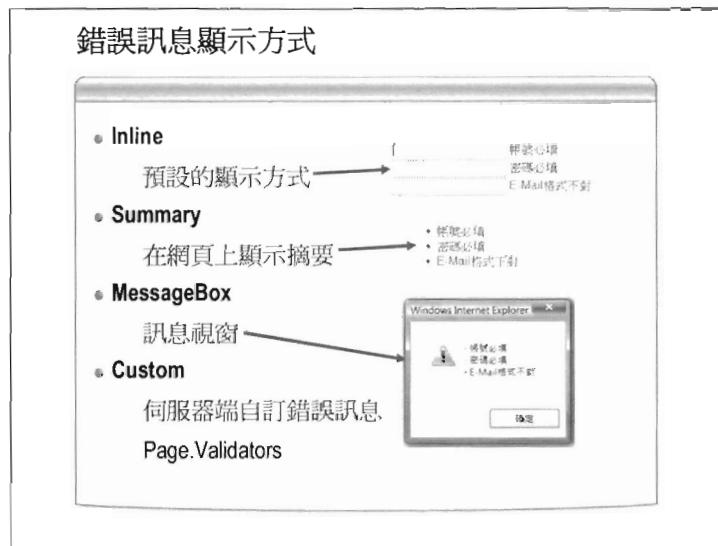
```
Protected Sub CustomValidator1_ServerValidate(ByVal source As Object, ByVal args As System.Web.UI.WebControls.ServerValidateEventArgs)
    If Convert.ToDecimal(args.Value) Mod 2 = 0 Then
        args.IsValid = True
    Else
        args.IsValid = False
    End If
End Sub
```

C#

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    if (Convert.ToDecimal(args.Value) % 2 == 0)
        args.IsValid = true;
    else
        args.IsValid = false;
}
```

8. 測試伺服器端驗證。點選 CustomValidator1 控制項，將其 EnableClientScript 屬性設定為 false。接著在「Solution Explorer」→點選「testCustomValidator.aspx」網頁→按滑鼠右鍵→選「View In Browser」。分別輸入 3 和 28。當輸入 3 的時候，由於違反驗證規則，所以，CustomValidator 控制項在 Postback 之後同樣會出現錯誤訊息。請參考下圖：





錯誤訊息的顯示方式

在 ASP.NET 的錯誤訊息的顯示部分，有下列四種方式可供設計參考。

Inline

預設的顯示方式，也就是將錯誤訊息顯示在各驗證控制項當初設計時所配置的位置。使用時要注意 Display 屬性對於訊息顯示與否的空間占用問題。

Summary

在網頁上顯示摘要。該方式可使用 ValidationSummary 控制項；在網頁上適當位置加入該控制項之後，確定 ShowSummary 屬性設定為 true，在執行時期，該控制項便會主動的去取得網頁上所有驗證失敗的驗證控制項錯誤訊息，並顯示在 ValidationSummary 控制項所配置的位置內。

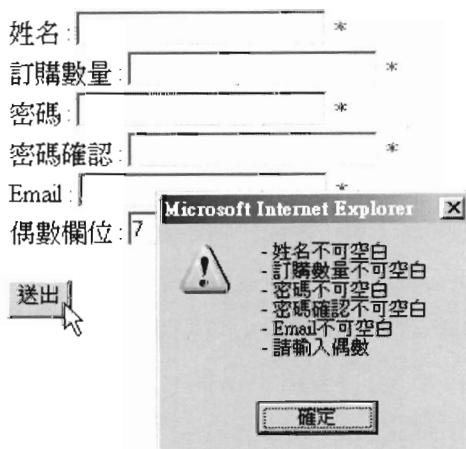
另外需注意的是，使用 ValidationSummary 控制項時，對驗證控制項的設定，最好能同時設定 Text 屬性及 ErrorMessage 屬性，其中，Text 屬性決定驗證控制項在網頁上的顯示內容，通常會設定為「*」。

符號，用來標示驗證未通過驗證檢查的欄位，而 ErrorMessage 屬性則是提供給 ValidationSummary 控制項使用。屆時得到的結果如下所示：

- 姓名不可空白
- 訂購數量不可空白
- 密碼不可空白
- 密碼確認不可空白
- Email不可空白
- 請輸入偶數

MessageBox

訊息視窗，也是使用 ValidationSummary 控制項；在網頁上加入該控制項之後，確定 ShowMessageBox 屬性設定為 true，在執行時期，該控制項便會主動的去取得網頁上所有驗證失敗的驗證控制項錯誤訊息。並主動的跳出訊息視窗，並在視窗內顯示所有的錯誤訊息。屆時得到的結果如下所示：



Custom

伺服器端自訂錯誤訊息顯示方式，當以上的訊息顯示方式都無法滿足您的需求時，可以使用自訂的方式來顯示，該方法必須要自行決定外觀的顯示方式與自行取得驗證失敗的驗證控制項錯誤訊息。

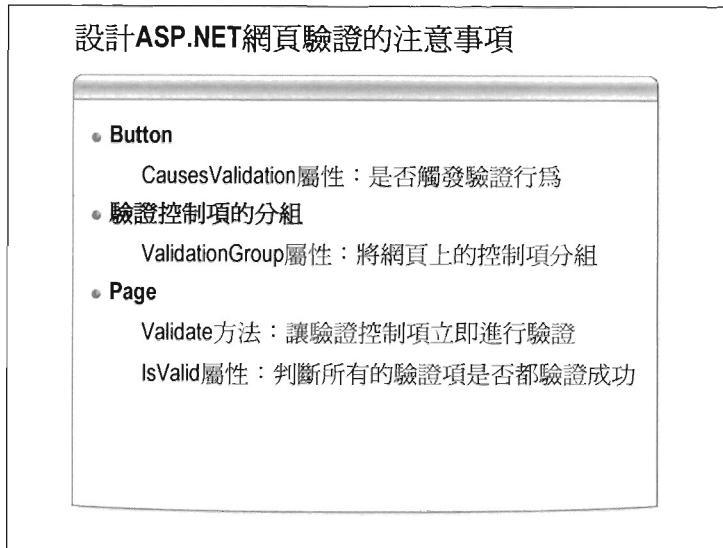
外觀所需的控制項就因需求而異，而自行取得驗證失敗的驗證控制項錯誤訊息方法則是在伺服器事件內，使用 `PageValidators` 集合來取得所有的驗證控制項，並進一步的針對 `PageValidators` 集合內的每一個驗證控制項成員的 `IsValid` 屬性來判斷是否驗證失敗，如果驗證失敗，便可自行將其 `ErrorMessage` 取出在顯示至控制項即可。程式碼如下所示：

Visual Basic

```
For Each vtor As IValidator In PageValidators
    If (Not vtor.IsValid) Then
        Response.Write(vtor.ErrorMessage)
    End If
Next
```

C#

```
foreach (IValidator vtor in thisValidators)
{
    if (!vtor.IsValid)
    {
        Response.Write(vtor.ErrorMessage);
    }
}
```



設計 ASP.NET 網頁驗證的注意事項

除了了解 ASP.NET 所提供的驗證控制項的使用與錯誤訊息的顯示之外，在開發 ASP.NET 網頁驗證時，還有些不可不知的注意事項。

Button

Button 控制項的 CausesValidation 屬性可決定是否觸發驗證控制項的檢查邏輯。例如「送出」按鈕與「取消並回首頁」按鈕，應該只有「送出」按鈕應該與網頁上的驗證有關，但是預設所有按鈕被點選時都會觸發驗證控制項檢查機制，所以此時可以利用 Button 控制項的 CausesValidation 屬性，將「取消並回首頁」按鈕的 CausesValidation 設定為 False，這樣就不會觸發驗證機制。

驗證控制項的分組

當網頁功能更為複雜時，可能會有兩張輸入表單提供使用者填寫，有可能有兩個送出按鈕，但是必須分別觸發兩張表單各自的驗證控制項，此時可以設定同一張表單的所有控制項（包括 TextBox 控制項、驗證控制項與按鈕控制項）的 ValidationGroup 屬性為同名，這樣就

能有所區分，如下圖的第二個按鈕控制項只會觸發自己表單的驗證機制：

姓名：

訂購數量：

密碼：

密碼確認：

Email：

偶數欄位：

送出 取消並回首頁

訂閱電子報電子郵件： 訂閱電子報電子郵件不可空白

送出

Page

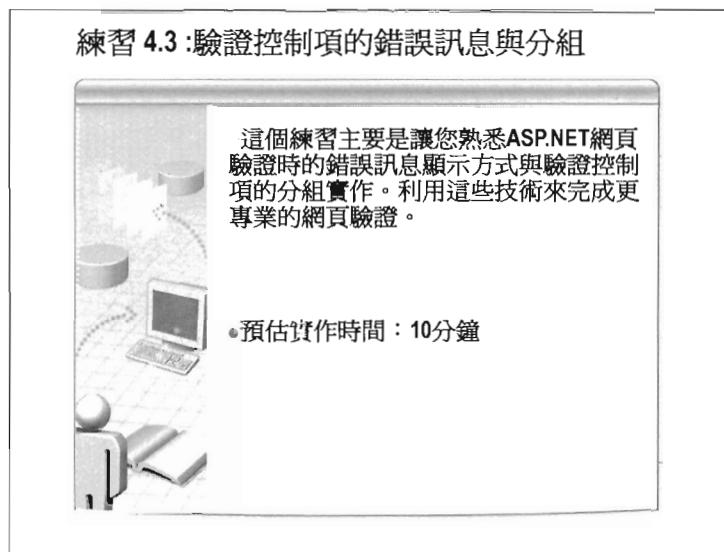
在開發 ASP.NET 網頁驗證時，也需要知道 Page 物件內兩個成員的用法：

- Validate 方法：

Validate 方法允許開發人員在任何時候，只要有需要做資料驗證，便可呼叫該方法，立即讓所有的驗證控制項或驗證群組進行資料驗證。例如希望在 Page_Load 事件內得知伺服器端驗證成功與否，就需要呼叫該方法。

- IsValid 屬性：

伺服器端的程式可使用 Page 的 IsValid 屬性來確認所有驗證控制項是否驗證通過。如果少了這個判斷，網頁就只剩客戶端驗證的功能。



練習 4.3 :驗證控制項的錯誤訊息與分組

目的：

這個練習主要是讓您熟悉 ASP.NET 網頁驗證時的錯誤訊息顯示方式與驗證控制項的分組實作。利用這些技術來完成更專業的網頁驗證。

功能描述：

這個練習是準備好一張已經完成網頁驗證的 ASP.NET 網頁，在此網頁上調整錯誤訊息的輸出，以及設定驗證控制項分組的功能。

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『File System』按鈕並選取『U9543\Practices\VB 或 CS\Mod04_3\Starter』目錄，完成後，按下『Open』按鈕即可開啟網站。

3. 在「Default.aspx」網頁內，從「Toolbox」工具箱中拖拉一個 ValidationSummary 控制項至「儲存」按鈕的下方
4. 修改所有驗證控制項的 Text 屬性，將其修改成「*」。請參考下圖：

姓名 *

工作年數 *

密碼 *

確認密碼

電子郵件 *

- 錯誤訊息 1。
- 錯誤訊息 2。

輸入偶數 *

5. 測試驗證訊息顯示。在「Solution Explorer」→點選「Default.aspx」網頁→按滑鼠右鍵→選「View In Browser」。當使用者輸入的資料格式錯誤時，錯誤訊息會在 ValidationSummary 控制內，而驗證控制項則是顯示「*」符號。請參考下圖：

姓名	<input type="text"/>	*
工作年數	<input type="text" value="99"/>	*
密碼	<input type="password" value="1"/>	*
確認密碼	<input type="password" value="2"/>	
電子郵件	<input type="text" value="444"/>	*

- 姓名不可空白
- 請輸入0-50之間的整數
- 密碼和確認密碼不一致
- 格式錯誤
- 請輸入偶數

*

6. 測試完成之後，會發現不管是按下「儲存」按鈕還是「確認」按鈕，均會觸發網頁上的所有驗證，所以接著就是要替網頁上驗證控制項作分組，分組以輸入偶數上方的橫線做為依據，以上一組，以下一組。
7. 將輸入偶數上方橫線上的所有控制項的 ValidationGroup 屬性設定為「group1」。
8. 將輸入偶數下方橫線上的所有控制項的 ValidationGroup 屬性設定為「group2」。
9. 測試驗證訊息顯示。在「Solution Explorer」→點選「Default.aspx」網頁→按滑鼠右鍵→選「View In Browser」。當使用者輸入的資料格式錯誤時，按下「儲存」按鈕，錯誤訊息僅會針對該群組作顯示。請參考下圖：

姓名 *

工作年數 100

密碼 2 *

確認密碼 221

電子郵件 333 *

儲存

- ◆ 姓名不可空白
- ◆ 密碼和確認密碼不一致
- ◆ 格式錯誤

輸入偶數 1

確認

10. 當使用者輸入的資料格式錯誤時，按下「確認」按鈕，錯誤訊息僅會針對該群組作顯示。請參考下圖：

姓名

工作年數 100

密碼 2

確認密碼 221

電子郵件 333

儲存

輸入偶數 1 *

確認

總結

- 關於ASP.NET的驗證控制項
- 使用驗證控制項
- 設計錯誤訊息顯示
- ASP.NET網頁驗證注意事項

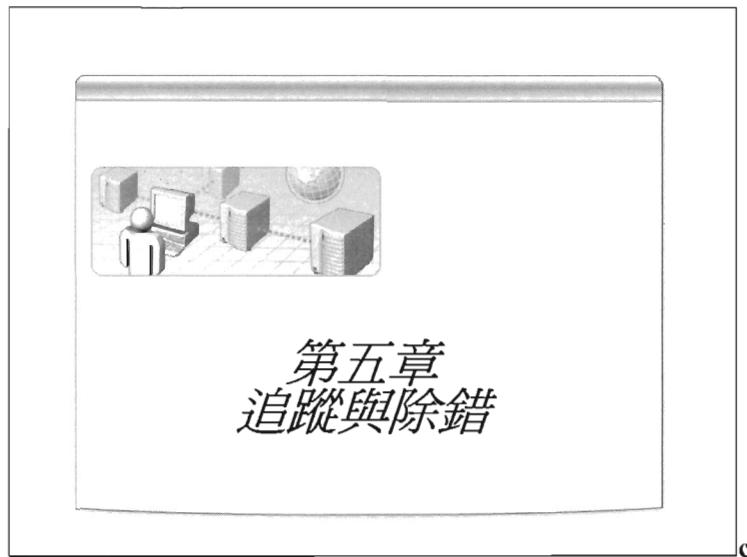
第五章：追蹤與除錯

本章大綱

課程大綱.....	3
應用程式中常見的錯誤種類.....	4
Visual Studio 支援的除錯種類.....	6
伺服器端程式除錯.....	7
練習 5.1：設定中斷點.....	10
用戶端指令碼除錯.....	13
應用程式追蹤.....	15
.NET Framework 應用程式追蹤.....	17
練習 5.2：使用 Debug 物件.....	19
ASP.NET 網頁追蹤.....	21
網頁層級追蹤.....	23
應用程式層級追蹤.....	25
練習 5.3：執行頁面層級追蹤.....	27
自訂輸出.....	30

作者：
周季賢





c

大綱

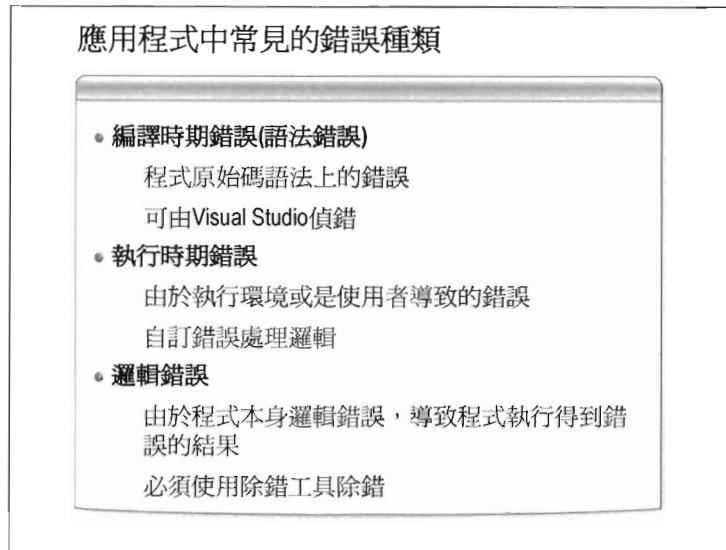
- 應用程式中常見的錯誤種類
- Visual Studio 支援的除錯種類
- 伺服器端程式除錯
- 用戶端指令碼除錯
- 應用程式追蹤
- 總結

課程大綱

在這個章節中將介紹 ASP.NET Web 應用程式在開發階段的各種除錯方式與應用程式執行狀況的追蹤方法。

本章介紹以下主題：

- 應用程式中常見的錯誤種類
- Visual Studio 支援的除錯種類
- 伺服器端程式除錯
- 用戶端指令碼除錯
- 應用程式追蹤



應用程式中常見的錯誤種類

開發應用程式時，難免會發生錯誤。不同的錯誤發生的原因不同，因此解決的方式也不相同。應用程式常發生的錯誤可以分為三類：編譯時期錯誤、執行時期錯誤、與邏輯錯誤。

編譯時期錯誤

在撰寫程式時，可能會因為對於程式語法不熟悉，或是因為打字上的疏忽，而造成程式原始碼語法上的錯誤。這一些錯誤在編譯程式時，可以由程式語言的編譯器檢查出來，因此稱之為「編譯時期錯誤」。

在 Visual Studio 開發工具中，提供了很多聰明的機制，來預防開發人員產生編譯時期錯誤。像是在使用變數或是物件類別時，只需要輸入名稱的前幾個字，再按下 ALT 加 → (向右鍵)，Visual Studio 開發工具就會幫你把變數或是類別的名稱自動完成。如果要使用物件的成員，只需要在輸入物件名稱後，按下英文句號，IntelliSense 就會自動列出物件所有可以使用的成員，這一些功能都可以用來避免開發人員產生編譯時期錯誤。

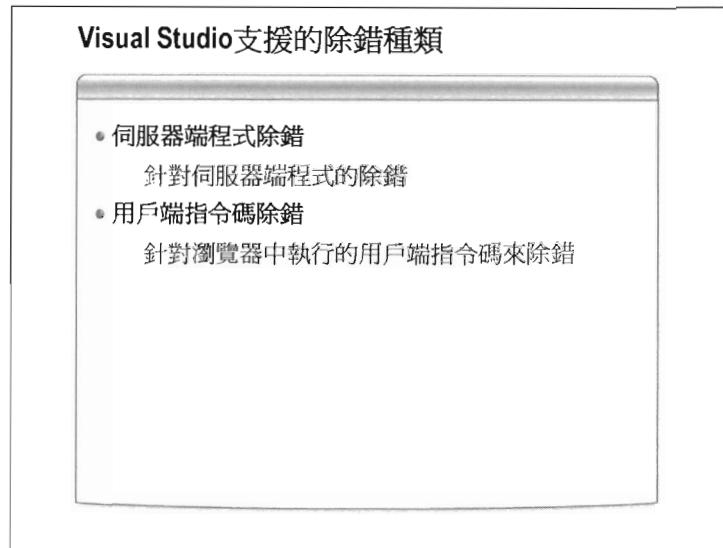
執行時期錯誤

程式編譯完成之後，並不能保證一定可以執行出正確的結果。因為程式在執行時期，經常需要根據使用者所輸入的資料做運算，或是存取像是檔案、資料庫等外部資源。而當成是在執行時期做這一些運算時，很容易就會因為使用者輸入了錯誤的資料，或是因為權限不足等原因，導致程式在執行時期無法正常的執行。此時 ASP.NET 執行環境會產生 `Exception` 物件，若是沒有撰寫處裡例外狀況的程式的話，使用者就會在瀏覽器中看到錯誤訊息。

在執行時期所發生的錯誤，通常編譯的時候是看不出來的，因此需要透過除錯工具追蹤或是程式的記錄檔進行除錯。ASP.NET 執行環境比一般的應用程式又更加複雜，同時間可能會有很多使用者一起執行，因此要除錯就更加的困難。ASP.NET 提供了追蹤的機制，可以用來追蹤頁面的執行狀態。如果想要自己設定執行時期追蹤的條件，也可以利用 `Trace` 物件撰寫程式進行執行時期的追蹤。

邏輯錯誤

有些時候，程式編譯時沒有問題，也可以正常的執行，但是產生的結果卻是錯誤的。通常這一類錯誤是因為開發人員在撰寫程式時，邏輯上的錯誤所造成的。邏輯上的錯誤相當不容易除錯，因此需要透過除錯工具從程式的流程開始逐行進行追蹤。



Visual Studio 支援的除錯種類

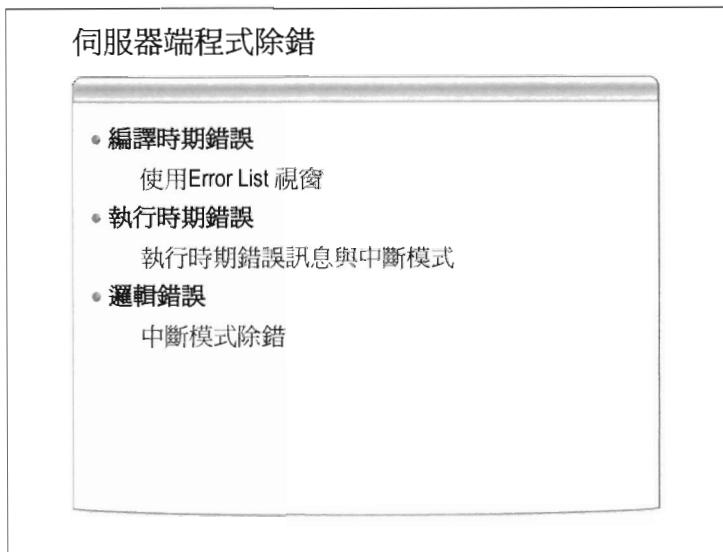
Visual Studio 提供了在開發 ASP.NET Web 應用程式時的各種除錯方式，如下所示：

- **伺服器端程式除錯**

針對伺服器端程式的除錯，支援性最高，Visual Studio 提供各式工具視窗來針對伺服器程式在執行時期的狀態監控與處理。

- **用戶端指令碼除錯**

針對瀏覽器中執行的用戶端指令碼來除錯，當瀏覽器在執行網頁的用戶端指令碼時，可以利用 Visual Studio 來監控用戶端指令碼的執行狀態與相關的變數資訊。

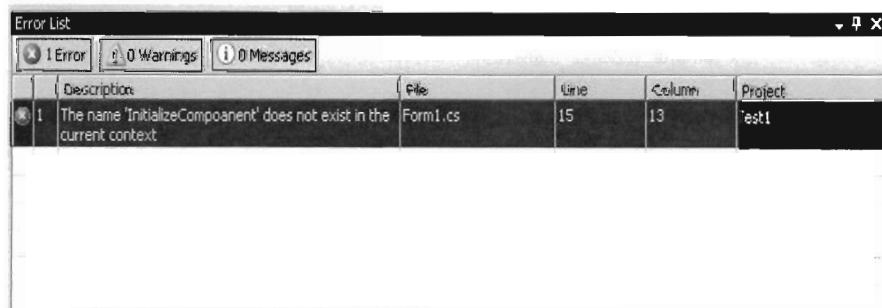


伺服器端程式除錯

Visual Studio 在針對伺服器程式的除錯功能上支援性最齊全的，以下就針對各種錯誤種類來介紹除錯的工具與方式。

編譯時期錯誤

若是在編譯程式時有語法上的錯誤，Visual Studio 開發工具會自動在編譯完程式之後，將所有程式中的編譯時期錯誤整理到 Error List 的視窗之中：



只需要在錯誤上面點兩下，Visual Studio 開發工具就會開啟程式中發生錯誤的位置，你就可以直接針對程式中的錯誤進行修正了。

執行時期錯誤

Visual Studio 在執行時期發生錯誤時，執行階段其實就會 ASP.NET 執行環境則會動態產生錯誤網頁，錯誤網頁內通常會有錯誤的原因與錯誤程式碼所在位址的說明，錯誤網頁的檢視方式將在第十一章介紹。

另一種方式則是中斷模式；中斷模式是利用 Visual Studio 來監控正在執行中的程式碼，搭配中斷點來指定欲停止的程式碼位置，則可在開啟除錯功能後，當程式執行到中斷點指定的位置，即可進入中斷模式來，再利用開發工具的各類除錯視窗獲取相關資訊，然後進行除錯的工作。進入中斷模式所做的設定如下：

1. 在程式碼編輯視窗，決定欲停止的程式碼位置，建立中斷點。
2. 指定除錯的起始網頁。在「Solution Explorer」視窗找到想要除錯的網頁上，按滑鼠右鍵點選並選擇「Set As Start Page」即可。(此步驟是選擇性的，預設會除錯目前開啟的網頁)
3. 接著啓動除錯功能，在工具列點選「Start」按鈕進入除錯模式。

上述三項設定完成之後，只需要等待程式碼執行至中斷點行，即可進入中斷模式。

邏輯錯誤

邏輯錯誤別無他法，僅能使用中斷模式來除錯。在使用中斷模式進行除錯時，也可以利用開發工具所提供的各類視窗來取得程式執行狀況或是變數的相關資訊。在 Visual Studio 開發工具中，常用到的除錯視窗包括：

- Output Window :

用來輸出應用程式執行時的狀態，或是使用 `Debug.WriteLine` 指令輸出的資料。

- Locals Windows :

用來顯示目前中斷點所在的程式位置中，所有區域變數的資料。

- Watch Window :

顯示目前應用程式所監看中的變數或是物件的狀態，使用前須先加入要監看的變數或物件。

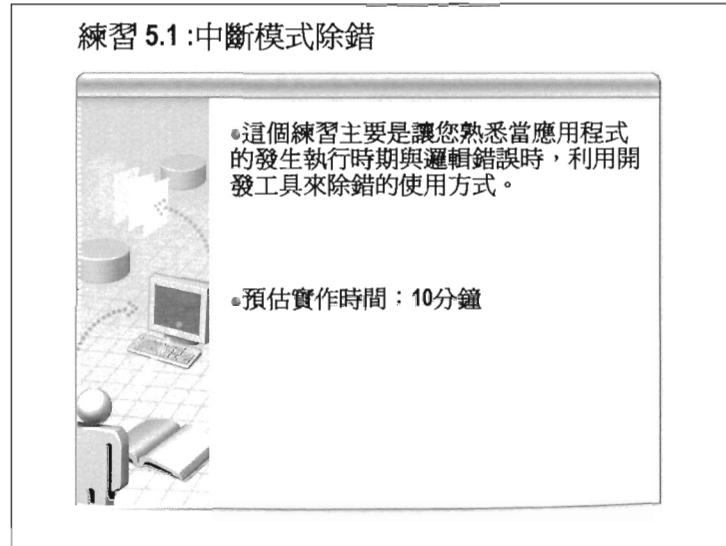
- Call Stack Window :

顯示目前記憶體堆疊配置狀態。

- BreakPoints Window :

可以管理網站所有程式的中斷點設定。

在除錯過程中，如果能善用這些工具視窗，將可以得到事半功倍的效果。



練習 5.1：設定中斷點

目的：

這個練習主要是讓您熟悉當應用程式的發生執行時期與邏輯錯誤時，利用開發工具來除錯的使用方式。

功能描述：

這個練習會開啓一個 ASP.NET Web Form，在加入控制項與程式碼之後，設定中斷點檢查使用者輸入的資料。

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」→「Open」→「Web Site」→選取「File System」按鈕並選取「U9543\Practices\VB 或 CS\Mod05_1\Starter\Mod05_1」目錄，完成後，按下「Open」按鈕即可開啓網站。
3. 從「Solution Explorer」視窗中，開啓專案的 Hello.aspx 表單。
4. 在設計視窗中，用滑鼠點選 Button 控制項兩次，然後在程式碼視窗中，輸入以下程式碼：

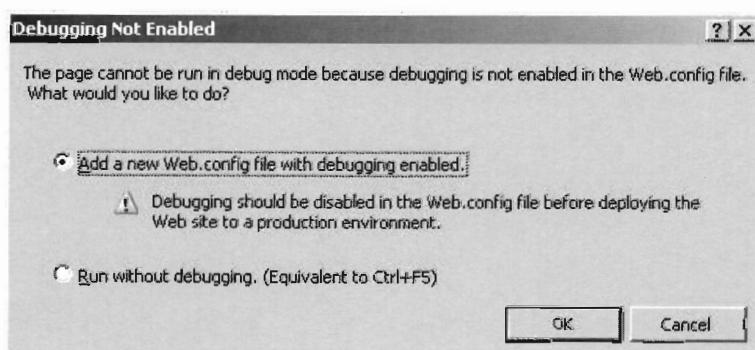
Visual Basic

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim data As String = TextBox1.Text
    Label1.Text = data
End Sub
```

C#

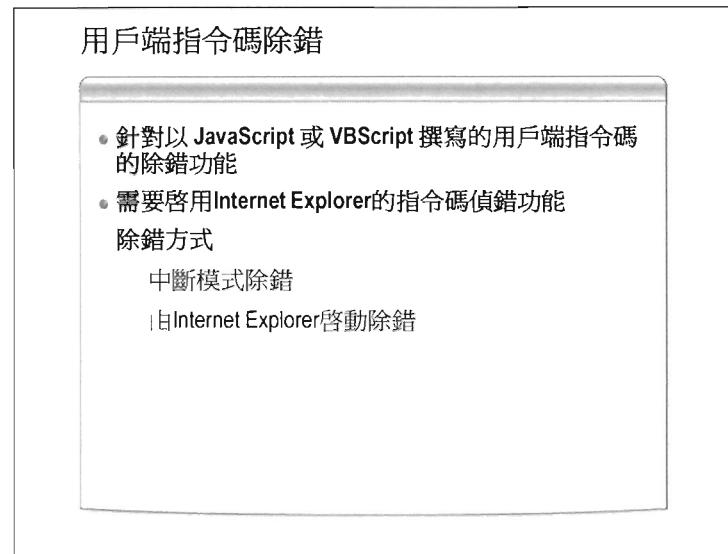
```
protected void Button1_Click(object sender, EventArgs e)
{
    String data = TextBox1.Text;
    Label1.Text = data;
}
```

4. 在剛剛加入的方法的程式碼的左邊區域，按一下滑鼠左鍵，會出現中斷點標示的紅色記號。
5. 在「Solution Explorer」視窗中，展開專案，在 Hello.aspx，按下滑鼠右鍵，選擇「Set As Start Page」。
6. 從工具列或是「Debug」選單中，選擇 按鈕或是「Start Debugging」，開始執行程式。
7. 若是專案中未加入 Web.Config 組態檔，則會出現下面視窗：



選擇「Add a new Web.Config file with debugging enabled.」後。按下「OK」按鈕。

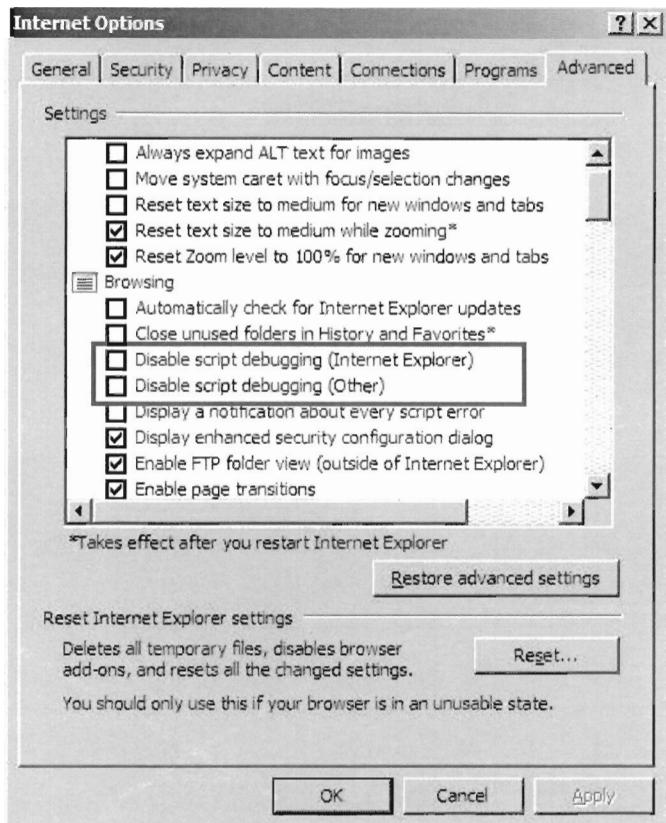
8. 在開啟的 Hello.aspx 表單中，輸入你的名字之後，按下按鈕送出表單。此時 Visual Studio 會停止在中斷點位置，進入除錯模式。此時在螢幕下方會出現一些除錯視窗。
9. 將滑鼠移至 data 變數位置，就可以檢視目前變數的值。
10. 從除錯工具列中，按下  按鈕，移動至下一行後，檢視 data 變數狀態。
11. 按下  按鈕，執行剩下的程式。



用戶端指令碼除錯

用戶端指令碼與伺服器應用程式碼是不同的，用戶端指令碼所執行的位置是在用戶端的瀏覽器上，並非伺服器。Visual Studio 也提供了用戶端指令碼的偵錯，可以讓您在瀏覽器執行用戶端程式碼時利用中斷模式來除錯。

在使用用戶端指令碼除錯時，由於用戶端指令碼是執行瀏覽器上的，所以瀏覽器上也必須要進行相對的設定，其實只要將 Internet Explorer 的工具選項內的「Advanced」頁籤內的「Disable script debugging」選項取消即可。如下所示：



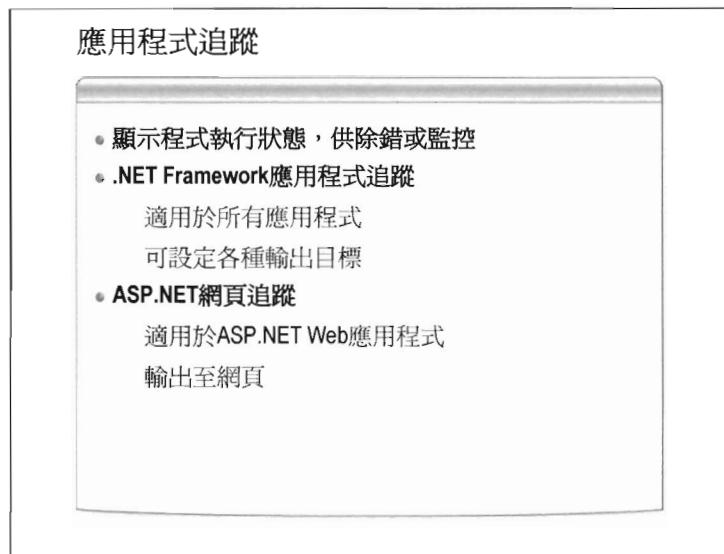
啓動瀏覽器支援用戶端指令碼除錯設定之後，可以使用以下兩種方式除錯：

由中斷模式除錯

1. 在用戶端程式碼編輯視窗，決定欲停止的程式碼位置，建立中斷點。
2. 指定除錯的起始網頁。在「Solution Explorer」視窗找到想要除錯的網頁上，按滑鼠右鍵點選並選擇「Set As Start Page」即可。(此步驟是選擇性的，預設會除錯目前開啟的網頁)
3. 接著啓動除錯功能，在工具列點選「Start」按鈕進入除錯模式。

由 Internet Explorer 啓動除錯

該模式必須要先使用 Internet Explorer 來開啓網頁內容，然後在開啓後，選擇 Internet Explorer 選單中的「View」→「Script Debugger」中的「Open」或是「Break at Next Statement」來啓動除錯即可。



應用程式追蹤

使用除錯工具設定中斷點除錯是一種相當方便的除錯方式，但不一定是最適用的。像是如果要使用加壓工具，模擬程式在實際上線環境中的狀態，並且進行除錯的話，除錯工具就不太適用了。

除了直接針對程式除錯之外，還可以利用應用程式追蹤的方式，來記錄下應用程式在執行過程中的所有狀態；並且在事後針對紀錄文件中的任何異常來調整應用程式的修正。

應用程式追蹤的方式分為.NET Framework 應用程式追蹤與 ASP.NET 網頁追蹤兩種。

.NET Framework 應用程式追蹤

.NET Framework 應用程式追蹤可以適用於各種.NET 所開發出來的應用程式，當然 ASP.NET 亦可使用。

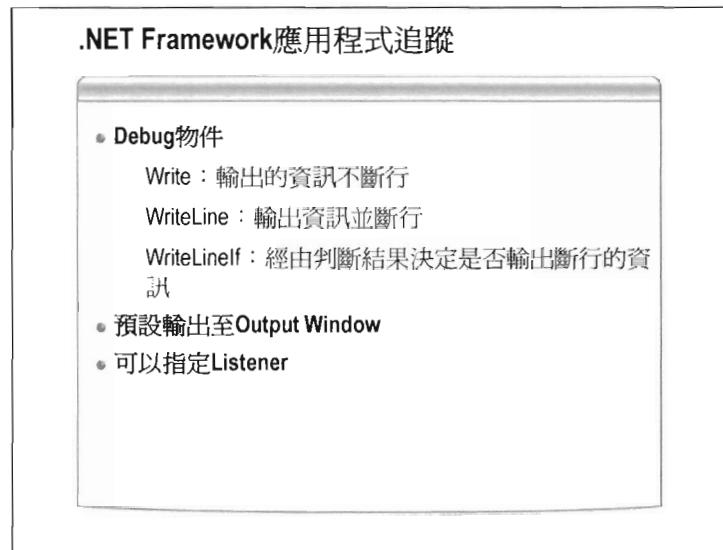
.NET Framework 應用程式追蹤追蹤除了可適用於各種應用程式之外，還具有相當大的設計彈性，可將其輸出的對象設定為.NET

Framework 內建好的目標，如 output 視窗、文字文件或 EventLog 內，也可以支援自訂輸出到任何的目標。

ASP.NET 網頁追蹤

相對於.NET Framework 應用程式追蹤，ASP.NET 網頁追蹤在架構上就單純許多，ASP.NET 網頁追蹤僅能適用於 ASP.NET Web 應用程式。

ASP.NET 網頁追蹤對於輸出目標以及記錄的彈性也不大，預設只會輸出到網頁上，但是由於只針對 ASP.NET Web 應用程式來做追蹤，所以在追蹤資訊上，ASP.NET 網頁追蹤也會主動的取得 ASP.NET Web 應用程式在執行上的大量相關資訊而不需要使用者自行撰寫程式取得。



.NET Framework 應用程式追蹤

在 System.Diagnostics 的命名空間中，提供了兩個物件：Debug 和 Trace，供程式可以將執行的狀態輸出到外界媒體中做紀錄。Debug 物件主要是使用在 Debug 模式中，而 Trace 物件主要則是用在上線環境中執行狀態的追蹤。

Debug 物件提供了以下指令可供操作：

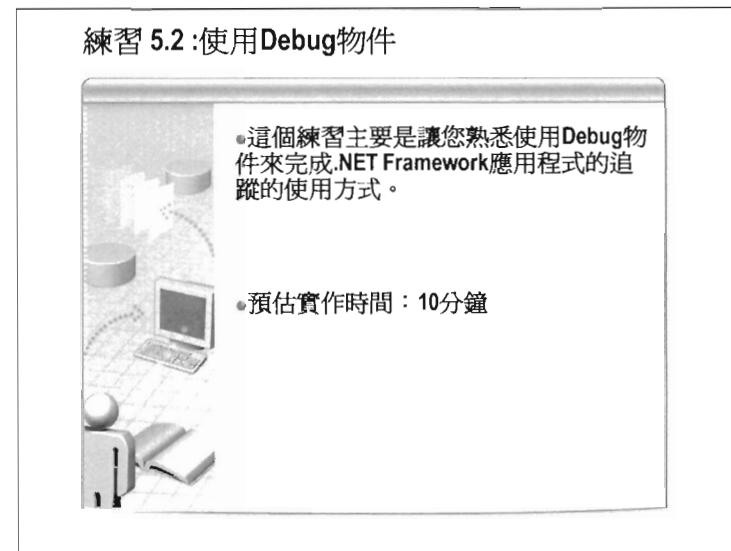
- Write：
 輸出的資訊不斷行
- WriteLine：
 輸出資訊並斷行
- WriteLineIf：
 經由判斷結果決定是否輸出斷行的資訊

Write 和 WriteLine 兩個指令，讓你可以將程式中執行狀態輸出到指定的外界媒體中。例如，你可以使用下面指令將 TextBox1 控制項中的文字輸出：

```
Visual Basic  
Debug.WriteLine (TextBox1.Text)
```

```
C#  
Debug.WriteLine(Textbox1.Text) :
```

Debug 物件預設輸出的位置是除錯工具的 Output Window。如果希望輸出至其他目標，則必須要指定 Listener。



練習 5.2：使用 Debug 物件

目的：

這個練習主要是讓您熟悉使用 Debug 物件來完成.NET Framework 應用程式的追蹤的使用方式。

功能描述：

透過 Debug 物件，將程式的執行狀態寫入指定的 Log 檔案中。

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」→「Open」→「Web Site」→選取「File System」按鈕並選取「U9543\Practices\VB 或 CS\Mod05_2\Starter\Mod05_2」目錄，完成後，按下「Open」按鈕即可開啓網站。

3. 從「Solution Explorer」視窗開啓 Hello.aspx 網頁。

4. 在 Button1_Click 的事件處理常式中，加入下面程式：

Visual Basic

```
Dim data As String = TextBox1.Text
System.Diagnostics.Debug.WriteLine("Data = " & data)
```

C#

```
String data = TextBox1.Text;
System.Diagnostics.Debug.WriteLine("Data = " + data);
```

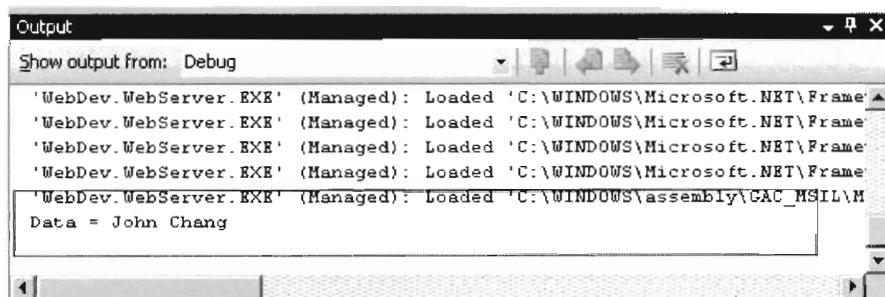
5. 在「Solution Explorer」視窗中，展開專案，在 Hello.aspx，
按下滑鼠右鍵，選擇「Set As Start Page」。

6. 在工具列的「Solution Configuration」下拉選單中，選擇
「Debug」。

7. 從工具列或是「Debug」選單中，選擇 按鈕或是「Start Debugging」(也可以按下 F5 鍵)，開始執行程式。

8. 在開啓的 Hello.aspx 表單中，輸入你的名字之後，按下按鈕
出表單。

9. 在 Visual Studio 開發工具中，選取「View」→「Output」開
啓輸出視窗，會看到 Debug 物件所輸出的除錯資訊：



ASP.NET 網頁追蹤

- Trace 物件
 - Write : 輸出資訊(黑色)
 - Warn : 輸出資訊(紅色)
- 網頁層級
 - 輸出至網頁內容下方
- 網站層級
 - 輸出至網頁內容下方或網站的 Cache 中
- 自訂輸出
 - 需自行訂閱事件

ASP.NET 網頁追蹤

ASP.NET 網頁追蹤僅針對 ASP.NET Web 應用程式來使用，預設會將資訊輸出至網頁上。ASP.NET 網頁追蹤除了會自動記錄網頁傳輸問的相關資訊之外，也提供了自訂輸出；自訂輸出的方式是使用 Trace 物件，Trace 物件提供了 Write 和 Warn 兩個輸出訊息的指令，都可以將資料輸出到報告中，但透過 Warn 輸出的文字將來在報告中會以紅色的文字顯示：

```
Visual Basic
Trace.Write("Test message","Trace.Write")
Trace.Warn("Test message","Trace.Warn")
```

```
C#
Trace.Write("Test message","Trace.Write");
Trace.Warn("Test message","Trace.Warn");
```

而 ASP.NET 網頁追蹤的設計與輸出目標的方式分為以下三個層級：

- 網頁層級

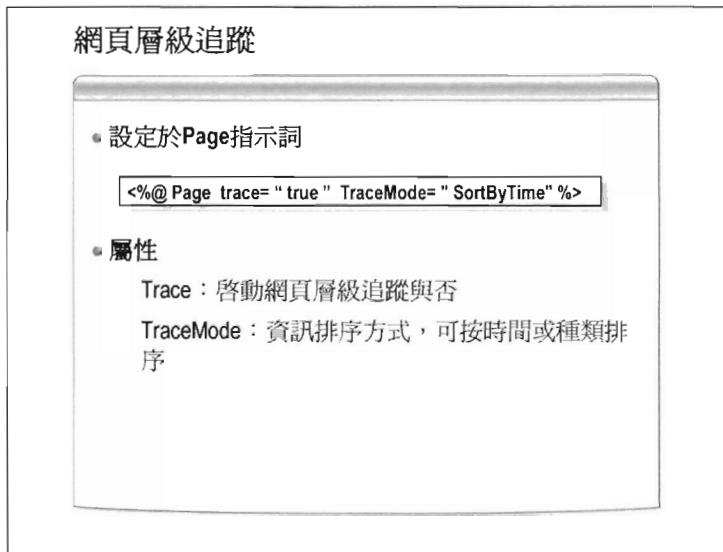
此層級需針對網頁的 Page 指示詞來設定，該層級預設會將資訊輸出至網頁內容下方。

- 網站層級

此層級需對 Web.Config 來設定，輸出目標可以是網頁內容下方或是網站的 Cache 等。

- 自訂輸出

此方法必需自行訂閱 Trace 物件的事件，並在事件發生當下自行撰寫輸出的程式碼。



網頁層級追蹤

如果要追蹤單一頁面執行結果的話，可以啓動頁面層級的追蹤功能。設定的方式很簡單，只需要設定 Page 指示詞中的 trace 屬性值為 true 就可以了：

```
<%@ Page Trace="True">
```

設定完成之後，重新利用瀏覽器開啓網頁，就會看到追蹤的結果，網頁層級的追蹤會直接將網頁的執行狀態自動顯示在輸出的結果中，分類如下：

- Request Details：顯示目前 HTTP 要求與回應的一般資訊。
- Trace Information：顯示頁面層級的事件發生順序。
- Control Tree：顯示此頁中所建立的伺服器控制項的資訊。
- Session State：顯示儲存在 Session 物件中的值。
- Application State：顯示儲存在 Application 物件中的值。
- Request Cookies Collection：透過 HTTP Request 傳送的 Cookie。
- Response Cookies Collection：伺服器端回應的 Cookie 資料。

- Headers Collection : HTTP Header 資料
- Response Headers Collection : 伺服器端回應的 Header 資料
- Form Collection : 使用者端上傳的參數資料。
- QueryString Collection : 使用者端上傳的 QueryString 資料。
- Server Variables : 伺服器端的 Server Variables 資料。

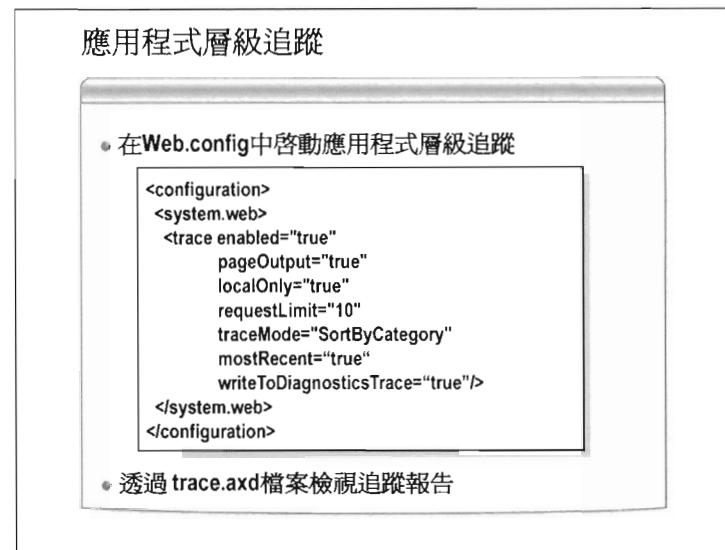
上述的資料也都可以利用程式的方式來取得，例如「Server Variables」內的資訊，可以利用 Request.ServerVariables 來取得。

在設定 Page 指示詞的追蹤功能時，可以順便利用 TraceMode 屬性設定追蹤訊息排序的方式：

```
<%@ Page Trace="True" TraceMode="SortByCategory" %>
```

ASP.NET 可以設定的 TraceMode 有兩種：

1. SortByTime : 按照時間排列追蹤的訊息。
2. SortByCategory : 按照類別排列追蹤的訊息。



應用程式層級追蹤

當多支網頁程式需要追蹤報告時，可以在 Web.Config 檔案中設定啓動應用程式追蹤，如此就不需在每張網頁都設定，設定的內容必須要值撰寫在 Web.Config 檔案中「configuration」→「system.web」→「trace」內，可設定的屬性如下：

- Enabled

用來啓動或取消 Trace 設定區段，預設為 false。

- pageOutput

屬性用來設定是否在網頁中出現追蹤報告，預設為 false。

- localOnly

屬性用來設定追蹤報告只能在本機瀏覽，預設為 true。因為如果網站正在線上使用，追蹤報告當然不能也讓使用者看到。

- requestLimit

屬性用來設定追蹤報告儲存數量，預設為 10。

- traceMode

追蹤訊息排序的方式。

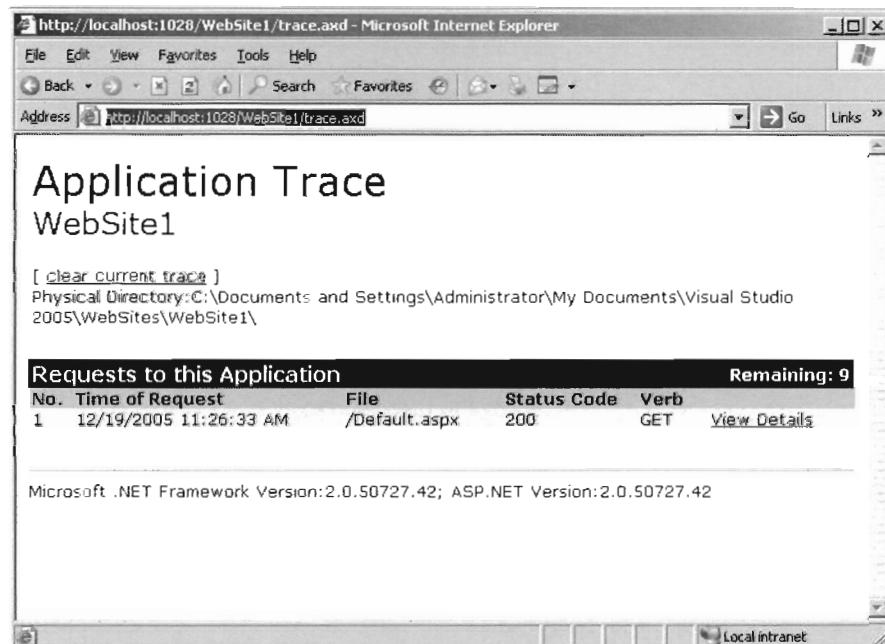
- mostRecent

所蒐集的追蹤報告是否以最新的版本為主。

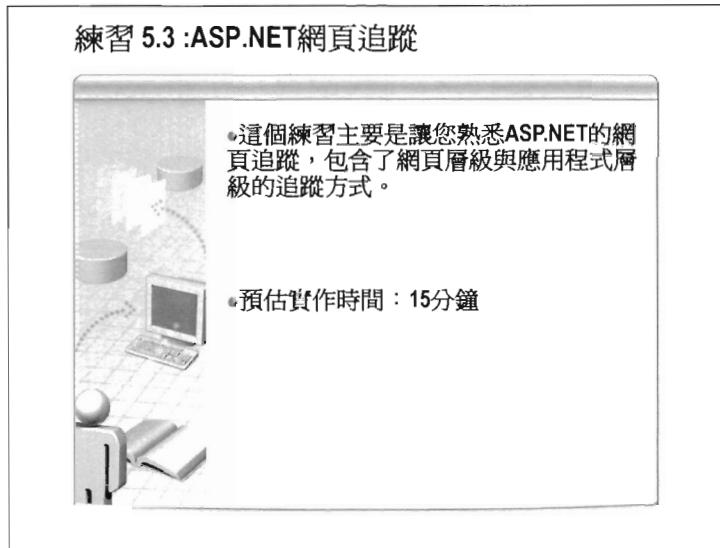
- `writeToDiagnosticsTrace`

追蹤報告是否要輸出一份至 `System.Diagnostics.Trace` 物件。

需注意的是 `pageOutput` 屬性如果設定為 `false`，那麼追蹤報告將不會出現在網頁下方，而是暫存在執行環境的記憶體之中。此時只需在網址列將瀏覽的檔案名稱改為「`Trace.axd`」即可進入 ASP.NET 準備好的追蹤報告管理介面，如下圖：



點選「`View Details`」超連結即可瀏覽每一份追蹤報告，
「`Remaining`」代表所剩儲存空間，圖中左上角「`clear current trace`」
超連結可以清除儲存空間。



練習 5.3：執行頁面層級追蹤

目的：

這個練習主要是讓您熟悉 ASP.NET 的網頁追蹤，包含了網頁層級與應用程式層級的追蹤方式。

功能描述：

本練習將會建立藉由頁面層級追蹤機制，追蹤使用者輸入資料。

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」→「Open」→「Web Site」→選取「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod05_3\Starter\」

Mod05_3」目錄，完成後，按下「Open」按鈕即可開啓網站。

3. 從「Solution Explorer」視窗開啓「Hello.aspx」網頁。
4. 將「Hello.aspx」切換到 Source 檢視模式，在表單最前面的 @Page 指示詞中，設定 Trace 屬性為 True，啓動頁面層級追蹤：

```
<%@ Page Trace="True" %>
```

5. 在 Button1_Click 的事件處理常式中，加入下面程式：

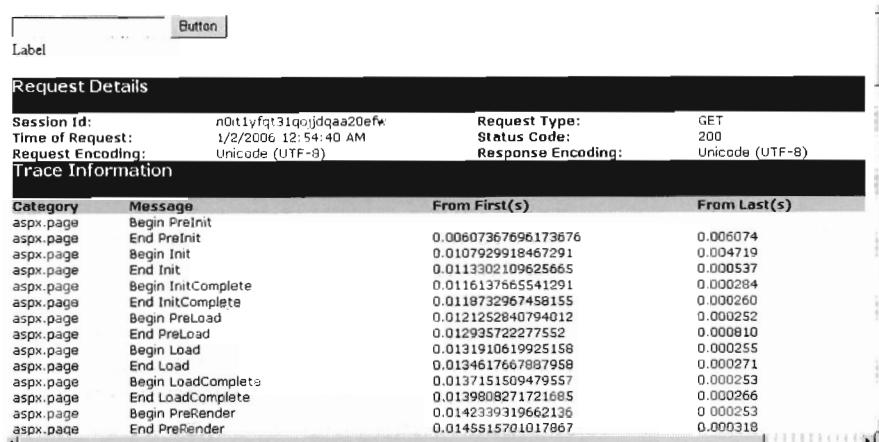
Visual Basic

```
Dim data As String = TextBox1.Text
If (data.Length = 0) Then
    Trace.Warn("My trace data" , "No data input!!")
Else
    Trace.Write("My trace data" , "Data = " & data)
End If
```

C#

```
String data = TextBox1.Text;
if (data.Length == 0)
{
    Trace.Warn("My trace data" , "No data input!!");
}
else {
    Trace.Write("My trace data" , "Data = " + data);
}
```

6. 在「Solution Explorer」視窗中，用滑鼠點選「Hello.aspx」網頁，按下滑鼠右鍵，選擇「View In Browser」之後，會看到如下結果：



7. 直接按下按鈕送出，在「Trace Information」中會顯示：

aspx.page	End Raise ChangedEvents	0.00577475628885794	0.000252
aspx.page	Begin Raise PostBackEvent	0.00635862937887357	0.000594
My trace data	No data input!	0.0087332328550137	0.002375
aspx.page	End Raise PostBackEvent	0.00901259796905371	0.000279

8. 輸入名字後再送出，在「Trace Information」中會顯示：

aspx.page	End Raise ChangedEvents	0.0154701225993806	0.000269
aspx.page	Begin Raise PostBackEvent	0.015729652791067	0.000260
My trace data	Data = John Chang	0.0164783512988383	0.000749
aspx.page	End Raise PostBackEvent	0.0169996466031297	0.000521

9. 接著從「Solution Explorer」視窗開啓「Hello.aspx」網頁，將
@Page 指示詞的 Trace 屬性刪除。

10. 開啓 Web.Config 設定檔，加入<Trace>區塊設定：

```
<system.web>
  <trace enabled="true"
    pageOutput="false"
    localOnly="true"
    requestLimit="10"
    traceMode="SortByCategory" />
</system.web>
```

11. 在「Solution Explorer」視窗中，用滑鼠點選「Hello.aspx」網
頁，按下滑鼠右鍵，選擇「View In Browser」之後，執行表
單內容數次。

12. 在網址列輸入[http://localhost:\[port 編號\]/Mod05_3/Trace.axd](http://localhost:[port 編號]/Mod05_3/Trace.axd)，
檢視應用程式追蹤報表。並點選「View Details」，檢視報表
內容。



自訂輸出

自訂輸出可以讓使用者自行決定輸出的目標，但是該方式就必須要自行來撰寫輸出的程式碼，設計自訂輸出的方式如下：

1. 訂閱 Trace 物件的 TraceFinished 事件，並建立 TraceFinished 的事件處理常式
2. 在 TraceFinished 的事件處理常式內取得 TraceContextEventArgs 型別的參數，再將該參數內的 TraceRecords 集合內的成員一一取出。
3. 取出 TraceRecords 集合內的成員後，自行撰寫輸出至自訂目標的程式碼即可。

設計時需注意的是，Trace 物件的 TraceFinished 事件並不會主動觸發，必須要將網頁或是網站的追蹤功能開啓。較好的設定方式為，開啓網站層級的追蹤功能，將 pageOutput 設定為 false，mostRecent 則設定為 true，如此才能開啓追蹤但不顯示於網頁內容，並以最新的版本為主。

總結

- 應用程式中常見的錯誤種類辨識
- 了解Visual Studio支援的除錯種類
- 介紹伺服器端、用戶端指令除錯
- 應用程式追蹤方式與設計
- 總結

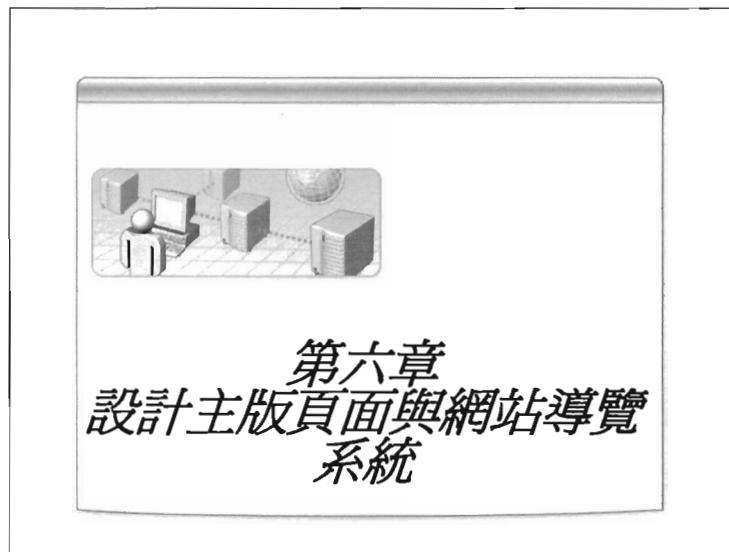
第六章: 設計主版頁面與網站導覽系統

本章大綱

課程大綱.....	3
什麼是 Master Page.....	4
Master Page 架構簡介.....	6
設計 Master Page	8
練習 6.2 : 設計 Master Page	10
Master Page 與 Content Page 之間的溝通.....	14
練習 6.2 : 在 Content Page 中控制 Master Page 的內容... ..	18
動態變更對應的 Master Page	21
什麼是網站導覽控制項？	22
建立 Web.sitemap 檔案.....	24
練習 6.3 : 建立 Web.sitemap 檔案.....	26
使用網站導覽控制項.....	28
Menu 雖服器控制項	29
TreeView 雖服器控制項.....	31
SiteMapPath 雖服器控制項.....	33
練習 6.4 : 使用網站導覽控制項	35

作者：
周季賢





大綱

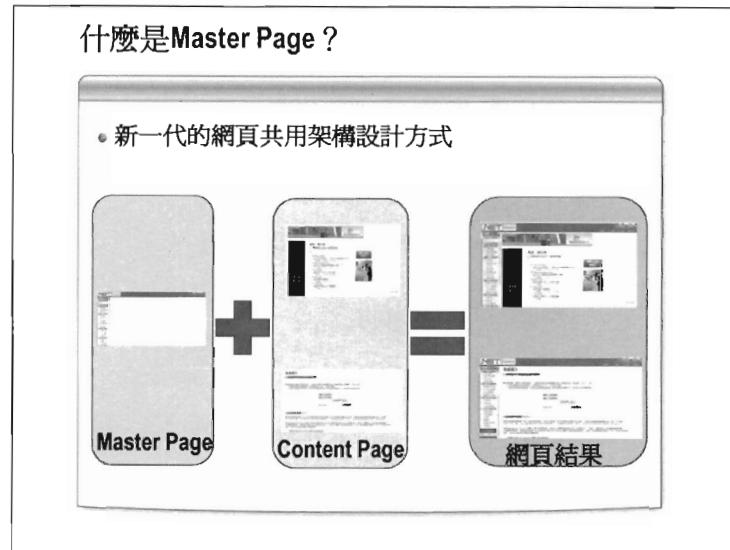
- 什麼是 Master Page?
- Master Page 架構簡介
- 設計 Master Page
- Master Page 與 Content Page 之間的溝通
- 動態變更對應的 Master Page
- 什麼是網站導覽控制項？
- Web.sitemap 檔案
- 使用網站導覽控制項
- 總結

課程大綱

在這個章節中將介紹 ASP.NET Web 應用程式的網站排版規劃方式，除了介紹主版頁面的設計方式之外，還會介紹可搭配主版頁面使用者網站導覽控制項的使用方式。

本章介紹以下主題：

- 什麼是 Master Page?
- Master Page 架構簡介
- 設計 Master Page
- Master Page 與 Content Page 之間的溝通
- 動態變更對應的 Master Page
- 什麼是網站導覽控制項？
- Web.sitemap 檔案
- 使用網站導覽控制項
- 總結

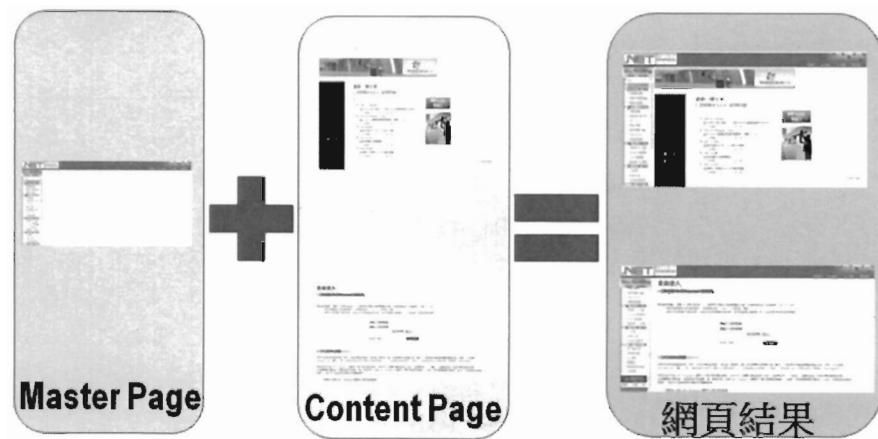


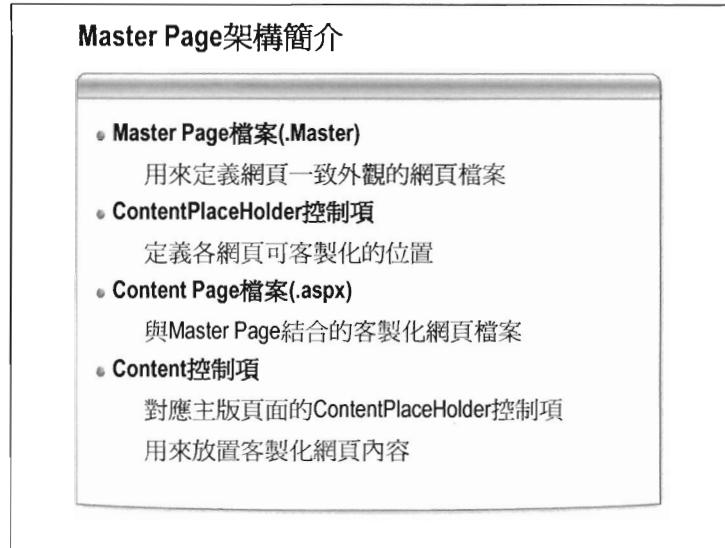
什麼是 Master Page

主版頁面(Master Page : *.master)設計用來讓網頁(*.aspx)便利地共用結構。以往大多使用框架頁(Frameset)或者是表格(Table)來規劃網站主結構，主版頁面成為新的一種設計方式。

當使用主版頁面及其相關的內容頁面時，只需要將必要與共用的HTML文件標記(例如html、head和body)及結構設計加入至主版頁面，內容網頁則定義內容以放入主版頁面上的預留位置。

當網頁執行時，主版頁面和內容頁面會組合成與內容頁面同名的單一類別。產生的已編譯、合併類別是繼承自 Page 類別。





Master Page 架構簡介

除了了解 Master Page 運作方式之外，以下還分別介紹了四個在建置 Master Page 時的重要檔案與控制項：

- **Master Page 檔案(.Master)**

此檔案建立後的設計方式與一般網頁(.aspx)並無太大差異，不同的僅是指示詞的關鍵字為 Master，此檔案用來設計網站內所有網頁的一致外觀。

- **ContentPlaceHolder 控制項**

在 Master Page 內使用，預設在建立 Master Page 之後就會加入，亦可從 Toolbox 中加入；ContentPlaceHolder 控制項是用来在 Master Page 中定義各網頁可客製化的位置，只要 Master Page 的任意位置中放入該控制項，屆時要與 Master Page 結合的網頁就可以在該部分置入要客製化的外觀；反之，如沒有加入該控制項的位置即無法讓各網頁客製化。

- **Content Page 檔案(.aspx)**

與 Master Page 結合的客製化網頁檔案，具有一般 Web Form 的副檔名 aspx；建立方式也與 Web Form 幾乎相同，僅在建立

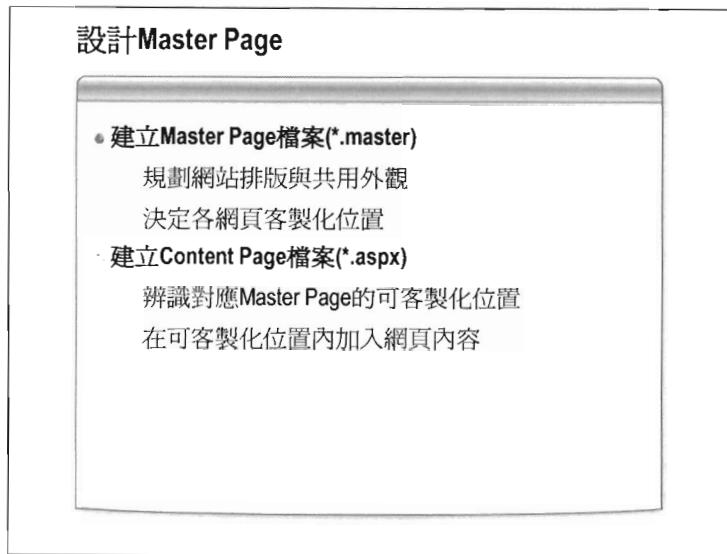
Web Form 時的對話視窗必須要另外勾選「Select Master Page」選項即可。

建立後的檔案內容與一般 Web Form 不同，只會有 Page 指示詞與數量和 Master Page 內 ContentPlaceHolder 控制項一樣多的 Content 控制項。一般的 HTML 文件標記(例如 html、head 和 body)並不會出現，也不需要，因為已經定義在 Master Page 中。而 Page 指示詞內會有一 MasterPageFile 屬性來對應至 Master Page 的檔案位置。

- Content 控制項

在 Content Page 中使用，預設在建立 Content Page 之後就會加入，亦可從 Toolbox 中加入；Content 控制項是用來對應 Master Page 的 ContentPlaceHolder 控制項，並且讓程式設計師在該控制項內放置客製化網頁內容；

Content 控制項內會有一個名為 ContentPlaceholderID 的屬性，用來對應至 Master Page 的 ContentPlaceHolder 控制項 ID，而放在 Content 控制項內的內容，就會在執行時期置入所對應 ID 的 ContentPlaceHolder 控制項在 Master Page 的位置。



設計 Master Page

建立以主版頁面設計為主的網站是必須有設計順序的，以下分別說明建立方式與順序。

建立 Master Page 檔案 (*.master)

自選單「Web Site」下「Add New Item...」，選取「Master Page」即可建立 Master Page 檔案。

建立後，便可規劃網站排版與共用的外觀，排版的部分，通常利用 Html 表格或是 CSS 來控制，而共用外觀所指的不外乎網站標題、網站導覽控制項或是會員功能等。

而完成排版與共用的外觀的規畫之後，接著就是利用 ContentPlaceholder 控制項來決定網頁可客製化的位置，設置好 ContentPlaceholder 控制項之後，Master Page 檔案的建立就完成了。

建立 Content Page 檔案(*.aspx)

自主選單「Web Site」下「Add New Item...」，選取「Web Form」並且勾選「Select Master Page」選項，選擇對應的 Master Page 檔案之後，即可建立 Content Page 檔案。

建立完成後的 Content Page 檔案內容僅會有 Page 指示詞與數量與 Master Page 內 ContentPlaceHolder 控制項一樣多的 Content 控制項；此時必須要先辨識出每一個 Content 控制項是對應到 Master Page 內的哪一個 ContentPlaceHolder 控制項，以利後續的客製化，亦或直接切換至 Content Page 檔案的 Design 頁面，也可辨識出 Content 控制項所對應到 Master Page 內可客製化的位置。

辨識出可客製化位置之後，接著就是在 Content 控制項內置入客製化內容，如此就可以完成 Content Page 的設計了。

練習 6.1 : 設計 Master Page



練習 6.2 : 設計 Master Page

目的：

這個練習主要是讓您熟悉 Master Page 的設計方式，包含了 Master Page 與 Content Page 的建立與設定。

功能描述：

在這個練習中，將會建立一個網站共用的 Master Page 檔案，利用表格排版的方式完成共用內容的建立，並另外建立三個 Content Page 檔案來對應至 Master Page 檔案。

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「Empty Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9543\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「Master Page」，檔名命名為「myMasterPage.master」，清除「Place code in seperate file」核取方塊，建立 Master Page 檔案。
4. 在「myMasterPage. master」檔案內，加入表格標籤，加入之後再將名為「ContentPlaceHolder1」的 ContentPlaceHolder 控制項移動至標籤內，完成後檔案標籤應如下所示：

```
<%@ Master Language="VB" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

</script>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder id="head" runat="server">
    </asp:ContentPlaceHolder>

</head>
<body>
    <form id="form1" runat="server">
        <div>
            <table style=" width: 99%;height: 468px; border-style: solid; border-width: 2px;">
                <tr>
                    <td colspan="2" style="border-style: solid; width: 100%; height: 20%">
                        <h1>UCOM資訊網站</h1>
                    </td>
                </tr>
                <tr>
                    <td style="border-style: solid; width: 20%; height: 80%">
                        &nbsp;
                    </td>
                    <td style="border-style: solid; width: 80%; height: 80%" valign="top">
                        <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

                        </asp:ContentPlaceHolder>
                    </td>
                </tr>
            </table>
        </div>
    </form>
</body>
</html>
```

```

<br />
</div>
</form>
</body>
</html>

```

5. 接著切換至 Design 頁面，外觀應如下圖：



6. 完成之後，接著就是加入 Content Page 網頁，自選單「Web Site」下「Add New Item...」，選取「Web Form」，檔名命名為「Content1.aspx」，清除「Place code in separate file」核取方塊，勾選「Select Master Page」核取方塊，在「Select a master page」對話視窗內選取「myMasterPage.master」建立 Content Page 網頁。
7. 完成之後，在「Content1.aspx」內名為「Content2」的 Content 控制項標籤內，加入 Label 控制項，並將其 Text 屬性設定為「這是 Content1.aspx 的網頁內容」。完成後網頁標籤應如下所示：

```

<%@ Page Title="" Language="VB" MasterPageFile="~/myMasterPage.master" %>

<script runat="server">

</script>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
    <asp:Label ID="Label1" runat="server" Text="這是Content1.>

```

```
aspx的網頁內容"></asp:Label>
</asp:Content>
```

8. 依照步驟 6 與 7，依序再建立名為「Content2.aspx」與「Content3.aspx」的 Content Page，並且分別將兩網頁內的 Label 控制項 Text 屬性設定為「這是 Content2.aspx 的網頁內容」與「這是 Content3.aspx 的網頁內容」
9. 完成之後，接著在「Solution Explorer」→ 分別點選「Content1.aspx」、「Content2.aspx」與「Content3.aspx」→ 按滑鼠右鍵 → 選「View In Browser」。結果應如下圖：

UCOM資訊網站

這是Content1.aspx的網頁內容



UCOM資訊網站

這是Content2.aspx的網頁內容

UCOM資訊網站

這是Content3.aspx的網頁內容

Master Page與Content Page之間的溝通

FindControl方法

Visual Basic	Dim h As HyperLink = CType(Master.FindControl("HyperLink1"), HyperLink) h.ImageUrl = "http://www.netmag.com.tw/images/frame/top01.gif"
C#	HyperLink h = (HyperLink)Master.FindControl("HyperLink1"); h.ImageUrl = "http://www.netmag.com.tw/images/frame/top01.gif";
<ul style="list-style-type: none"> • 使用強行別存取Master Page公開屬性 	
<%@ MasterType VirtualPath="~/Mod06/MasterPage.master" %>	

Master Page 與 Content Page 之間的溝通

雖然完成了主版頁面的設計，在使用上還是有些問題存在；最重要的問題之一，就是主版頁面上的資訊溝通問題；雖然 Content Page 在瀏覽時會與 Master Page 合併成為一張網頁，但式設計內容實際上是分別在 Master Page 與 Content Page 兩份檔案內的；也因為這樣，當 Master Page 與 Content Page 內的資訊如果要互相傳遞的話，其實並不是跟平常的網頁一樣，直覺性的利用控制項或是物件就可以操作。以下分別介紹雙方之間的資訊溝通方式。

Master Page 控制 Content Page 內容

Master Page 如果要控制 Content Page 的內容，必須要利用 Master Page 內的 ContentPlaceHolder 控制項，要先找到 Content Page 中要操作的控制項，並且辨識該控制項位於哪個 Content 控制項內，再進一步的依照 Content 控制項中所對應的 ContentPlaceHolder 控制項 ID 來找到 Master Page 中的 ContentPlaceHolder 控制項。

找到 Master Page 中的 ContentPlaceHolder 控制項之後，便可以在 Master Page 中利用程式的方式，至該 ContentPlaceHolder 控制項內使

用 FindControl 的方法，在方法中指定控制項 ID 來取得 Content Page 中的控制項，取回之後便可操作該控制項內容了。程式碼如下所示：

```
Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim h As HyperLink = CType(ContentPlaceHolder1.FindControl("HyperLink1"), HyperLink)
    h.ImageUrl = "http://www.netmag.com.tw/images/frame/top01.gif"
End Sub
```

```
C#
protected void Button1_Click(object sender, EventArgs e)
{
    HyperLink h=(HyperLink) ContentPlaceHolder1.FindControl("HyperLink1");
    h.ImageUrl = "http://www.netmag.com.tw/images/frame/top01.gif";
}
```

Content Page 控制 Master Page 內容

Content Page 控制 Master Page 的方式亦可使用 FindControl 的方法，只不過要改成使用 Content Page 的 Master 屬性即可。程式碼如下所示：

```
Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim h As HyperLink = CType(Master.FindControl("HyperLink1"), HyperLink)
    h.ImageUrl = "http://www.netmag.com.tw/images/frame/top01.gif"
End Sub
```

```
C#
protected void Button1_Click(object sender, EventArgs e)
{
    HyperLink h=(HyperLink)Master.FindControl("HyperLink1");
    h.ImageUrl = "http://www.netmag.com.tw/images/frame/top01.gif";
}
```

另一種方法，是可以使用強型別的方式來操作 Master Page 的外部屬性，此種方式必須要先在 Master Page 中建立外部屬性，如下所示：

```
Visual Basic
Public Property lbText() As String
    Get
        Return label1.Text
    End Get
    Set(ByVal value As String)
        label1.Text = value
    End Set
End Property
```

```
C#
public string lbText
{
    get
    {
        return Label1.Text;
    }
    set
    {
        Label1.Text = value;
    }
}
```

完成之後，還必須要在 Content Page 中加入 MasterType 指示詞，並使用 VirtualPath 屬性來指定 Master Page 的檔案位址，如下所示：

```
<%@ MasterType VirtualPath="~/Mod06/MasterPage.master" %>
```

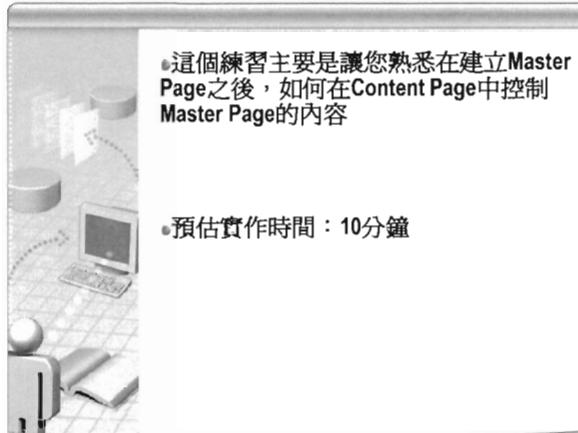
做完指定之後，在 Content Page 內的 Master 屬性就會指定為 Master Page 所對應的型別，也就是說可以直接利用 Master 來取得 Master Page 的外部屬性了。如下所示：

```
Visual Basic
Protected Sub Page_Load (ByVal sender As Object, ByVal e As System.EventArgs)
    Response.Write(Master.lbText)
End Sub
```

```
C#
protected void Page_Load(object sender, EventArgs e)
```

```
{  
    Response.Write(Master.lbText);  
}
```

練習 6.2：在 Content Page 中控制 Master Page 的內容



練習 6.2：在 Content Page 中控制 Master Page 的內容

目的：

這個練習主要是讓您熟悉在建立 Master Page 之後，如何在 Content Page 中控制 Master Page 的內容

功能描述：

這個練習會在網頁中 Master Page 與 Content Page 中分別加入不同的控制項，並且讓 Content Page 在跟使用者互動時，能夠去更動 Master Page 中控制項的值。

預估實作時間：15 分鐘

實作步驟：

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

11. 從「File」→「Open」→「Web Site」→選取「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod06_2\Starter\Mod06_2」目錄，完成後，按下「Open」按鈕即可開啓網站。
12. 在「myMasterPage.master」檔案內，從「Toolbox」工具箱中拖拉一個 Label 控制項至表格標籤的左下方，完成後在 Design 設計頁面應如下所示：



13. 在「Content1.aspx」網頁內，從「Toolbox」工具箱中拖拉一個 Button 控制項至名為「Content2」的 Content 控制項標籤內。
14. 對 Button 控制項連點滑鼠右鍵兩下，進入 Button1_Click 事件處理常式，在程式碼區塊中加入下面的程式碼：

```
Visual Basic
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim myLabel As Label = CType(Master.FindControl("Label1"), Label)
    myLabel.Text = "Hello World"
End Sub
```

```
C#
protected void Button1_Click(object sender, EventArgs e)
{
    Label myLabel = (Label)Master.FindControl("Label1");
    myLabel.Text = "Hello World";
}
```

15. 完成之後，接著在「Solution Explorer」→點選
「Content1.aspx」→按滑鼠右鍵→選「View In Browser」。結果應如下圖：

UCOM資訊網站

這是Content1.aspx的網頁內容

Label

16. 點選按鈕之後，結果應如下圖：

UCOM資訊網站

這是Content1.aspx的網頁內容

Hello World

動態變更對應的Master Page

- 利用Page_PreInit事件處理程式
- 更改Content Page的MasterPage屬性

Visual Basic

```
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As EventArgs)
    Page.MasterPageFile = "MasterPage2.master"
End Sub
```

C#

```
protected void Page_PreInit(object sender, EventArgs e)
{
    Page.MasterPageFile = "MasterPage2.master";
}
```

動態變更對應的 Master Page

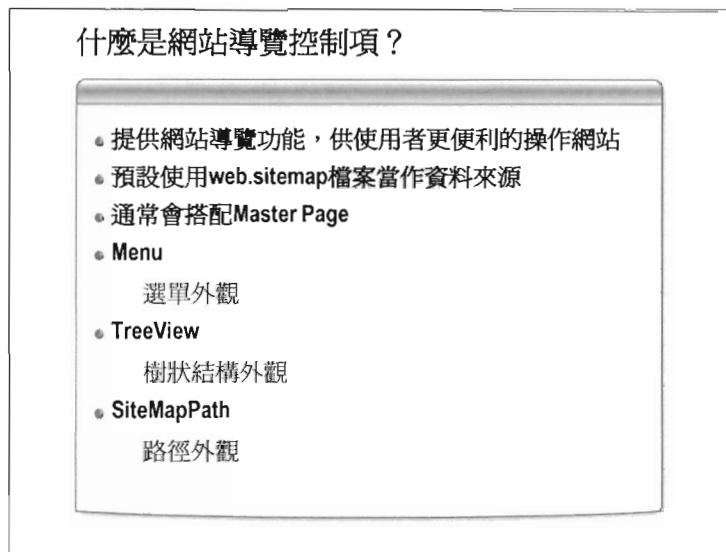
除了在設計時期利用 Page 指示詞的 MasterPageFile 屬性來對指定 Master Page 之外，還可以在執行時期利用程式碼的方式來指定 page 物件的 MasterPageFile 來操作；另外需注意的是，程式執行的時機點僅能針對 Page 的 PreInit 事件來操作。程式碼如下所示：

Visual Basic

```
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As EventArgs)
    Page.MasterPageFile = "MasterPage2.master"
End Sub
```

C#

```
protected void Page_PreInit(object sender, EventArgs e)
{
    Page.MasterPageFile = "MasterPage2.master";
}
```



什麼是網站導覽控制項？

一個對外的網站，特別是提供資訊的新聞網站、技術網站…等，通常具備大量的網頁，為了方便使用者瀏覽網站的內容，這些網站會透過一些專門輔助使用者導覽用的操作界面，這些導覽用的操作界面，無論是超連結或者是圖片，目的都是為了讓瀏覽網站的人，快速找到所需要的資料。

預設使用 web.sitemap 檔案當作資料來源

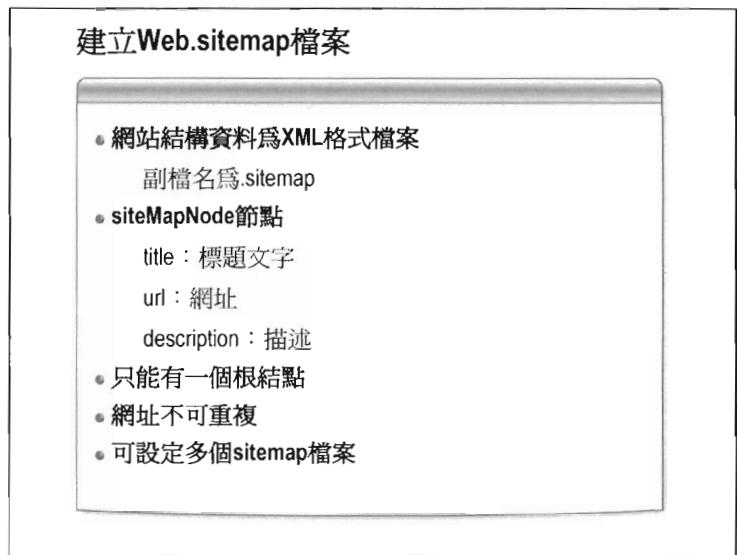
ASP.NET 預設用來設定網站導覽控制項內容的方式，是透過 XML 格式的 web.sitemap 檔案來定義，建立網站導覽控制項的第一步，就是要先建立這些網站導覽控制項所需要的資料來源。

通常會搭配 Master Page

在 ASP.NET 中，讓網頁操作界面或控制項重覆使用的機制稱為 Master Page，透過 Master Page 可以將所有網頁所共用的部分，容易設計和維護，而網站導覽控制項就是符合所有網頁需要的功能，故網站導覽控制項通常會放到 Master Page 內。

ASP.NET 提供導覽功能的控制項就是網站導覽控制項，包含：

- 選單外觀：Menu 伺服器控制項。
- 樹狀結構外觀：TreeView 伺服器控制項。
- 路徑外觀：SiteMapPath 伺服器控制項。



建立 Web.sitemap 檔案

預設檔案為 Web.sitemap，如果要修改檔名，必須透過組態檔指定。網頁導覽控制項會以此檔當作資料來源，並依據 Xml 標籤定義的結構展現結果。

siteMapNode 節點

Web.sitemap 檔案格式有一定的規則，首先根節點為<siteMap>標籤，網頁結構的每一個節點則都使用<siteMapNode>標籤來定義，並且以 url 屬性指定超連結的網頁，title 屬性指定超連結的文字，description 屬性則指定提示字。以下是一個標準的 Web.sitemap 檔案內容：

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-
File-1.0" >
    <siteMapNode url="Default.aspx" title="首頁" description="">
        <siteMapNode url="News.aspx" title="最新消息" description =
        "" />
        <siteMapNode url="Products.aspx" title="產品介紹" descrip-
        tion="" />
        <siteMapNode url="Links.aspx" title="相關資訊" description =
        "" />
    </siteMapNode>
</siteMap>
```

只能有一個根節點

Web.sitemap 檔案的根節點為`<siteMap>`標籤，不過屆時辨識網頁架構時其實並不會讀取該節點標籤當作網站內容，而是讀取`<siteMap>`標籤下的第一層`<siteMapNode>`標籤，而這份`<siteMapNode>`標籤在撰寫時是有限制的，就是只能有一份，而這份`<siteMapNode>`標籤也通常是拿來描述「首頁」用的節點。

網址不能重複

除了只能有一個根節點之外，在撰寫 Web.sitemap 檔案內容時還需要小心的是，`<siteMapNode>`標籤內的 `url` 屬性；該屬性所描述的網址在此份文件中是不能重複的，也就是必須要唯一，這個限制也造成了 SiteMapPath 控制項運作的可能性。

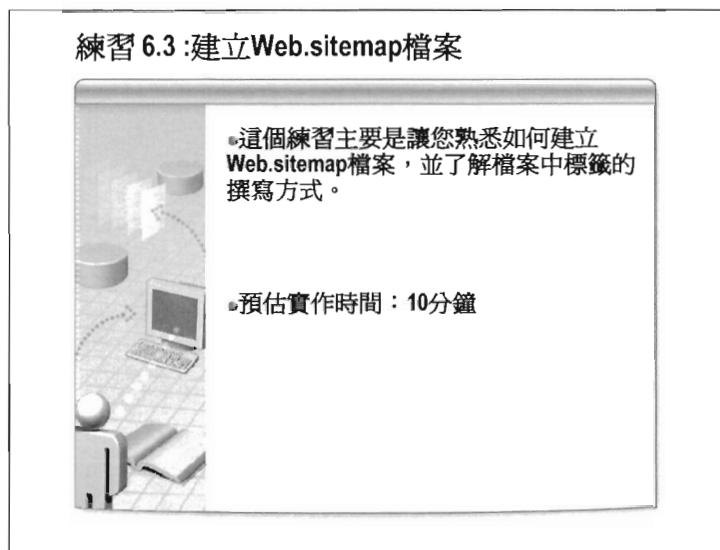
可設定多個 Web.sitemap 檔案

有時候在因為開發上的需求，可能會有一個以上的架構需要描述，而 Web.sitemap 檔案也允許多組架構的描述。

預設 Web.sitemap 檔案是不能更改檔名的，也就是.NET 的工具在使用時是不需要指定檔名的，工具會自動去尋找名為 Web.sitemap 檔名的檔案來使用，所以如果要使用多組架構的話，首先就是在加入 Web.sitemap 檔案時，變更 Web.sitemap 的檔名，接著就是在 Web.Config 檔案中，利用`<siteMap>`標籤內`<providers>`標籤的來定義檔名不為 Web.sitemap 的檔案，如下所示：

```
<providers>
<add
  name="aaProvider"
  type="System.Web.XmlSiteMapProvider"
  siteMapFile = "Web2.sitemap" />
</providers>
</siteMap>
```

定義完成之後，就是將`<providers>`標籤內的 `name` 屬性值提供給工具（如 SiteMapDataSource 或 SiteMapPath）做辨識即可。



練習 6.3：建立 Web.sitemap 檔案

目的：

這個練習主要是讓您熟悉如何建立 Web.sitemap 檔案，並了解檔案中標籤的撰寫方式。

功能描述：

在這個練習中，將建立一個 Web.sitemap 檔案，並且在該檔案中描述網站架構。

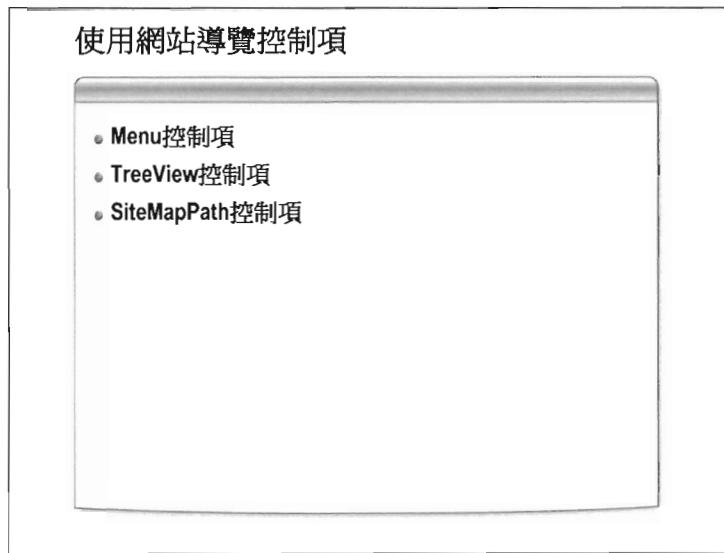
預估實作時間：15 分鐘

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「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod06_3\Starter\Mod06_3\」目錄，完成後，按下「Open」按鈕即可開啟網站。

3. 自主選單「Web Site」下「Add New Item...」，選取「SiteMap」，檔名命名為「Web.sitemap」，建立 Web.sitemap 檔案。
4. 建立好 Web.sitemap 檔案之後，接著在該檔案內建立網站架構描述，完成後應如下所示：

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-
File-1.0" >
    <siteMapNode url="Content1.aspx" title="首頁" description="這是Content1.aspx">
        <siteMapNode url="" title="分類一" description="" >
            <siteMapNode url="Content2.aspx" title="網頁內容1" description="這是Content2.aspx" />
        </siteMapNode>
        <siteMapNode url="" title="分類二" description="" >
            <siteMapNode url="Content3.aspx" title="網頁內容2" description="這是Content3.aspx" />
        </siteMapNode>
    </siteMapNode>
</siteMap>
```



使用網站導覽控制項

本小節將介紹如何在 Web.sitemap 檔案建立完成之後，搭配網站導覽控制項的使用方式，本小節將介紹的控制項如下：

- Menu 控制項
- TreeView 控制項
- SiteMapPath 控制項



Menu 伺服器控制項

Menu 伺服器控制項是常見的展現網站結構的一種方式，幾乎所有的網站都需要這樣的控制項，以往的解決方案如果是自行開發，那就必須學會許多 DHTML 的技巧；而如果找別人寫好的，這就必須先看懂才能做修改的動作。

設定 Web.sitemap 檔案

預設資料來源為網站應用程式根目錄下的 Web.sitemap 檔案，所以必須先編寫好。

建立 SiteMapDataSource 控制項

在使用 Menu 控制項之前，必須要先建立 SiteMapDataSource 控制項，該控制項是用來將 Web.sitemap 檔案內容解析之後，在執行時期將 Web.sitemap 檔案內的資訊提供給 Menu 控制項。該控制項可以在「Toolbox」→「Data」內找到，找到之後還只需要開啓 SiteMapDataSource 控制項的「Data Source Configuration」對話視窗，在該視窗內點選「Site Map」圖示後點選「OK」即可。

加入 Menu 控制項

Menu 控制項可在「ToolBox」→「Navigation」內找到，在網頁上適當位置加入 Menu 控制項即可。另外，使用 Menu 控制項搭配 Master Page 會有不錯的展現效果。

指定 SiteMapDataSource 控制項 ID 即可

加入 Menu 控制項之後，最後只需要在該控制項的 DataSourceID 的屬性設定為先前加入的 SiteMapDataSource 控制項的 ID，即可完成 Menu 控制項的設定。

特殊屬性

- Orientation

由於 Menu 控制項擺放在網頁的上方，適合使用水平方式展現，但預設為垂直展現，所以可以透過修改 Orientation 屬性為 Horizontal。

- StaticDisplayLevels

由於網頁結構的第一層並沒有設定節點內容，而預設 Menu 控制項展現時會因為 StaticDisplayLevels 屬性為 1，所以只會顯示出第一層，只要將這個屬性值改為 2，一開始就會展現前兩層的選單結構內容。



TreeView 雜項

TreeView 雜項也是常見的網站結構展現方式，通常應用在多層次資料的分類，像 MSDN Library 網站就需要這種控制項來展現資料的群組關係。

設定 Web.sitemap 檔案

預設資料來源為網站應用程式根目錄下的 Web.sitemap 檔案，所以必須先編寫好。

建立 SiteMapDataSource 控制項

在使用 TreeView 控制項之前，必須要先建立 SiteMapDataSource 控制項，該控制項是用來將 Web.sitemap 檔案內容解析之後，在執行時期將 Web.sitemap 檔案內的資訊提供給 Menu 控制項。該控制項可以在「Toolbox」→「Data」內找到，找到之後還只需要開啟 SiteMapDataSource 控制項的「Data Source Configuration」對話視窗，在該視窗內點選「Site Map」圖示後點選「OK」即可。

加入 TreeView 控制項

TreeView 控制項可在「ToolBox」→「Navigation」內找到，在網頁上適當位置加入 Menu 控制項即可。另外，使用 TreeView 控制項搭配 Master Page 會有不錯的展現效果。

指定 SiteMapDataSource 控制項 ID 即可

加入 TreeView 控制項之後，最後只需要在該控制項的 DataSourceID 的屬性設定為先前加入的 SiteMapDataSource 控制項的 ID，即可完成 TreeView 控制項設定。

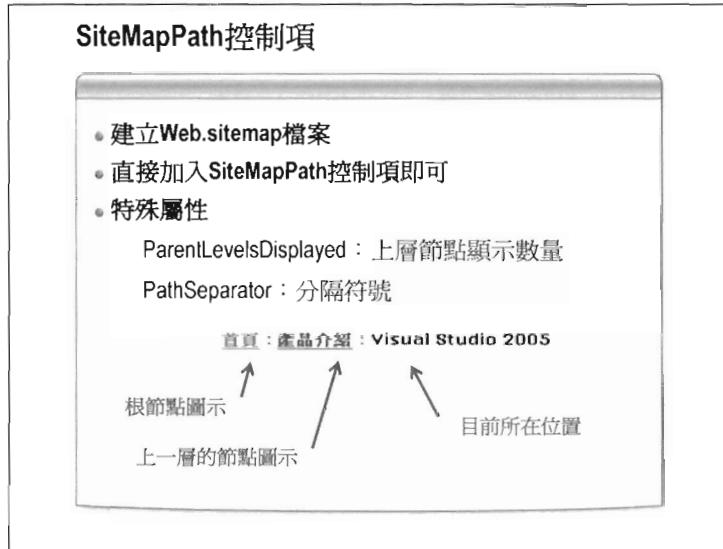
特殊屬性

- ExpandDepth

由於 TreeView 控制項展現時會因為 ExpandDepth 屬性為 FullyExpand 而會展開所有的項目，但是這樣會展現出太大量的內容，也會佔用太多版面。所以需將這個屬性設定為 1，代表僅展開到第二層，0 代表第一層。

- ShowCheckBoxs

該屬性為 Boolean 值，可決定是否要在節點內出現核取方塊，如果設定為 true，即可在程式中利用 TreeView 控制項的 CheckedNodes 屬性來取得被選取的節點，並做相關的處理。



SiteMapPath 伺服器控制項

SiteMapPath 伺服器控制項是 ASP.NET 以路徑方式，顯示目前網頁和其他相關網頁之間關係。

設定 Web.sitemap 檔案

預設資料來源為網站應用程式根目錄下的 Web.sitemap 檔案，所以必須先編寫好。

直接加入 SiteMapPath 控制項即可

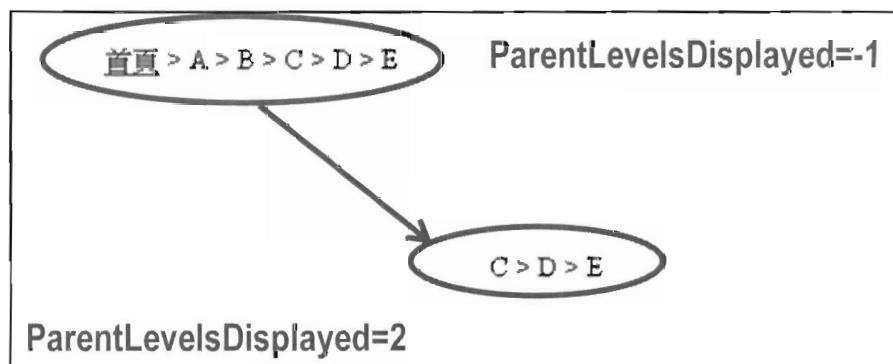
該控制項特別的地方在於不需要搭配 SiteMapDataSource 控制項來使用，預設即會自動去找尋名為 Web.sitemap 的檔案當作其資料來源。所以建立時，只需要去「Toolbox」→「Navigation」內找到 SiteMapPath 伺服器控制項，於網頁上適當位置加入即可。

由於 SiteMapPath 伺服器控制項是網站所有網頁所需要的共用資訊，通常在設計 SiteMapPath 伺服器控制項時，都會將 SiteMapPath 伺服器控制項配置在 Master Page 內。

特殊屬性

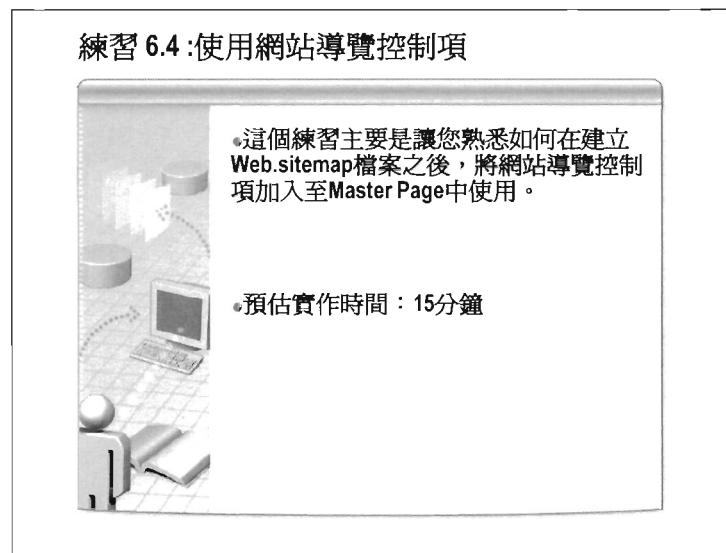
- ParentLevelsDisplayed

有時候，網站的階層關係太過複雜，特別在大型網站內容易發生。由於網頁內容太多，當使用者選取網頁時，其 SiteMapPath 的層級關係可能超過三層以上，甚至到十多層，這對使用者來既不好看又不方便，在 SiteMapPath 伺服器控制項可以使用 ParentLevelsDisplay 屬性用來限制上一層路徑顯示的數量。如下列的設定方式



- PathSeparator

該屬性可以設定每個節點之間的分隔符號，預設值為「:」，可視需要自行修改，修改之後 SiteMapPath 伺服器控制項即以該內容為分隔符號。



練習 6.4 : 使用網站導覽控制項

目的：

這個練習主要是讓您熟悉如何在建立 Web.sitemap 檔案之後，將網站導覽控制項加入至 Master Page 中使用。

功能描述：

這個練習會將會利用已經建立好的 Web.sitemap 檔案當作資料來源，並且將分別建立 Menu、TreeView 與 SiteMapPath 控制項在 Master Page 內來使用。

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「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod06_4\Starter\Mod06_4」目錄，完成後，按下「Open」按鈕即可開啟網站。
3. 在「myMasterPage.master」檔案內，從「Toolbox」→「Data」工具箱中拖拉一個 SiteMapDataSource 控制項至網站標題的下方，完成後在 Design 設計頁面應如下所示：



4. 在「myMasterPage.master」檔案內，從「Toolbox」→「Navigation」工具箱拖拉一個 Menu 控制項至 SiteMapDataSource 控制項下方，點選「Menu Task」自行選擇「Auto Format」內容，Menu 控制項的屬性名稱設定如下：

控制項	屬性	屬性值
Menu	ID	Menu 1
	DataSourceID	SiteMapDataSource1
	Orientation	Horizontal
	StaticDisplayLevels	2

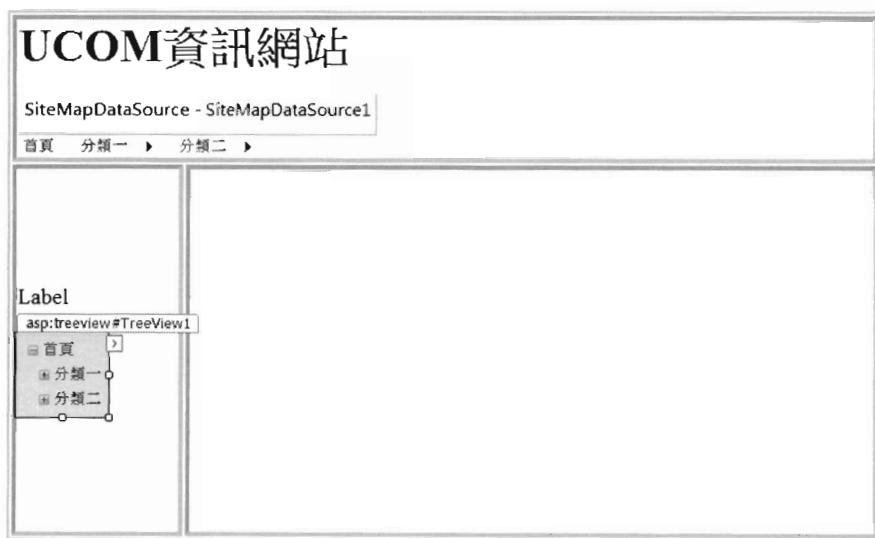
完成之後應如下圖：



5. 在「myMasterPage.master」檔案內，從「Toolbox」→「Navigation」工具箱拖拉一個 TreeView 控制項至表格右下方，點選「TreeView Task」自行選擇「Auto Format」內容，TreeView 控制項的屬性名稱設定如下：

控制項	屬性	屬性值
TreeView	ID	TreeView1
	DataSourceID	SiteMapDataSource1
	ExpandDepth	1

完成後應如下所示：



在「myMasterPage.master」檔案內，從「Toolbox」→「Navigation」工具箱拖拉一個 SiteMapPath 控制項至 Menu 控

制項下方，點選「SiteMapPath Task」自行選擇「Auto Format」內容，SiteMapPath 控制項的屬性名稱設定如下：

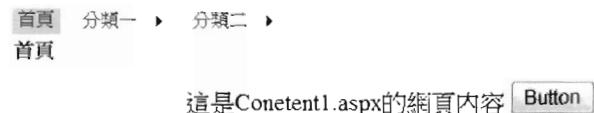
控制項	屬性	屬性值
SiteMapPath	ID	SiteMapPath1
	PathSeparator	>

完成後應如下所示：



6. 完成之後，接著在「Solution Explorer」→分別點選「Content1.aspx」→按滑鼠右鍵→選「View In Browser」。應可使用 Menu 與 TreeView 控制項來變更欲瀏覽的網站內容，而 SiteMapPath 會自動顯示出目前所瀏覽網頁位於網站結構的何處。結果應如下圖：

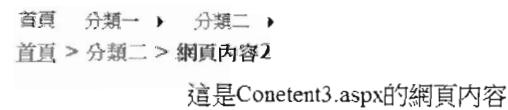
UCOM資訊網站



Label
 直頁
 分類一
 分類二

7. 任意選擇 Menu 與 TreeView 控制項的其他節點後，應如下圖所示：

UCOM資訊網站



Label
 首頁
 分類一
 分類二

總結

- 利用Master Page來設計所有網頁共通的部分
- 瞭解Master Page和Content Page之間的關係
- 瞭解ContentPlaceHolder和Content控制項的關係
- 瞭解如何設計巢狀的Master Page
- 瞭解如何設定SiteMap檔
- 瞭解如何使用Menu控制項
- 瞭解如何使用TreeView伺服器控制項
- 瞭解如何使用SiteMapPath控制項



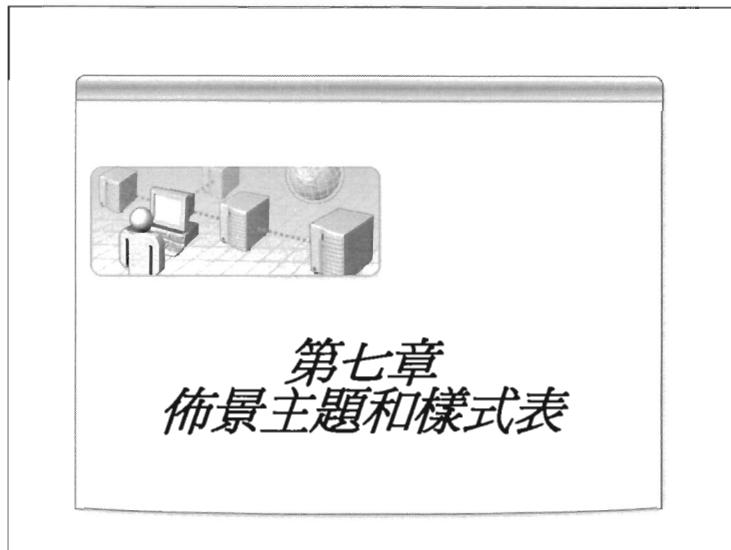
第七章: 佈景主題與樣式表

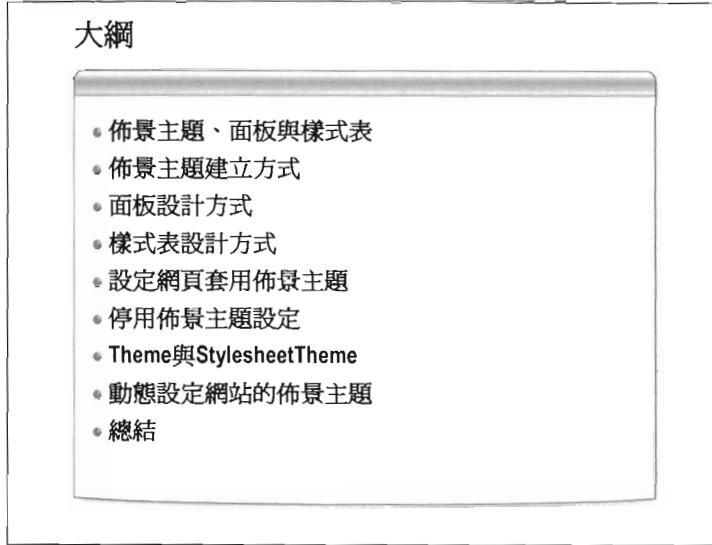
本章大綱

佈景主題、面板與樣式表	4
佈景主題建立方式	6
面板設計方式	7
樣式表設計方式	9
設定網頁套用佈景主題	13
練習 7.1 : 設計佈景主題-利用面板檔	15
練習 7.2 : 設計佈景主題-利用樣式表	18
停用佈景主題設定	21
Theme 與 StylesheetTheme	23
動態變更網頁的樣式	25
練習 7.3 : 動態變更網頁的樣式	26

作者：
周季賢





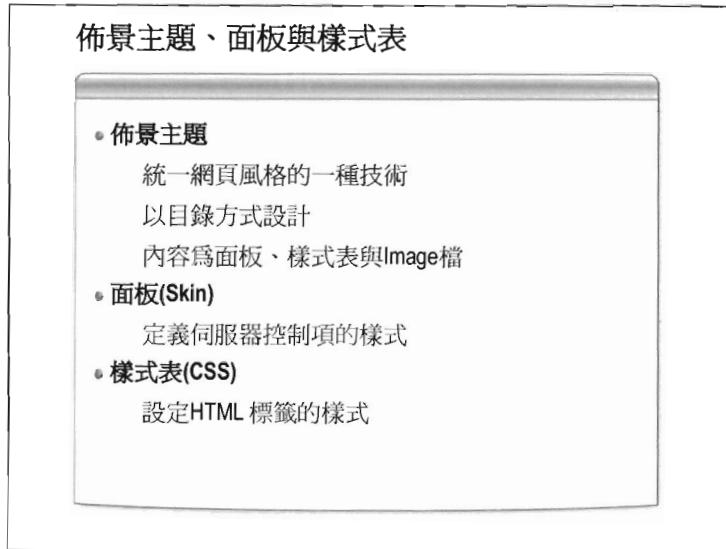


課程大綱

ASP.NET 提供統一網站的樣式的機制，讓所有的網頁和伺服器控制項可以具備相同的風格。本章將說明佈景主題和面板如何建立及在 Web 應用程式中動態變更佈景主題。本章將介紹以下的內容：

本章介紹以下主題：

- 佈景主題、面板與樣式表
- 佈景主題建立方式
- 面板設計方式
- 樣式表設計方式
- 設定網頁套用佈景主題
- 停用佈景主題設定
- Theme 與 StylesheetTheme
- 動態設定網站的佈景主題
- 總結



佈景主題、面板與樣式表

當我們在設計 ASP.NET Web 應用程式的網站外觀時，可以考量的是網站整體的風格是不是需要一致化或易於變動；如果有這樣的需求，那麼，佈景主題、面板與樣式表就是您必須要了解的設計方式。

佈景主題

當你建立一個 Web 應用程式，佈景主題可以讓網站內所有網頁具備相同的樣式，例如：文字的格式、顏色及圖片。這些影響網頁外觀的設定，可以透過佈景主題來定義。無論是一般的 Html 控制項或伺服器控制項，佈景主題皆可以設定其展現的風格。

就如同 Windows 作業系統提供「佈景主題」，讓使用者可以自行決定整個 Windows 作業系統的展現風格，在 ASP.NET Web 應用程式裡則是利用佈景主題來達成這個目標，所有跟樣式有關的設定，全部可以儲存於佈景主題內。

佈景主題是一種目錄，開發人員設計的樣式必須儲存於該目錄內。

包含決定伺服器控制項樣式的面板檔，HTML 控制項的 CSS 檔及圖片檔。

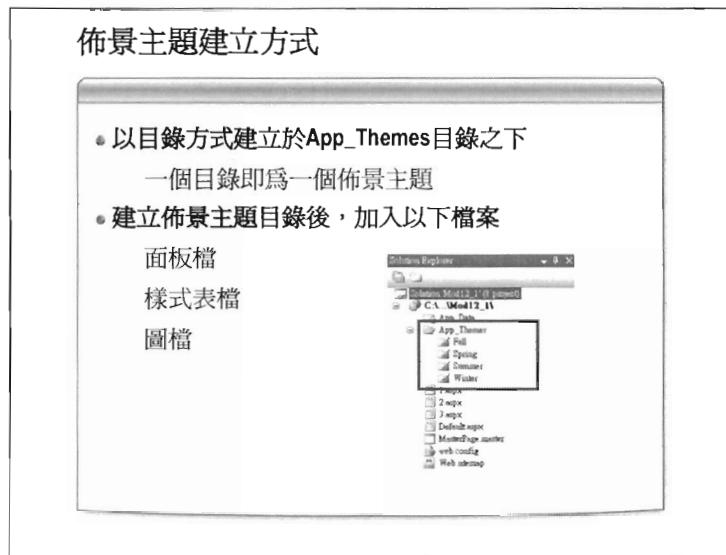
面板

ASP.NET 的伺服器控制項是透過一些設計時期的標籤來決定控制項的內容，包含樣式。其屬性設定內容跟最後在瀏覽器收到的結果並不相同。ASP.NET 利用面板將控制項的設計時期的樣式設定，集中儲放以利重複使用。

如果要在 Web 應用程式加入面板設定，必須在佈景主題目錄下加入面板檔，其副檔名為.skin。

CSS

Cascading Style Sheet(CSS)是用來定義 HTML 控制項的樣式，HTML 的開發人員，將內文的部分透過 HTML 標籤定義，而網頁所需要的樣式則交給 CSS 來處理，這樣的做法，對於網站的維護人員來說，變更網頁的展現風格時，可以在不影響內文的前提下，調整網站的外觀，對於網站開發是一個非常重要的技術。CSS 主要是用來設定 HTML 控制項，對於伺服器控制項，在 ASP.NET 則需要利用佈景主題和面板來定義。

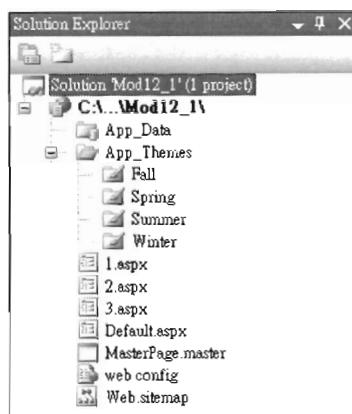


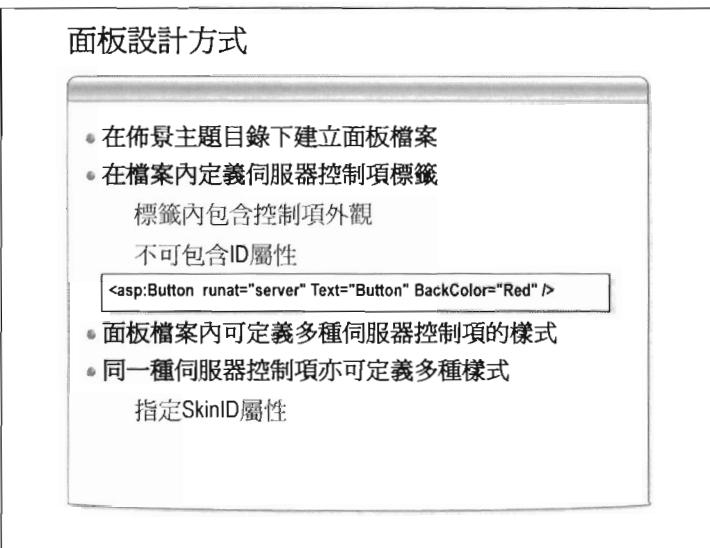
佈景主題建立方式

佈景主題是一個目錄，裡面包含所有影響網站樣式的檔案，例如：面板檔、樣式表檔及各式圖檔。

以目錄的方式建立於 App_Themes 目錄之下

在 Visual Studio 2008 開發工具設定佈景主題，必須先建立一個「App_Themes」目錄，然後在該目錄下，分別對整體網站的樣式儲存於不同的子目錄。例如：想要讓網站根據四個季節呈現風格，可以設計以下的 Spring、Summer、Fall 和 Winter 目錄結構。





面板設計方式

面板檔案是用來定義伺服器控制項的樣式，也是佈景主題目錄中的一員，我們可以利用以下方式來建立。

在佈景主題目錄下建立面板檔案

在 ASP.NET 建立面板檔的方式，必須先建立佈景主題，如果一個網站如果有 5 種佈景主題的話，開發人員可能需要建立 5 個面板檔案，再將這些面板檔分別儲存於不同的佈景主題目錄。

在檔案內定義伺服器控制項標籤

建立面板檔案之後，接著就是在面板檔案內定義伺服器控制項的標籤。定義的伺服器控制項標籤可以設定好任何的屬性外觀，例如顏色或是字型大小等。

標籤寫法基本上與一般在開發控制項時所加入在網頁上的標籤十分類似，唯一需要特別注意的是，面板檔內的控制項標籤是不能擁有 ID 屬性的。

所以最簡單的面板檔標籤建立方式，就是直接在網頁上針對控制項作各種外觀的設定，待設定完成後，再將該控制項標籤由網頁剪下貼至面板檔案內，最後在刪除掉 ID 屬性即大功告成。

面板檔案內可定義多種伺服器控制項的樣式

面板檔案內並沒有限制只能定義一種伺服器控制項樣式，所以可以在一個面板檔內同是定義多種的伺服器控制項樣式，如 Button 與 Label 控制項。

同一種伺服器控制項亦可定義多種樣式

預設面板檔內並不能對同一種伺服器控制項定義兩種以上的樣式，但是實務開發上同一種伺服器控制項具有兩種以上的風格卻屢見不顯，例如同樣是 Button 控制項，可能就會有送出按鈕與取消按鈕的兩種風格。

如果需要針對同一種伺服器控制項定義兩種以上的風格，只需要在針對每一組伺服器控制項的樣式標籤內，都分別加入 SkinID 的屬性，並且分別為他們命名為不同的值即可。如下所示：

```
<asp:Button SkinID="submitStyle" runat="server"  
    BackColor=Blue Text="送出" />  
<asp:Button SkinID="cancelStyle" runat="server"  
    ForeColor=Red Text="離開" />
```

屆時僅需要針對控制項來指定 SkinID，即可決定套用的風格。



樣式表設計方式

Cascading Style Sheet(CSS)是用來定義 HTML 控制項的樣式，也是佈景主題目錄中的一員，我們可以利用以下方式來建立。

在佈景主題目錄下建立樣式表檔案

在 ASP.NET 建立樣式表的方式，必須先建立佈景主題，如果一個網站如果有 5 種佈景主題的話，開發人員可能需要建立 5 個樣式表，再將這些樣式表分別儲存於不同的佈景主題目錄。

在檔案內定義樣式表語法

樣式表語法由兩大部分所構成，分別為 Selector (選取器) 與 declaration(宣告)。每一個 Selector 內可以在「{」與「}」符號內定義多組的 declaration，而每一組 declaration 可使用「;」符號來區隔。而 declaration 則由 property(屬性)與 value(值)組成。基本語法如下：

```
Selector { property : value ; property2 : value2 }
```

而 Selector (選取器)的寫法有下列幾種：

- Element-based styles :

針對 HTML 項目的寫法，僅需要將 Selector (選取器) 設為 HTML 項目的名稱即可，屆時套用該樣式表的網頁即會自動套用樣式至相同名稱的 HTML 項目。

- Class-based styles :

不針對固定的 HTML 項目的寫法，Selector (選取器) 必須要使用「.」加上名稱，如「.myClass」。而套用該樣式表的網頁不會自動套用，必須在開發時自行挑選 HTML 項目之後，在該 HTML 項目內加上「class=Selector 名稱」，如「class=myClass」，才可套用。

- ID-based styles

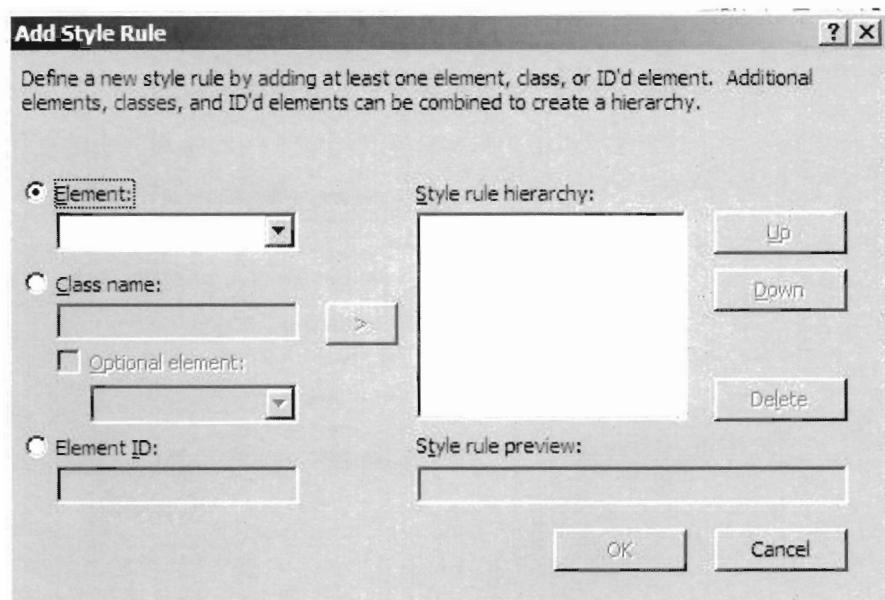
不針對固定的 HTML 項目的寫法，Selector (選取器) 必須要使用「#」加上名稱，如「#Header」。而套用該樣式表的網頁不會自動套用，必須在開發時自行挑選 HTML 項目之後，在該 HTML 項目內指定 ID 名稱，如「ID=Header」，才可套用。

而在編寫樣式表與套用樣式表內容至網頁內容時，Visual Studio 2008 提供了多種的工具來供使用，以下分別介紹各工具。

CSS Creation Tool

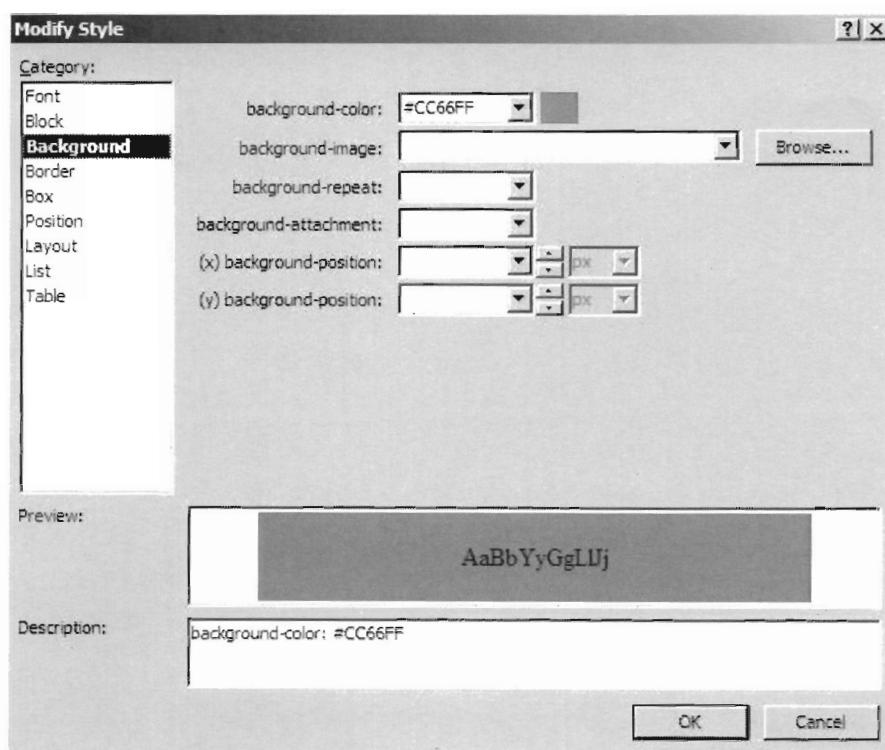
用來建立 CSS 樣式成員，可指定要用 Element-based styles、Class-based styles 或 ID-based styles 來建立 Selector (選取器) 語法，如果選擇 Element-based styles，還可以進一步協助建立階層式的 Selector (選取器)。

該工具可以在編輯樣式表檔案時，利用「Style」→「Add Style Rule」即可開啟。



Style Builder

用來編輯 CSS 樣式成員內容，當 Selector (選取器)建立完成之後，該工具視窗可針對每一個 Selector (選取器)來建立其 declaration(宣告)，使用者只需要在 Category 內選擇大項分類，再針對每一分類內的屬性來設定對應值，declaration(宣告)便會自動建立至 Selector (選取器)內了。



Apply Styles Window

用來套用 CSS 樣式至項目，當網頁引用樣式表之後，可利用該視窗來針對不同的 HTML 控制項樣式內容做樣式表的套用。



設定網頁套用佈景主題

- 網頁層級

設定於網頁的 Page 指示詞中
Theme 屬性指定至佈景主題目錄名稱

- 網站層級

設定於 web.config 檔案中
pages 標籤內 Theme 屬性指定至佈景主題目錄名稱

- 伺服器層級

.NET Framework 目錄下的 web.config 內
pages 標籤內 Theme 屬性指定至佈景主題目錄名稱

設定網頁套用佈景主題

設計好佈景主題之後，便可將之套用到網頁內容內，以下針對不同的層級套用分別做說明。

網頁層級

ASP.NET 可以將設計好的佈景主題，在 Web Form 的網頁中透過修改 Page 指示詞的 Theme 屬性來選擇所需要的佈景主題。在 .aspx 的網頁內，對 Page 加上 Theme 屬性，指定設計的佈景主題目錄名稱，就可以讓該網頁套用對應的佈景主題目錄下樣式設定，如下：

```
Visual Basic
<%@ Page Language="VB" Theme="Winter" %>
```

```
C#
<%@ Page Language="C#" Theme="Winter" %>
```

網站層級

除了可以在個別網頁設定佈景主題外，逐一設定網頁的布景主題太辛苦了，如果網站內所有的網頁都是相同的樣式，ASP.NET 提供在

web.config 設定應用程式層級的佈景主題的方式。設定於網站內的 web.config 檔案之後，該網站內的所有網頁即會套用該設定。如下所示：

```
<configuration>
  <system.web>
    <pages theme="Winter"></pages>
  </system.web>
</configuration>
```

伺服器層級

當整個伺服器內所有的網站都需要相同的佈景主題，ASP.NET 同樣也有提供設定的方式，這種設定方式，通常用來定義網站的預設的佈景主題。例如：企業的對內、對外網站都需要具備該企業風格的佈景主當作預設的佈景主題，當該企業建立新的網站時，自動會選取伺服器層級的設定。

.NET 平台對伺服器層級的設定是透過.NET Framework 安裝目錄下的 web.config 檔案來設定，如果希望在伺服器下的所有 Web 應用程式使用相同的佈景主題設定，可以透過記事本去編輯伺服器層級的 web.config 的內容。

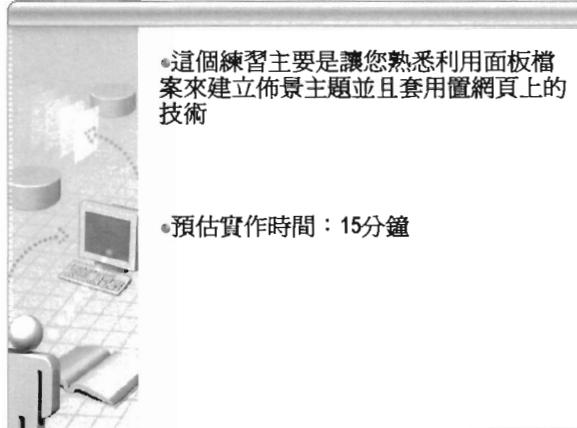
在伺服器層級的 web.config 路徑在.NET Framework 的 CONFIG 目錄下，例如：

```
C:\Windows\Microsoft.NET\Framework\v.2.0.XXXX\CONFIG
```

加上以下的設定即可以讓該伺服器下所有 Web 應用程式使用相同的佈景主題設定：

```
<pages theme="Winter" />
```

練習 7.1 : 設計佈景主題-利用面板檔



練習 7.1 : 設計佈景主題-利用面板檔

目的：

這個練習主要是讓您熟悉利用面板檔案來建立佈景主題並且套用置網頁上的技術。

功能描述：

這個練習會在建立一個佈景主題與一張網頁，並在佈景主題中加入一個面板檔，在面板檔內分別描述兩組的 Button 控制項與一個 Label 控制項外觀，最後再將之套用到網頁上。

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「Empty Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9543\Practices\VB 或 CS\Mod07_1\Starter」目錄，與使

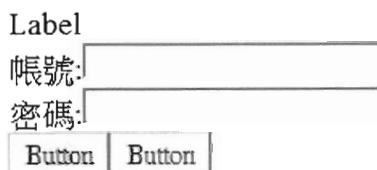
用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod07_1」。

3. 建立佈景主題目錄。在專案目錄下，按滑鼠右鍵選取「Add ASP.NET Folder」→「Theme」來建立佈景主題，建立後的佈景主題資料夾名稱命名為「myTheme」。
4. 接著建立面板檔，在佈景主題目錄「myTheme」按下滑鼠選取「Add New Item」，選擇「Skin File」並且命名為「mySkinFile.skin」。
5. 接著在「mySkinFile.skin」加入以下面板描述：

```
<asp:Label runat="server" Text="UCOM網站登入畫面"
    Font-Size="Larger" Font-Bold="True"
    Font-Underline="True"></asp:Label>
<asp:Button SkinID="loginBtn" runat="server"
    Text="登入" BackColor="#66FF99" />
<asp:Button SkinID="cancelBtn" runat="server"
    Text="取消" BackColor="#CC66FF"
    ForeColor="Red" />
```

6. 完成之後，自主選單「Web Site」下「Add New Item...」，選取「Web Form」，檔名命名為「testTheme.aspx」，清除「Place code in separate file」核取方塊，建立來源網頁。
7. 在「testTheme.aspx」網頁內，從「Toolbox」工具箱中拖拉一個 Label 控制項、兩個 TextBox 控制項與兩個 Button 控制項，並且適當編排，完成後標籤與外觀應如下所示：

```
<asp:Label ID="Label1" runat="server" Text="Label">
</asp:Label>
<br />
帳號:<asp:TextBox ID="TextBox1" runat="server">
</asp:TextBox><br />
密碼:<asp:TextBox ID="TextBox2" runat="server">
</asp:TextBox><br />
<asp:Button ID="Button1" runat="server" Text="Button" />
<asp:Button ID="Button2" runat="server" Text="Button" />
```



8. 接著設定 ID 為 Button1 的 Button 控制項，將其 SkinID 屬性設定為「loginBtn」

9. 接著設定 ID 為 Button2 的 Button 控制項，將其 SkinID 屬性設定為「cancelBtn」。

10. 最後在「testTheme.aspx」的 Page 指示詞中將「Theme」屬性設定為「myTheme」即可，如下所示：

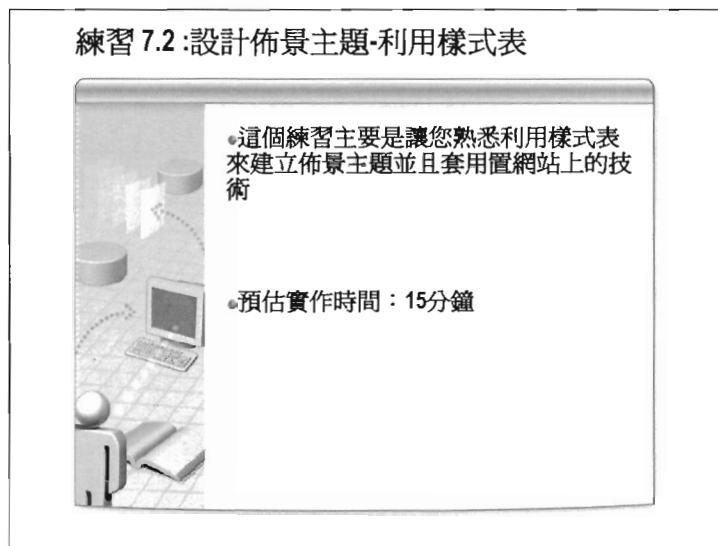
```
<%@ Page Language="C#" Theme="myTheme" %>
```

11. 執行網頁測試。在「Solution Explorer」→點選「testTheme.aspx」網頁→按滑鼠右鍵→選「View In Browser」。結果如下所示：

UCOM網站登入畫面

帳號: _____

密碼:



練習 7.2 :設計佈景主題-利用樣式表

目的：

這個練習主要是讓您熟悉利用樣式表來建立佈景主題並且套用置網站上的技術。

功能描述：

這個練習會延續練習 7.1 的網站，在佈景主題內加入樣式表來更進一步的設計網頁外觀，並且將套用層級提升至應用程式層級來使用。

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. 本練習為上一練習的延續，如果您有完成上一練習，請直接開啓上一練習的網站即可。如果沒有，可以從「File」→「Open」→「Web Site」→選取「File System」按鈕並選取

「\U9543\Practices\VB 或 CS\Mod07_2\Starter\ Mod07_2」目錄，完成後，按下「Open」按鈕即可開啟網站。

3. 接著建立樣式表，在佈景主題目錄「myTheme」按下滑鼠選取「Add New Item」，選擇「StyleSheet」並且命名為「myStyleSheet.css」。
4. 完成之後，可利用 CSS Creation Tool 與 Style Builder 或直接手動編輯樣式表內的「body」與「.divClass」Selector，完成後樣式應如下所示：

```
body
{
    background-color: #9999FF;
}
.divClass
{
    border-style: double;
    position: absolute;
    top: 37%;
    right: 40%;
    background-image:url('bg.png');
}
```

5. 將「testTheme.aspx」的 Page 指示詞中將「Theme」屬性取消即可，完成後 Page 指示詞應如下所示：

```
<%@ Page Language="C#" %>
```

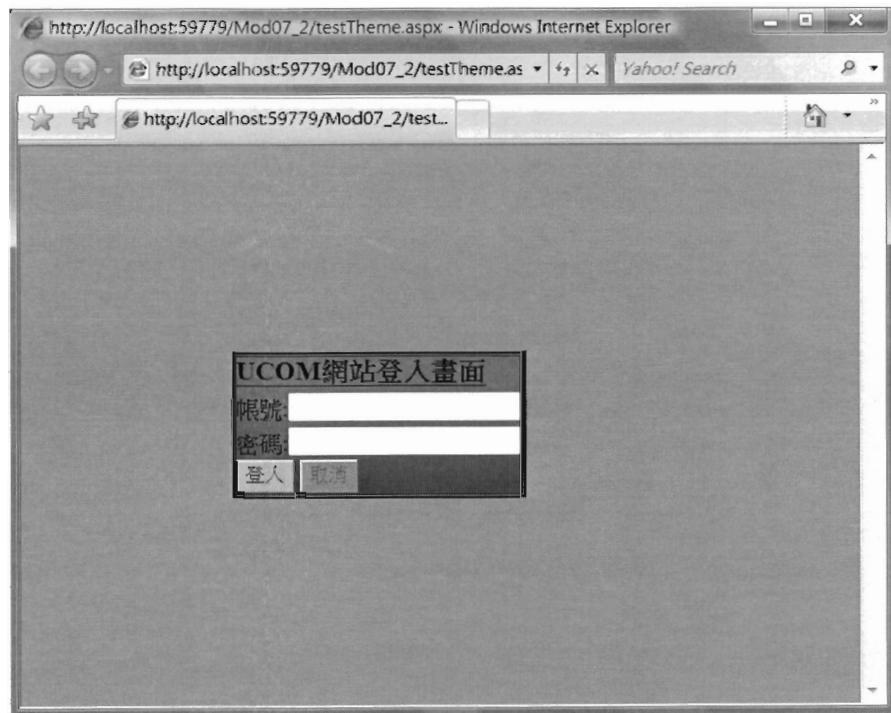
6. 將「testTheme.aspx」的內「div」標籤的「Class」屬性設定為「divClass」，完成後標籤應如下所示：

```
<div class="divClass" >
<asp:Label ID="Label1" runat="server" Text="Label">
</asp:Label>
<br />
帳號:<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br />
密碼:<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox><br />
<asp:Button ID="Button1" runat="server" Text="Button" SkinID=
"loginBtn" />
<asp:Button ID="Button2" runat="server" Text="Button" SkinID=
"cancelBtn" />
</div>
```

7. 加入 web.config 檔案。自「Web Site」下「Add New Item...」，選取「Web Configuration File」，檔名命名為「Web.config」，建立 web.config 檔案。
8. 在 web.config 檔案的「configuration」→「system.web」內加入以下內容：

```
<pages theme="myTheme"></pages>
```

9. 執行網頁測試。在「Solution Explorer」→點選「testTheme.aspx」網頁→按滑鼠右鍵→選「View In Browser」。結果應如下所示：



停用佈景主題設定

- 應用程式層級

設定於web.config檔案中

pages標籤內Theme屬性指定為空字符串

- 網頁層級

設定於網頁的 Page 指示詞中

EnableTheming屬性設定為false

- 伺服器控制項層級

設定於控制項標籤中

EnableTheming屬性設定為false

停用佈景主題設定

如果不分對象不需要套用到整體佈景主題的設定，可以依下面的層級分別停用。

應用程式層級

取消 Web 應用程式層級以上的佈景主題設定，包含伺服器層級。分別可以在 web.config 檔案內將佈景主題屬性設定成空字符串。如下所示：

```
<configuration>
  <system.web>
    <pages theme=""></pages>
  </system.web>
</configuration>
```

網頁層級

如果設定了應用程式層級以上的佈景主題設定，但是單一網頁卻不需要套用，那麼可以在該網頁內將 Page 指示詞的 EnableTheming 屬性設定成 false。如下：

```
<% @ Page EnableTheming="false" %>
```

伺服器控制項層級

如果設定了網頁層級以上的佈景主題設定，但是網頁中的某一個控制項卻不需要套用，那麼可以使用控制項的 EnableTheming 屬性，如下：

```
<asp:TextBox EnableTheming="false" runat="Server"></asp:Text  
Box>
```

Theme 與 StylesheetTheme

- 與佈景主題套用至網頁的優先順序有關

- **Theme**

以佈景主題內容為主

會覆寫套用對象於網頁內的設定

- **StylesheetTheme**

以網頁內容為主

佈景主題內容僅會附加，不會覆寫。

Theme 與 StylesheetTheme

在套用佈景主題至網頁內容時，還有一個需要考慮的問題，就是同一個樣式的設定如果同時出現在網頁內容與佈景主題設定內，那應該如何取捨？這時候就可以利用 Theme 與 StylesheetTheme 這兩個屬性來分別決定套用的優先權。

以下分別說明其不同之處：

- **Theme**

以佈景主題內容為主，如果設定重複，則會覆寫套用對象於網頁內的設定。

- **StylesheetTheme**

以網頁內容為主，如果設定重複，則佈景主題內容則不會覆寫，但是還是會將不重複但佈景主題有的設定附加至套用對象。

以下使用 Image 控制項作為說明，例如有一佈景主題名為「aaa」，內有一面板檔，面板檔內容如下：

```
<asp:Image runat="server"
```

```
ImageUrl="~/App_Themes/testStyleSheetTheme/UCOM.gif" />
```

而網頁內有兩個 Image 指令項，設定標籤如下：

```
<asp:Image ID="Image1" runat="server" />
<br />
<asp:Image ID="Image2" runat="server"
    ImageUrl="~/netmag.gif" />
```

可以看出，ID 為 Image2 的控制項已經設定了其 ImageUrl 的屬性，如果套用佈景主題「aaa」，那麼將會該屬性將會重複。

如果將 Page 指示詞的 Theme 屬性設定成「aaa」來套用，那麼 Image2 控制項的內的屬性設定將會被覆寫，結果將如下所示：



如果改為將 Page 指示詞的 StyleSheetTheme 屬性設定成「aaa」來套用，那麼 Image2 控制項的內的屬性設定將不會被覆寫，結果將如下所示：



動態變更網頁的樣式

- 針對Page物件的Theme屬性變更
- 僅能在Page的PreInit事件進行

```
Visual Basic
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As
System.EventArgs)
    Page.Theme = Request.QueryString("ThemeID")
End Sub
```

```
C#
protected void Page_PreInit(object sender, EventArgs e)
{
    Page.Theme = Request.QueryString("ThemeID");
}
```

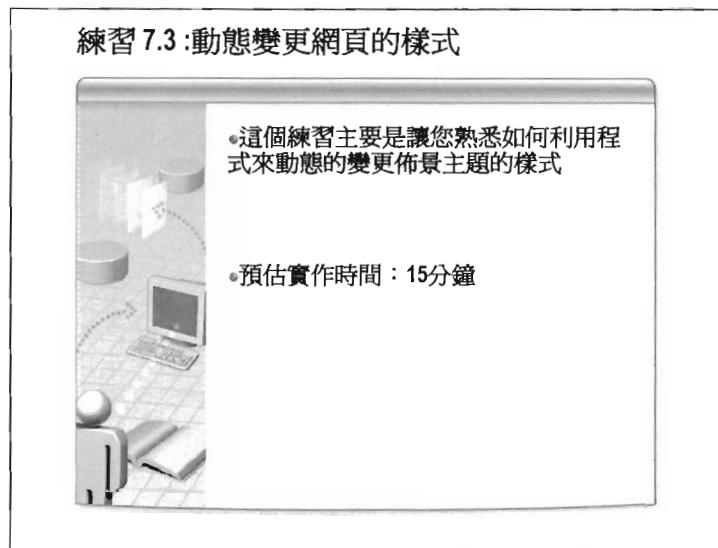
動態變更網頁的樣式

前面所介紹的方式主要是透過宣告的方式來定義網站、網頁或控制項所需要的樣式，除此之外，也可以利用撰寫程式來動態控制佈景主題。

改變網頁的樣式，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 = Request.QueryString("ThemeID")
    End Sub
</script>
```

```
C#
<script runat="server">
    protected void Page_PreInit(object sender, EventArgs e)
    {
        Page.Theme = Request.QueryString("ThemeID");
    }
</script>
```



練習 7.3 : 動態變更網頁的樣式

目的：

這個練習主要是讓您熟悉如何利用程式來動態的變更佈景主題的樣式。

功能描述：

這個練習準備了兩組的佈景主題，並且尚未在網頁上指定任何佈景主題，在練習中，將會建立一張新網頁，在新網頁加入兩個超連結選項指定至欲套用佈景主題的網頁，並且利用超連結的QueryString 來讓欲套用佈景主題的網頁能在 Page_PreInit 事件動態更換佈景主題內容。

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「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod07_3\Starter\Mod07_3」目錄，完成後，按下「Open」按鈕即可開啓網站。
3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，檔名命名為「testChangeTheme.aspx」，清除「Place code in separate file」核取方塊，建立來源網頁。
4. 在「testChangeTheme.aspx」網頁內，從「Toolbox」工具箱中拖拉兩個 HyperLink 控制項，控制項的名稱屬性設定如下所示：

控制項	屬性	屬性值
HyperLink	ID	HyperLink1
	Text	佈景主題 1
	NavigateUrl	~/testTheme.aspx?theme=myTheme
HyperLink	ID	HyperLink2
	Text	佈景主題 2
	NavigateUrl	~/testTheme.aspx?theme= <u>myTheme</u>

，完成後設計頁面應如下所示：

佈景主題1

佈景主題2

5. 在「testTheme.aspx」網頁內建立網頁的 PreInit 事件處理常式，建立後程式碼應如下所示：

```
Visual Basic
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As System.EventArgs)

End Sub
```

```
C#
protected void Page_PreInit(object sender, EventArgs e)
{



}
```

6. 在網頁的 PreInit 事件處理常式程式碼區塊中加入下面的程式碼：

```
Visual Basic
Protected Sub Page_PreInit(ByVal sender As Object, ByVal e As System.EventArgs)
    Page.Theme = Request.QueryString("theme")
End Sub
```

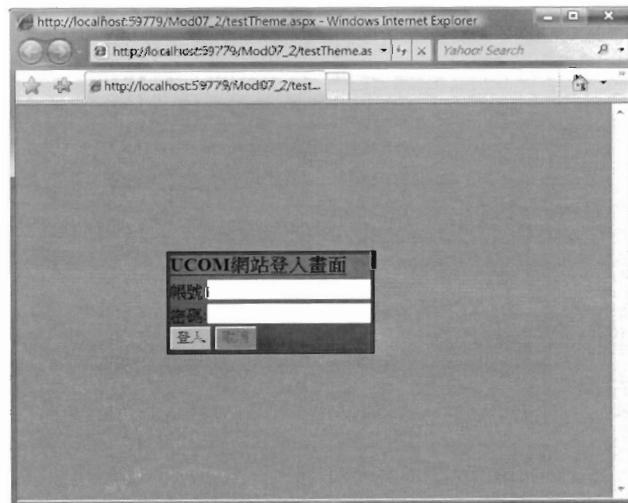
```
C#
protected void Page_PreInit(object sender, EventArgs e)
{
    Page.Theme = Request.QueryString["theme"];
}
```

7. 執行網頁測試。在「Solution Explorer」→點選「testChangeTheme.aspx」網頁→按滑鼠右鍵→選「View In Browser」。畫面應如下所示：

佈景主題1

佈景主題2

8. 點選「佈景主題 1」後，網頁會被導向至「testTheme.aspx」網頁，畫面應如下所示：



9. 點選「佈景主題 2」後，網頁會被導向至「testTheme.aspx」網頁，畫面應如下所示：



總結

- 佈景主題是用來統一網站的風格
- 面板可以設定伺服器控制項的樣式
- 對個別網頁套用佈景主題
- 對整個網站套用佈景主題
- 讓使用者動態變更佈景主題

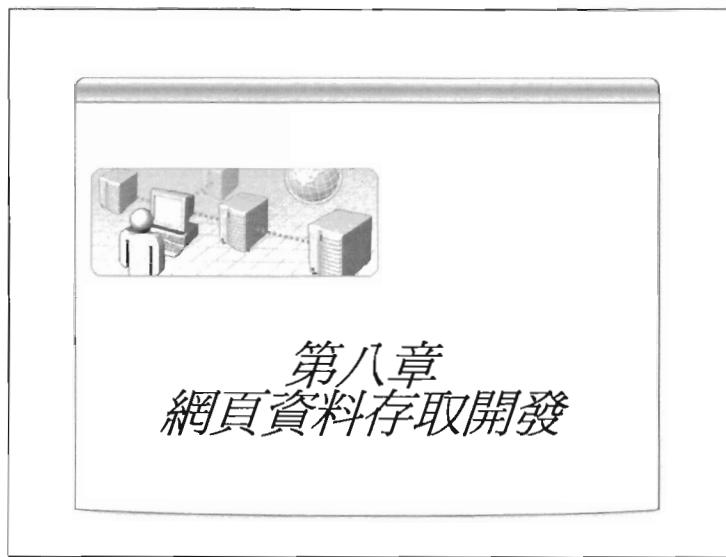
第八章：網頁資料存取開發

本章大綱

ADO.NET 簡介	4
使用 Visual Studio 2008 工具管理資料庫	5
練習 8.1：利用 Visual Studio 2008 管理資料庫	7
ASP.NET 資料來源模型 (Data Source Model)	11
資料繫結語法	15
GridView 控制項	16
練習 8.2 : 使用 GridView 控制項	17
FormView 控制項	21
練習 8.3 : 使用 FormView 控制項	22
DetailsView 控制項	25
DataList 控制項	26
Repeater 控制項	28
ListView 控制項	29
練習 8.4 : 使用 ListView 控制項	31
其他相關控制項	35

作者：
周季賢





大綱

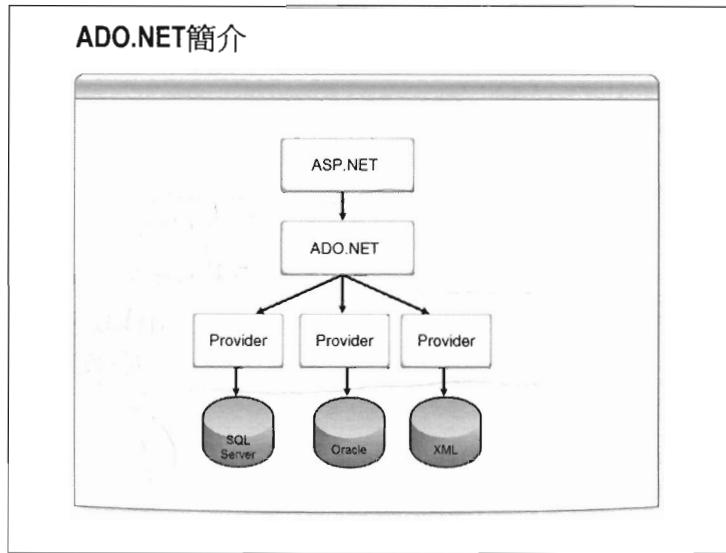
- ADO.NET簡介
- Visual Studio 2008管理資料庫工具
- ASP.NET資料來源模型簡介
- 資料來源控制項
- 資料繫結控制項
- 資料繫結語法
- 其他相關控制項
- 總結

大綱

在舊.NET時期提供了 ADO.NET 的資料存取架構供程式設計師們可以開發資料存取的網頁。而在新一代的 ASP.NET 中，則將網頁資料存取的技術更上一層樓，可讓程式設計師們不需太深入了解資料操作的核心技術，就可以輕鬆使用 ASP.NET 的資料來源模型來開發資料存取網頁。

本章介紹以下主題：

- ADO.NET 簡介
- Visual Studio 2008 管理資料庫工具
- ASP.NET 資料來源模型簡介
- 資料來源控制項
- 資料繫結控制項
- 資料繫結語法
- 其他相關控制項
- 總結

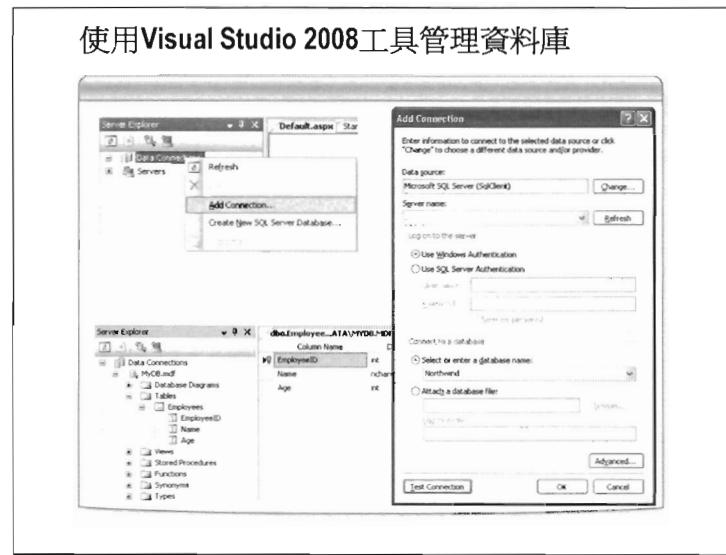


ADO.NET 簡介

ADO.NET 是.NET 以往用來存取各種資料來源技術的統稱，該技術是將與各種資料來源溝通的底層運作技術封裝起來，並提供一致化的資料存取方式。

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。

不過在新的開發工具之中，你不需要直接使用 ADO.NET 的物件，ASP.NET 提供一個中介層，將底層 ADO.NET 的物件包裝起來，利用一大堆新的資料來源控制項 (Data Source Control) 讓你利用更少的程式碼就可以達到資料存取的效果。



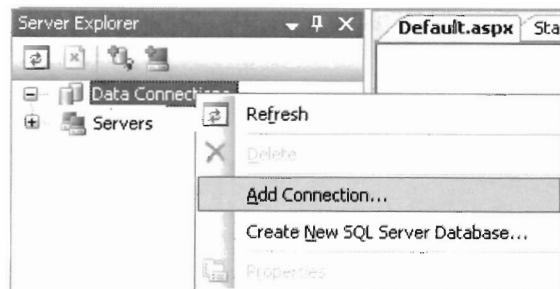
使用 Visual Studio 2008 工具管理資料庫

Visual Studio 2008 開發工具可以連接到 SQL Server、SQL Server Express...等資料庫進行例常的管理工作。你可以使用伺服器總管 (Server Explorer) 連接到各式各樣資料來源。

在 Visual Studio 2008 工具中，你可以連接到資料來源、建立本機資料庫、新增資料表、建立欄位、輸入資料、查詢資料、部署資料庫...等等作業。

連接到資料庫的步驟：

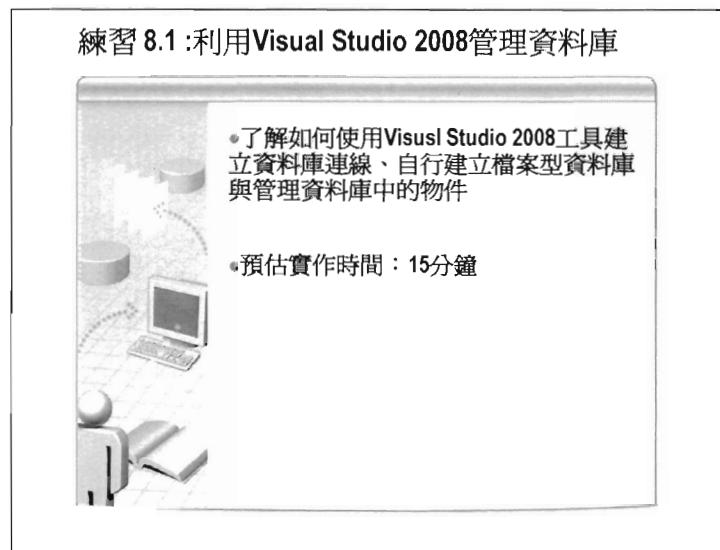
1. 開啓「Server Explorer」視窗，在「Data Connections」節點上
→按滑鼠右鍵→選「Add Connection」。



2. 在「Add Connection」視窗中，選取適當的資料來源 (Data Source) 和資料庫伺服器名稱、驗證方式與資料庫。



利用「Server Explorer」視窗進行資料庫的管理作業。



練習 8.1：利用 Visual Studio 2008 管理資料庫

目的：

了解如何使用 Visusl Studio 2008 工具建立資料庫連線、自行建立檔案型資料庫與管理資料庫中的物件。

功能描述：

在這個練習中，將會使用 Visusl Studio 2008 工具建立一個與資料庫的連線，並且在建立連線之後利用 Visusl Studio 2008 工具來建立一個檔案型資料庫，最後再利用 Visusl Studio 2008 工具來管理該資料庫內容。

預估實作時間：15 分鐘

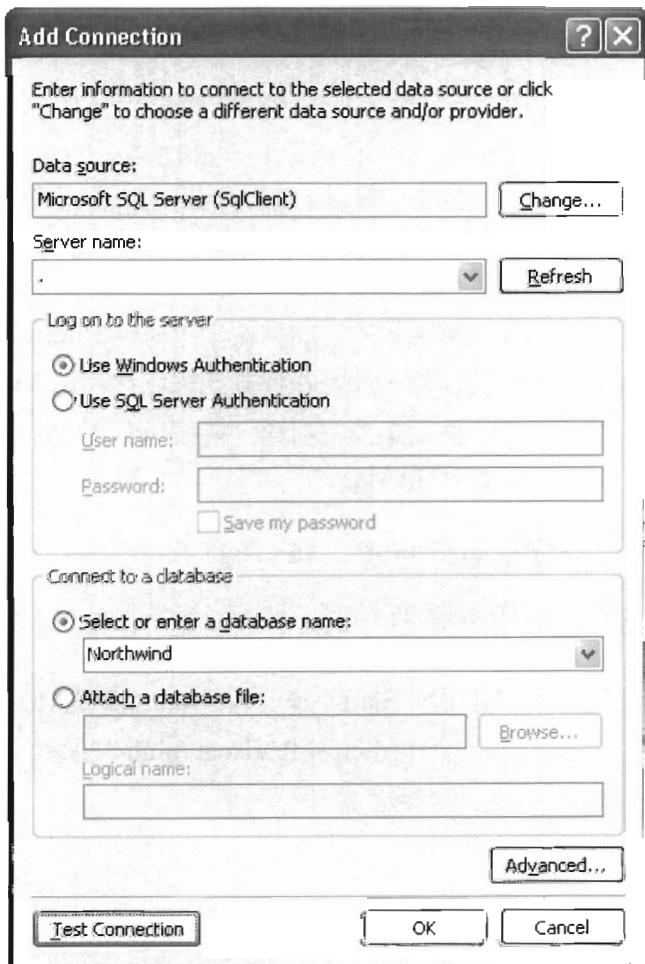
實作步驟：

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「Empty Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕

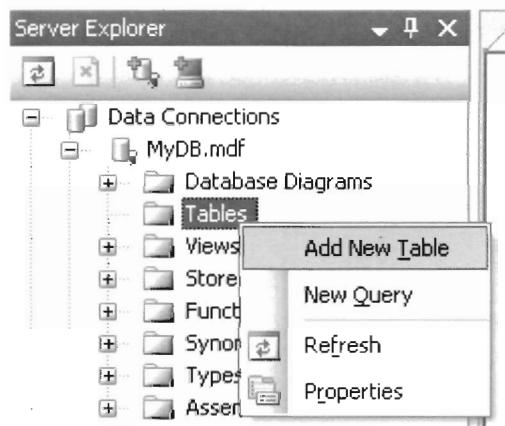
選取「\U9543\Practices\VB 或 CS\Mod08_1\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod08_1」。

3. 開啓「Server Explorer」，在「Data Connection」節點→按滑鼠右鍵→選「Add Connection」。
4. 在「Add Connection」視窗中，設以下屬性：
 - 資料來源 (Data Source)：Microsoft SQL Server (SQLClient)。
 - Server name 欄位：輸入「」。
 - Log on to the server 欄位：選擇「Use Windows Authentication」。
 - Select or enter a database name 欄位：選擇 Northwind 資料庫。

完成後應如下圖所示：



5. 按「Test Connection」按鈕，確認是否可以正確連接到資料庫。按「OK」結束。您已經建立一個 SQL Server 的連線。
6. 在專案之中加入資料庫檔案線。在「Solution Explorer」點選「Mod08_1」節點，按滑鼠右鍵，「Add New Item」選「SQL Server Database」，取一個名稱，如「MyDB.mdf」，按「Add」，若有任何訊息，則按「OK」。檢視「Server Explorer」視窗，在「Data Connections」節點下將出現 MyDB.mdf 的連線。
7. 新增一個資料表。在「Tables」→按滑鼠右鍵→「Add New Table」。



8. 參考下圖，建立 EmployeeID、Name、Age 三個欄位：

Column Name	Data Type	Allow Nulls
EmployeeID	int	<input type="checkbox"/>
Name	nchar(50)	<input checked="" type="checkbox"/>
Age	int	<input checked="" type="checkbox"/>

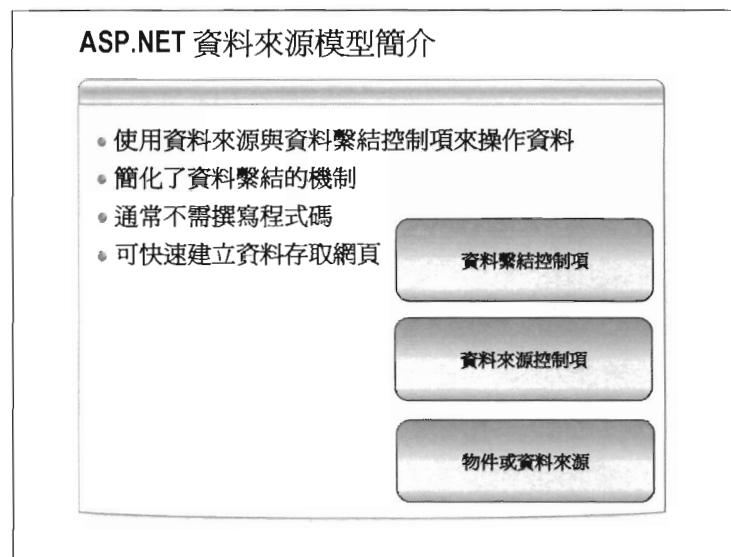
9. 將 EmployeeID 設為主鍵，點選設計畫面上的 EmployeeID 欄位→按滑鼠右鍵，選「Set Primary Key」。

Column Name	Data Type	Allow Nulls
EmployeeID		
Name		<input checked="" type="checkbox"/>
Age		<input checked="" type="checkbox"/>

Set Primary Key
 Insert Column
 Delete Column
 Relationships

10. 儲存資料表，將資料表名稱命為「Employees」。
11. 輸入員工資料。在「Server Explorer」視窗 → MyDB 連線 → Tables → 「Employees」資料表 → 按滑鼠右鍵 → 「Show Table Data」。
12. 輸入幾筆測試的員工資料，如：

	EmployeeID	Name	Age
	1	Rose	... 25
	2	Mary	... 30
	3	Bet	... 32
**	NULL	NULL	NULL



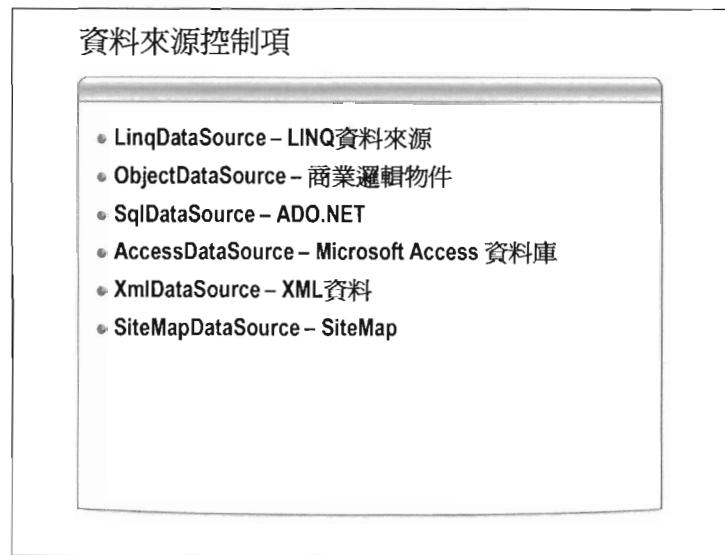
ASP.NET 資料來源模型 (Data Source Model)

ASP.NET 資料來源模型簡化了資料繫結的機制。ASP.NET 使用了資料來源控制項 (Data Source Control) 來協助程式設計師們與各種的不同的資料來源或物件溝通，與直接操作物件與資料來源的最大不同處在於，使用資料來源控制項只需要決定要從何處取得資料，與如何初始化資料查詢的操作即可。

使用資料來源模型的好處包含：

- 提供簡化的程式模型，利用進階的控制項操作複雜的動作，如執行 I/O。
- 所有通用的資料存取動作都不需要撰寫任何一行程式碼。

除了使用資料來源控制項來簡化資料的操作，ASP.NET 資料來源模型的繫結技術也是在設計上的一大助力；ASP.NET 的新資料繫結技術使用了資料繫結控制項來與資料來源控制項互相結合，讓程式設計師能夠輕易的完成網頁介面的設計，該技術能夠很簡單的完成網頁介面的資料操作，或是進階的資料分頁技巧、排序處理，除了設計簡易之外，更能讓程式設計師增加網站建置的產能。



資料來源控制項

資料來源控制項 (Data Source Control) 能夠協助我們與各種不同的資料來源或物件溝通，利用資料來源控制項來操作資料時，不需要了解太底層的資料存取方式，通常只需要知道資料位於何處與資料查詢的操作即可。以下介紹簡介各種資料來源控制項。

- LinqDataSource

能夠使用語言整合查詢(LINQ)來建立 ASP.NET 網頁，該控制項也支援各種資料操作，包含了查詢、修改、新增和刪除功能，也支援資料的排序、篩選與分頁的能力。

- ObjectDataSource

利用.NET 自訂的商業邏輯物件(Business Object)來操作資料，這個控制項可以讓程式設計師開發三層式 (3-Tier)(或稱分散式)應用程式。

- SqlDataSource

並非只能用在 SQL Server 上，而是能夠與任何 ADO.NET 的 data Provider 可支援的所有資料來源作溝通，包含透過 OLE DB 或 ODBC 存取的資料來源。

- AccessDataSource

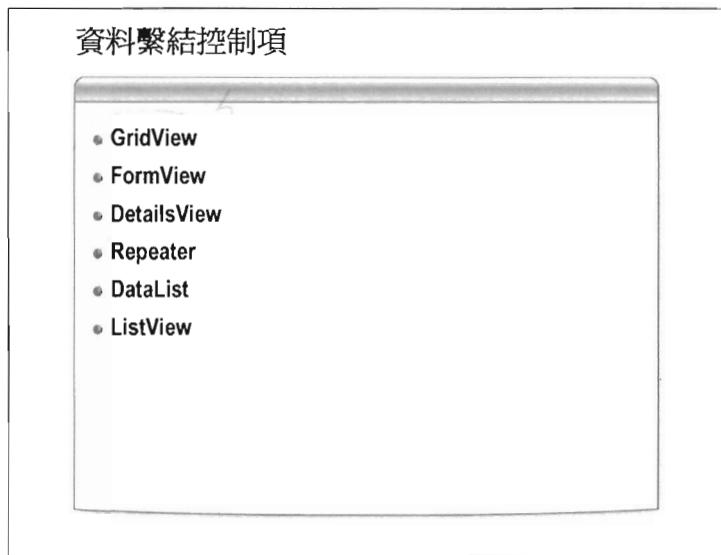
代表一個 Access 資料庫的連線。它繼承自 SqlDataSource 控制項，利用 DataFile 屬性指向真正的 MDB 資料庫檔案。此控制項使用 Jet 4.0 OLE DB provider 連接到資料庫。

- XmlDataSource

同時支援表格式與階層式的資料。你可以把 XmlDataSource 控制項繫結到其它控制項，如 TreeView 以展示樹狀結構的資料。

- SiteMapDataSource

提供設計與展示網站地圖的功能。利用圖型展示網站中所有的網頁以及目錄的資訊。



資料繫結控制項

ASP.NET 提供許多控制項，讓程式設計師建立資料驅動的網頁。只要搭配資料來源控制項並詳加設定之後，就能輕易的夠將資料庫的資料展現在網頁上。

常見的資料繫結控制項 (Data-Bound Control) 如下：

- GridView
- FormView
- DetailsView
- Repeater
- DataList
- ListView

資料繫結語法

- **Eval** 方法
單向的
僅能顯示資料

```
<%# Eval("Name") %>
```
- **Bind** 方法
雙向的
可以讀取資料，也可以修改資料

```
<%# Bind("Name") %>
```

資料繫結語法

ASP.NET 使用新版的資料繫結語法，讓 DataBinder 類別的語法變成更為精簡。利用該語法能夠輕易的將網頁內容與資料內容結合。

Eval 方法

Eval 方法是一個包裝程式，以 DataBinder.Eval 為基礎建立的。語法如下：

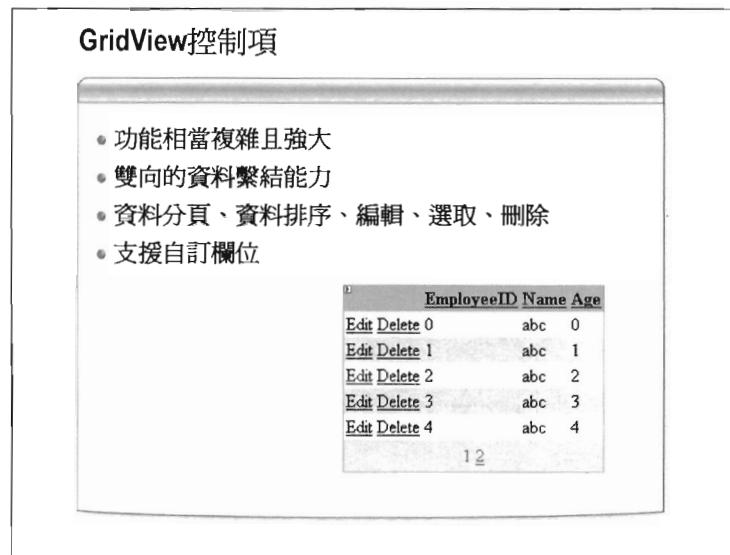
```
<%# Eval("Name") %>
```

Bind 方法

Bind 方法和 Eval 方法類似，語法為：

```
<%# Bind("Name") %>
```

Bind 方法和 Eval 方法兩者的差別在於 Eval 方法是單向的，而 Bind 方法是雙向的。



GridView 控制項

GridView 控制項是一個功能相當複雜且強大的控制項。提供雙向的資料繫結能力，。這個控制項和資料來源控制項緊密地結合在一起，並能夠直接新增資料到資料來源，也可以修改資料來源中的資料。此外你也可以輕易地利用它來設計資料分頁、資料排序功能。

GridView 控制項可以顯示表格式的資料。表格中的每個欄位 (Column) 代表資料來源中的欄位 (Data Source Column)，每一筆資料 (Row)，就代表資料來源中的一筆紀錄。

GridView 控制項也可建立自訂欄位來客製化外觀，可加入的自訂欄位包含：CheckBoxField、HyperLinkField、ImageField、ButtonField、CommandField 與 TemplateField 等等。

設計 GridView 的方法相當簡單，僅需要先設定好資料來源控制項，並且將資料來源控制項的 ID 指定給 GridView 的 DataSourceID 屬性，最後再決定要啓用 GridView 控制項的何種功能即可。

練習 8.2 :使用 GridView 控制項



•了解如何使用 GridView 與 SqlDataSource 控制項來建立資料存取的網頁。

•預估實作時間：15分鐘

練習 8.2 :使用 GridView 控制項

目的：

這個練習主要是讓您熟悉在了解如何使用 GridView 控制項來展現資料、編修資料。

功能描述：

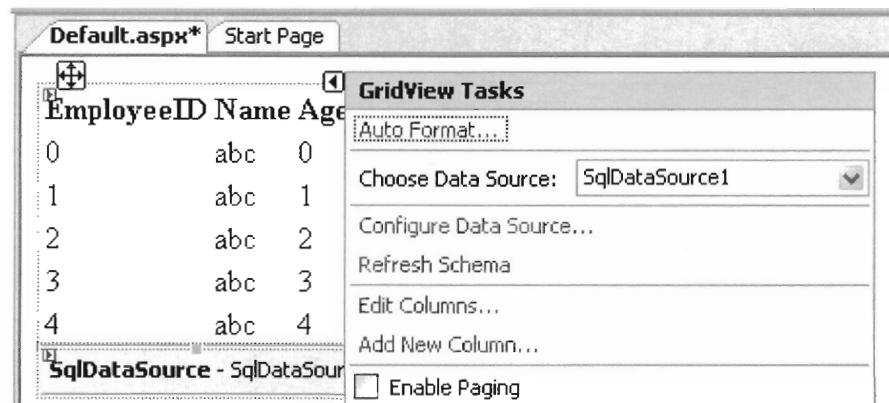
在這個練習中，將利用 GridView 控制項展現 Employees 資料表的員工資料。並利用 GridView 控制項的功能設計資料分頁、資料排序功能。以及進行資料編輯，再把編輯過的結果儲存到資料庫。

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「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod08_2\Starter\Mod08_2」目錄，完成後，按下「Open」按鈕即可開啓網站。
3. 建立一個新網頁。從 Visual Studio 2008 開發工具「Web Site」選單 → 選取「Add New Item」，選取 Web Form，清除「Place code in separate file」核取方塊，將檔案命名為「UseGridView.aspx」。
4. 從「Server Explorer」將 MyDb 連線下的 Employees 資料表拖曳到「UseGridView.aspx」設計畫面。



5. 點選畫面上的 GridView 控制項之智慧型標籤 → 選「Auto Format」選單進行格式化。
6. 在「Solution Explorer」→ 點選「UseGridView.aspx」網頁 → 按滑鼠右鍵 → 選「View In Browser」。

EmployeeID	Name	Age
1	Rose	25
2	Mary	30
3	Bet	32

7. 結束程式執行，關閉瀏覽器。
8. 檢視工具產生的 GridView 控制項程式碼，請注意 DataSourceID 設定為 SqlDataSource1。

```
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False"
BackColor="LightGoldenrodYellow"
BorderColor="Tan" BorderWidth="1px"
```

```

CellPadding="2" DataKeyNames="EmployeeID"
DataSourceID="SqlDataSource1"
EmptyDataText="There are no data records to display."
ForeColor="Black" GridLines="None">
<FooterStyle BackColor="Tan" />
<Columns>
    <asp:BoundField DataField="EmployeeID"
        HeaderText="EmployeeID" ReadOnly="True"
        SortExpression="EmployeeID" />
    <asp:BoundField DataField="Name"
        HeaderText="Name"
        SortExpression="Name" />
    <asp:BoundField DataField="Age"
        HeaderText="Age" SortExpression="Age" />
</Columns>
<SelectedRowStyle BackColor="DarkSlateBlue"
    ForeColor="GhostWhite" />
<PagerStyle BackColor="PaleGoldenrod"
    ForeColor="DarkSlateBlue"
    HorizontalAlign="Center" />
<HeaderStyle BackColor="Tan" Font-Bold="True" />
<AlternatingRowStyle BackColor="PaleGoldenrod" />
</asp:GridView>

```

9. 檢視工具產生的 SqlDataSource 程式碼，請注意
ConnectionString 與 ProviderName 的設定。

```

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString
="<%$ ConnectionStrings:MyDBConnectionString1 %>

DeleteCommand="DELETE FROM [Employees] WHERE
[EmployeeID] = @EmployeeID"
InsertCommand="INSERT INTO [Employees] ([EmployeeID],
[Name], [Age]) VALUES (@EmployeeID, @Name, @Age)"

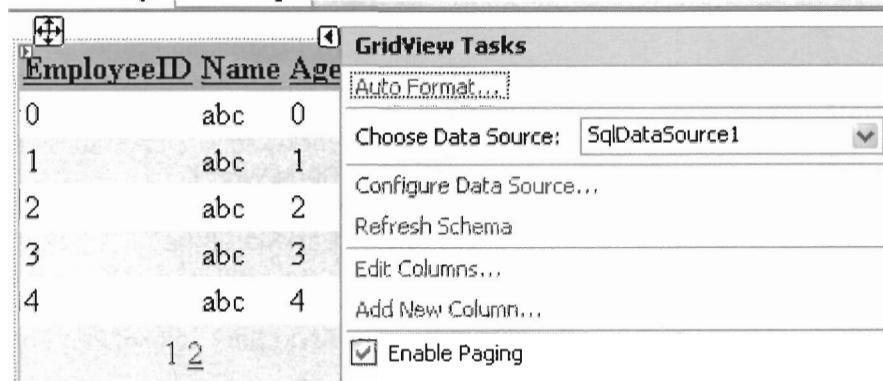
ProviderName="<%$ ConnectionStrings:MyDBConnectionString
1.ProviderName %>"
SelectCommand="SELECT [EmployeeID], [Name], [Age]
FROM [Employees]"

UpdateCommand="UPDATE [Employees]
SET [Name] = @Name, [Age] = @Age WHERE [EmployeeID]
= @EmployeeID">
<InsertParameters>
    <asp:Parameter Name="EmployeeID" Type="Int32" />
    <asp:Parameter Name="Name" Type="String" />
    <asp:Parameter Name="Age" Type="Int32" />
</InsertParameters>
<UpdateParameters>
    <asp:Parameter Name="Name" Type="String" />
    <asp:Parameter Name="Age" Type="Int32" />
    <asp:Parameter Name="EmployeeID" Type="Int32" />
</UpdateParameters>

```

```
<DeleteParameters>
  <asp:Parameter Name="EmployeeID" Type="Int32" />
</DeleteParameters>
</asp:SqlDataSource>
```

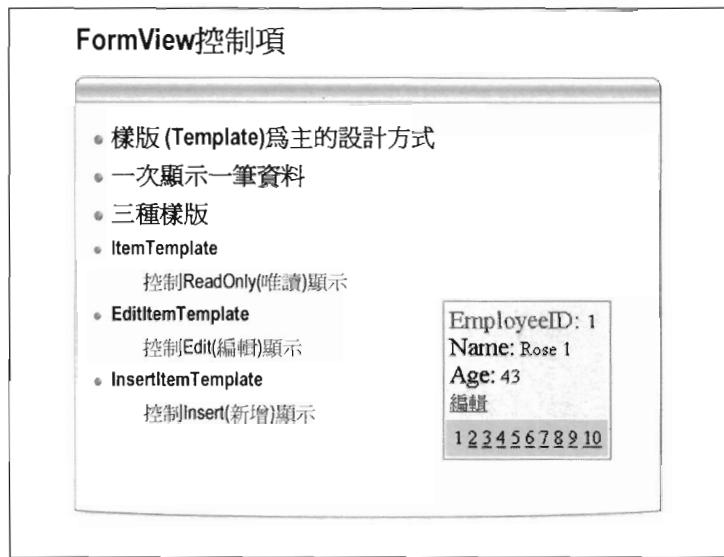
10. 啓用分頁能力。按一下 GridView 控制項的智慧型標籤，點選「Enable Paging」。再按一下 GridView 控制項的智慧型標籤關閉選單。



11. 測試分頁效果。在「Solution Explorer」→點選「UseGridView.aspx」網頁→按滑鼠右鍵→選「View In Browser」。

預設 GridView 控制項的資料一頁顯示 10 筆，請確認您的資料超過 10 筆，或是修改 GridView 控制項的 PageSize 屬性，變更一頁顯示的資料筆數。

12. 結束程式執行，關閉瀏覽器。
13. 啓用排序功能。按一下 GridView 控制項的智慧型標籤，點選「Enable Sorting」。再按一下 GridView 控制項的智慧型標籤關閉選單。
14. 測試排序效果。在「Solution Explorer」→點選「UseGridView.aspx」網頁→按滑鼠右鍵→選「View In Browser」。您將發現，每個欄位標題變成超連結，按下欄位標題，則資料將依此欄位進行排序～



FormView 控制項

FormView 是一種使用樣版 (Template)來描述資料的展現樣式的控制項。FormView 控制項一次從資料來源取出一筆資料，可以選擇性地顯示分頁按鈕，以用來瀏覽資料。

FormView 控制項支援三種顯示：

- ReadOnly：唯讀。
- Edit：編輯。
- Insert：新增。

FormView 控制項支援的主要樣版如下：

- ItemTemplate：定義檢視模式 (View Mode) 資料展示樣式。
- EditItemTemplate：定義編輯資料時的展示樣式。
- InsertItemTemplate：定義新增資料時的展示樣式。

設計 FormView 的方式也很簡單，僅需要先設定好資料來源控制項，並且將資料來源控制項的 ID 指定給 FormView 的 DataSourceID 屬性，最後再決定要啓用 FormView 控制項的何種功能即可。

練習 8.3 :使用 FormView 控制項

•了解如何使用 FormView 與 SqlDataSource 控制項來建立資料存取的網頁。

•預估實作時間：15分鐘



練習 8.3 :使用 FormView 控制項

目的：

這個練習主要是讓您熟悉了解如何使用 FormView 與 SqlDataSource 控制項來建立資料存取的網頁

功能描述：

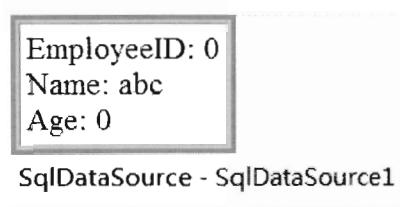
在這個練習中，將延續上一個練習，加入一個 FormView 控制項，把員工資料顯示 FormView 控制項上。

預估實作時間：15 分鐘

實作步驟：

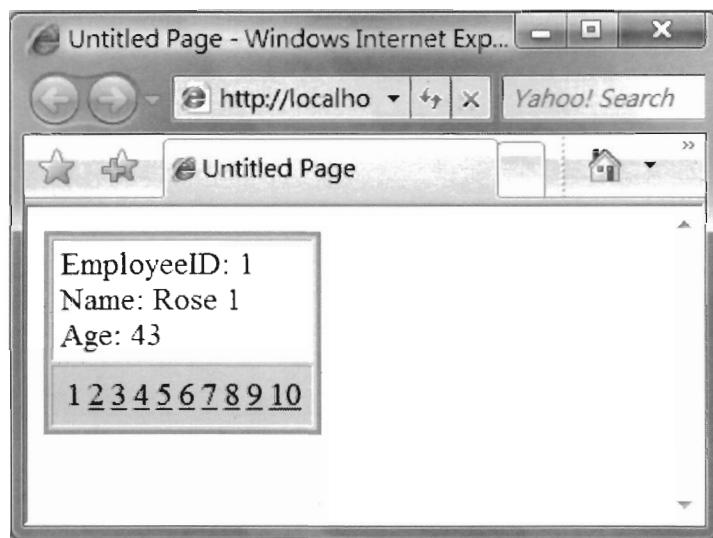
1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。

2. 從「File」→「Open」→「Web Site」→選取「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod08_3\Starter\Mod08_3」目錄，完成後，按下「Open」按鈕即可開啟網站。
3. 建立一個新網頁。從 Visual Studio 2008 開發工具「Web Site」選單 → 選取「Add New Item」，選取 Web Form，清除「Place code in separate file」核取方塊，將檔案命名為「ShowEmployee.aspx」。
4. 從「ToolBox」→「Data」頁籤 → 拖曳 FormView 控制項到「ShowEmployee.aspx」網頁內。
5. 按一下 FormView 控制項的智慧型標籤，點選 Choose Data Source 右方的下拉式清單方塊，選「New Data Source」。
6. 選「Database」，按「OK」。
7. 選取「MyDbConnectionString1」，按「Next」。
8. 選取「Specify columns from a table or view」，勾選「EmployeeID」、「Name」、「Age」欄位，按「Next」。
9. 按「Finish」完成精靈的設定。
10. 點選畫面上的 FormView 控制項之智慧型標籤，選「Auto Format」選單進行格式化。



11. 在「Solution Explorer」→點選「ShowEmployee.aspx」網頁按滑鼠右鍵 → 選「View In Browser」。
12. 預設將顯示第一筆員工資料在網頁上。
13. 結束程式執行，關閉瀏覽器。

14. 按一下 FormView 控制項的智慧型標籤，點選「Enable Paging」。
15. 在「Solution Explorer」→點選「ShowEmployee.aspx」網頁按滑鼠右鍵→選「View In Browser」。你將可以透過分頁超連結瀏覽每一個員工的資料。畫面應如下所示：



16. 結束程式執行，關閉瀏覽器。

使用 DetailsView 控制項

- 似 GridView，欄位為主的設計方式
- 一次顯示一筆資料
- 具備資料分頁、資料新增、刪除、修改能力
- 支援自訂欄位

EmployeeID 1 Name Rose 1 Age 43 照片	
New 1 2 3 4 5 6 7 8 9 10	

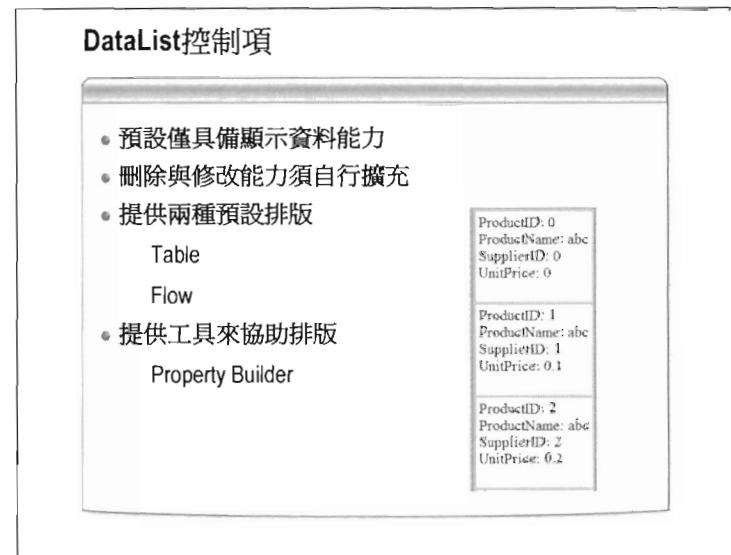
DetailsView 控制項

DetailsView 控制項主要是用來一次顯示一筆資料。外觀與 FormView 控制項非常相似，最大的不同在於 DetailsView 控制項並不是以樣板來控制外觀，DetailsView 控制項在設計上比較相似於 GridView，是以欄位的方式來設計。

DetailsView 控制項也可以繫結到任何資料來源控制項，進行資料操作，包含了資料分頁、資料新增、刪除、修改能力。

DetailsView 控制項也可建立自訂欄位來客製化外觀，可加入的自訂欄位包含：CheckBoxField、HyperLinkField、ImageField、ButtonField、CommandField 與 TemplateField 等等。

設計 DetailsView 的方式也很簡單，僅需要先設定好資料來源控制項，並且將資料來源控制項的 ID 指定給 DetailsView 的 DataSourceID 屬性，最後再決定要啓用 DetailsView 控制項的何種功能即可。



DataList 控制項

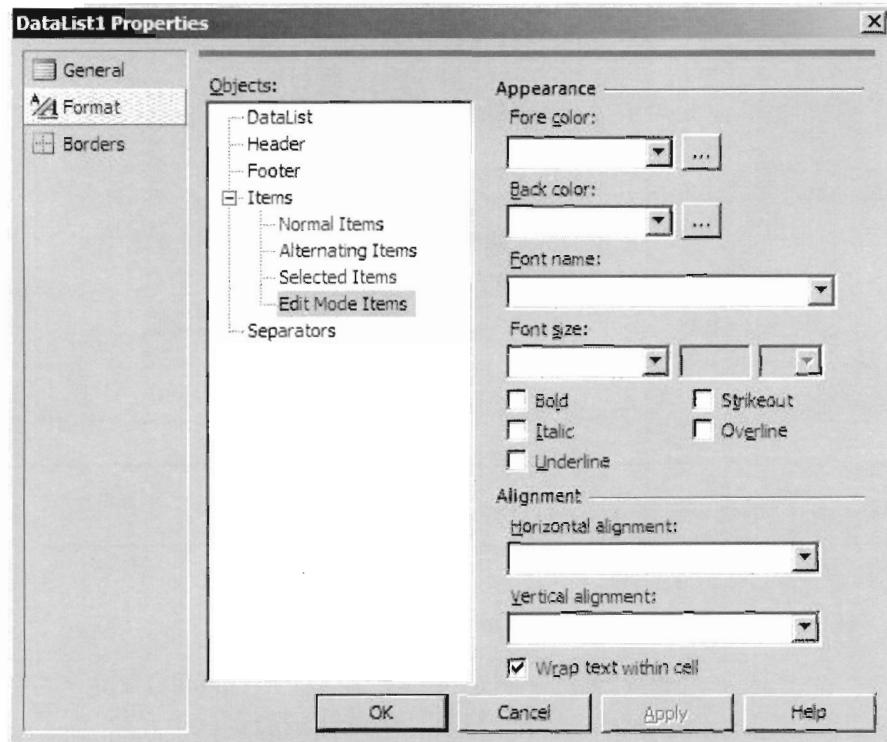
DataList 是一種是一種使用樣版 (Template)來描述資料的展現樣式的控制項，與 FormView 控制項不同的是，DataList 的自動化功能「陽春」很多，通常在指定完資料來源控制項的 ID 之後，就能完成繫結控制項的基本樣式設定，如 FormView，開發工具就會自動協助完成基本的外觀，如需要客製化在自行針對樣版 (Template)來做調整即可，但是 DataList 在指定完之後，預設僅會完成顯示資料樣式的外觀，如果需要讓該控制項支援修改或刪除能力，則需要自行針對樣版從頭開發。

另外，需注意的是，DataList 的修改或刪除功能，除了外觀要自訂之外，預設並不會利用到資料來源控制項的修改或刪除功能，也就是這一段程式碼亦需要自行實作。

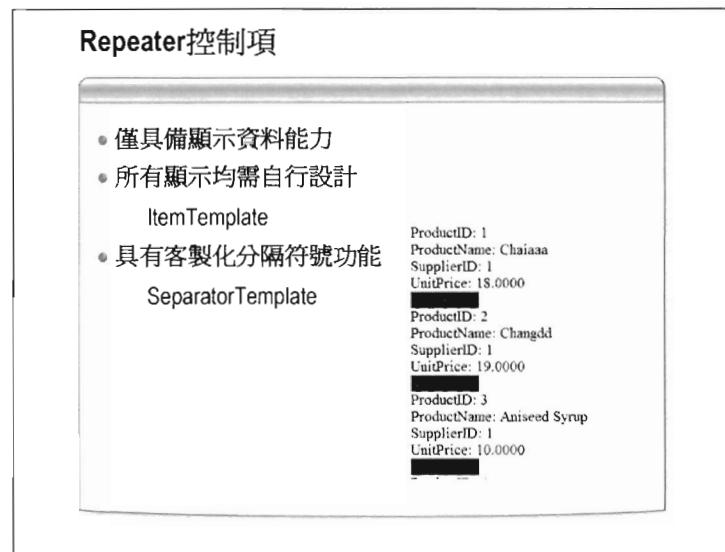
DataList 控制項支援的主要樣版如下：

- ItemTemplate：定義資料顯示時的樣式。
- EditItemTemplate：定義編輯資料時的展示樣式。

相較於其他控制項的樣版 (Template)都需要自行利用開發工具或直接針對控制項標籤來設計，DataList 控制項提供了一個名為「Property Builder」的工具來協助編輯各樣版的外觀與排版，如下所示：



而主要的排版方式有兩種「Table」與「Flow」，使用「Table」排版輸出，產生至使用者端的結果將會是以 Html 表格的方式呈現，而「Flow」排版則是僅有資料與斷行標籤，一般來說使用「Table」排版能夠呈現較為美觀的結果。排版方式可在 DataList 控制項的「RepeatLayout」屬性指定。



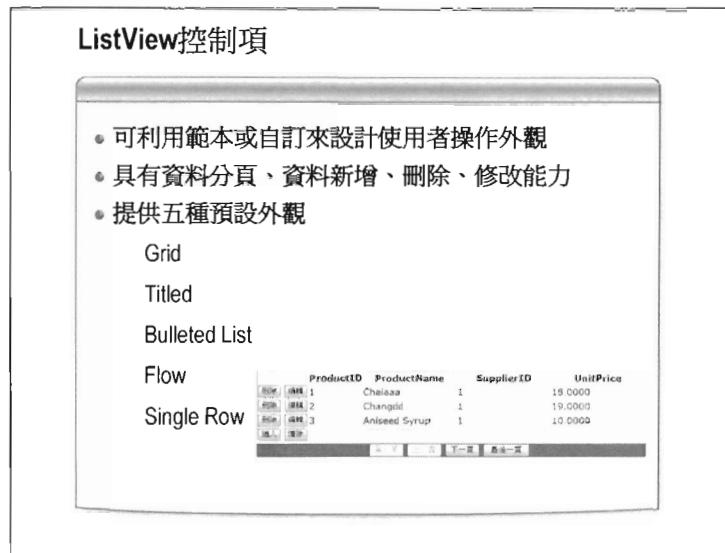
Repeater 控制項

Repeater 控制項也是一種使用樣版 (Template)來描述資料的展現樣式的控制項，特別的地方在於 Repeater 控制項的設計必須「全部自己來」，而且該控制項僅具有顯示資料用的樣板(Template)。

Repeater 控制項在指定資料來源控制項的 ID 之後，開發工具並不會自動協助完成基本的外觀，而 Design 頁面的介面也不會提供任何工具視窗來輔助編輯，使用者必須自行切換至 Source 頁面來自行編輯繫結語法與設計排版。

Repeater 控制項支援的主要樣版如下：

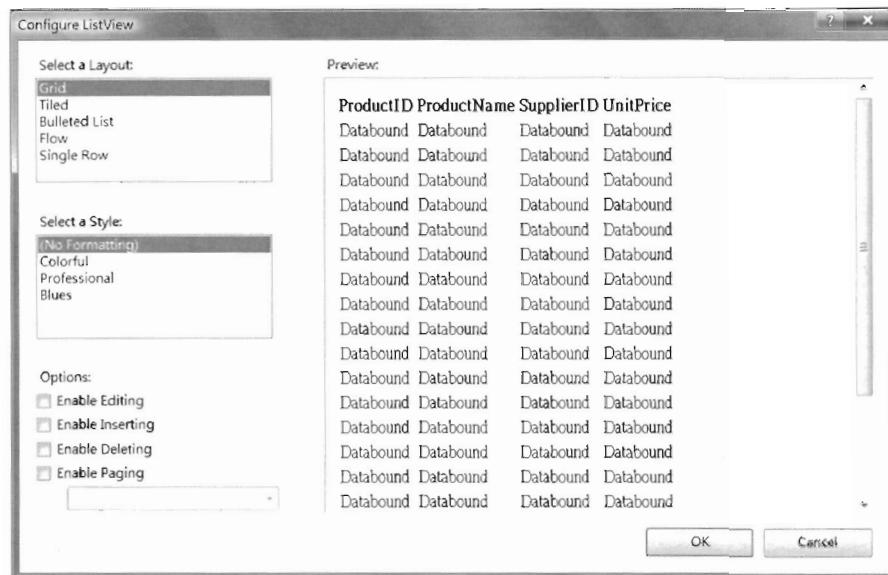
- ItemTemplate：定義資料顯示時的樣式。
- SeparatorTemplate：定義每一筆資料之間的分隔符號展示樣式。



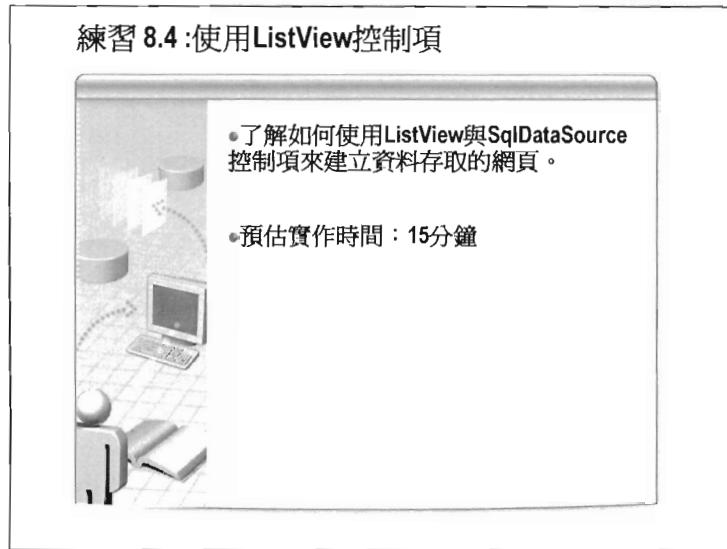
ListView 控制項

ListView 控制項感覺上像是許多控制項的大合體，能夠像 GridView 控制項一樣用來展示表格式的資料，也像 DataList 展現清單式的資料。除此之外，它還提供許多特殊的功能，是其它控制項所沒有的。最重要的是，ListView 控制項幾乎可以完全取代 ASP.NET 中的其他所有資料繫結控制項包含 GridView、DataList、Repeater、FormView、DetailsView 等等。和這些控制項比起來，ListView 控制項的資料繫結工作可以更簡化，也可以運用 CSS 樣式來設計外觀，可以搭配 DataPager 控制項進行分頁效果。另外，排序、插入、刪除及更新等功能當然也不缺席。

ListView 控制項在設計上和一般的資料繫結控制項相同，都可利用 DataSourceID 屬性來設定資料來源控制項；較為不同的是，在設定完來源控制項後，該控制項並沒有預設外觀，所以還必須進一步的設定樣版 (Template)來決定外觀，如果不甚熟悉樣版 (Template)的設計方式，開發工具也提供了幾種的預設樣版供使用者選擇，可以在智慧型標籤中點選「Configure ListView」來開啟該設定工具，工具畫面如下：



使用者可以在「Select a Layout」區塊中選擇欲套用的樣版 (Template)，而預設的樣版有 Grid、Titled、Bulleted List、Flow 與 Single Row 等五種；而「Select a Style」則可以選擇欲套用的風格；最下方的「Options」則是可選擇希望啓用的功能。



練習 8.4 :使用 ListView 控制項

目的：

這個練習主要是讓您熟悉了解如何使用 ListView 與 SqlDataSource 控制項來建立資料存取的網頁。

功能描述：

這個練習會延續之前的練習，新增一個網頁，並且加入一個 ListView 與 SqlDataSource 控制項，把員工資料顯示 ListView 控制項上，並讓 ListView 具備新增、查詢與修改能力。

預估實作時間：15分鐘

實作步驟：

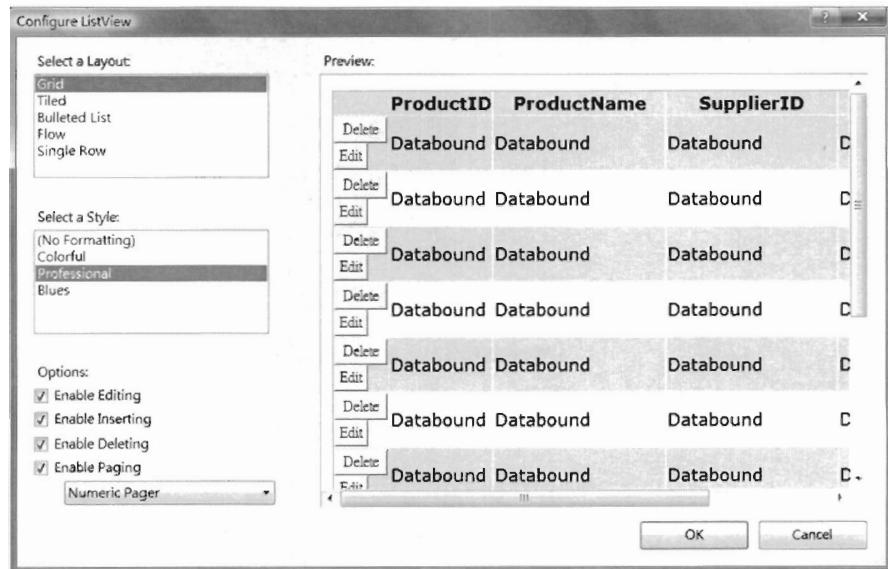
1. 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。

2. 從「File」→「Open」→「Web Site」→選取「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod08_4\Starter\Mod08_4」目錄，完成後，按下「Open」按鈕即可開啓網站。
3. 建立一個新網頁。從 Visual Studio 2008 開發工具「Web Site」選單 → 選取「Add New Item」，選取 Web Form，清除「Place code in separate file」核取方塊，將檔案命名為「UseListView.aspx」。
4. 從「ToolBox」→「Data」頁籤 → 拖曳 ListView 控制項到「UseListView.aspx」網頁內。
5. 按一下 ListView 控制項的智慧型標籤，點選 Choose Data Source 右方的下拉式清單方塊，選「New Data Source」。
6. 選「Database」，按「OK」。
7. 選取「MyDbConnectionString1」，按「Next」。
8. 選取「Specify columns from a table or view」，勾選「EmployeeID」、「Name」、「Age」欄位。
9. 點選「Advanced」按鈕，勾選「Generate INSERT,UPDATE, and DELETE statements」，按「OK」。
10. 按「Next」候再按「Finish」完成精靈的設定。完成後，畫面應如下所示：



11. 點選畫面上的 ListView 控制項之智慧型標籤，選「Configure ListView」選單進行設定。
12. 「Select a Layout」區塊中選擇「Grid」；「Select a Style」則選擇「Professional」；下方的「Option」則將所有的功能啓

用，將分頁功能選擇為「Numeric Pager」，完成後畫面應如下所示：



13. 設計完成後，畫面應如下所示：

	ProductID	ProductName	SupplierID	UnitPrice				
Delete Edit	0	abc	0	0				
Delete Edit	1	abc	1	0.1				
Delete Edit	2	abc	2	0.2				
Delete Edit	3	abc	3	0.3				
Delete Edit	4	abc	4	0.4				
Delete Edit	5	abc	5	0.5				
Delete Edit	6	abc	6	0.6				
Delete Edit	7	abc	7	0.7				
Delete Edit	8	abc	8	0.8				
Delete Edit	9	abc	9	0.9				
Insert Clear								
	First	1	2	3	4	5	...	Last

SqlDataSource - SqlDataSource1

14. 在「Solution Explorer」→點選「ListView.aspx」網頁 按滑鼠右鍵→選「View In Browser」。你將可以在該網頁上進行資料表的新增、修改、刪除、分頁與排序功能。畫面應如下所示：

The screenshot shows a Microsoft Internet Explorer window displaying a list of employee records. The page title is "http://localhost:56628/Mod08_4/UseListView.aspx". The table has columns for EmployeeID, Name, and Age. Each row contains a "Delete" button and an "Edit" button. At the bottom left are "Insert" and "Clear" buttons. At the bottom right are navigation buttons for "First", "Last", and "Page 1".

	EmployeeID	Name	Age
<input type="button" value="Delete"/>	<input type="button" value="Edit"/> 1	Rose 1	43
<input type="button" value="Delete"/>	<input type="button" value="Edit"/> 2	Mary	30
<input type="button" value="Delete"/>	<input type="button" value="Edit"/> 3	Bet	32
<input type="button" value="Delete"/>	<input type="button" value="Edit"/> 4	Candy	23
<input type="button" value="Delete"/>	<input type="button" value="Edit"/> 5	Jessie	29
<input type="button" value="Delete"/>	<input type="button" value="Edit"/> 6	Lilian	23
<input type="button" value="Delete"/>	<input type="button" value="Edit"/> 7	Jack	23
<input type="button" value="Delete"/>	<input type="button" value="Edit"/> 8	Apple	34
<input type="button" value="Delete"/>	<input type="button" value="Edit"/> 9	Landy	43
<input type="button" value="Delete"/>	<input type="button" value="Edit"/> 11	ee	22
<input type="button" value="Insert"/>	<input type="button" value="Clear"/>		

其他相關控制項

- **清單系列控制項**

亦可繫結資料來源控制項

僅提供顯示資料能力

針對Text與Value兩欄位繫結即可

- **DataPager控制項**

提供分頁功能

僅針對實作IPageableItemContainer 介面的資料繫
結控制項

預設僅支援ListView控制項

其他相關控制項

除了功能強大的資料繫結控制項之外，在利用資料來源控制項開發網
頁時，其實還有其他控制項能夠提供相關的操作，以下就分別介紹。

清單系列控制項

清單系列控制項泛指具有提供 ListItem 型別集合屬性的控制項，如
RadioButtonList、CheckBoxList、ListBox、DropDownList 與
BulletList 等。這些控制項的特性都是能夠以各種不同的清單方式來
提供檢視或選擇，而該系列控制項在設計上，也能夠支援資料來源控
制項的繫結；利用該設計方式，就能夠將不同外部資料來源的資料輕
易的顯示至清單系列控制項上而不需要撰寫程式碼。

在使用上必須要留心以下幾點：

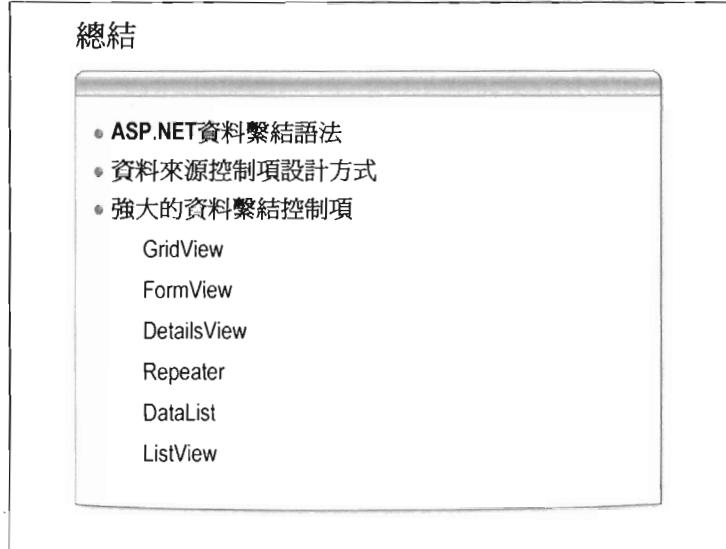
- 由於清單系列控制項有別於資料繫結控制項，僅具備有資料
顯示的能力，所以在設定資料來源控制項時，並不需要額外
設定資料的新增、修改與刪除部分。

- 雖然清單系列控制項能夠輕易的顯示由資料來源控制項所提供的資料，但是由於清單系列控制項僅具備有 Text 與 Value 兩個屬性，所以在使用資料來源控制項在做資料查詢時，也不需要繫結過多的欄位，簡單的來說，兩個欄位即可。

DataPager

DataPager 控制項是一種用來輔助用的控制項，任何有實作 IPageableItemContainer 介面的資料繫結控制項 (如 ListView 控制項)，都能夠利用它來提供資料的分頁功能。

該控制項的使用方式也很簡單，利用 PagedControlID 屬性來指定欲分頁的控制項 ID，再利用 PageSize 屬性決定一頁要分幾筆資料之後，最後只要決定頁面巡覽樣式為頁碼式或上下頁式就完成了。



第九章: AJAX 介紹與運用

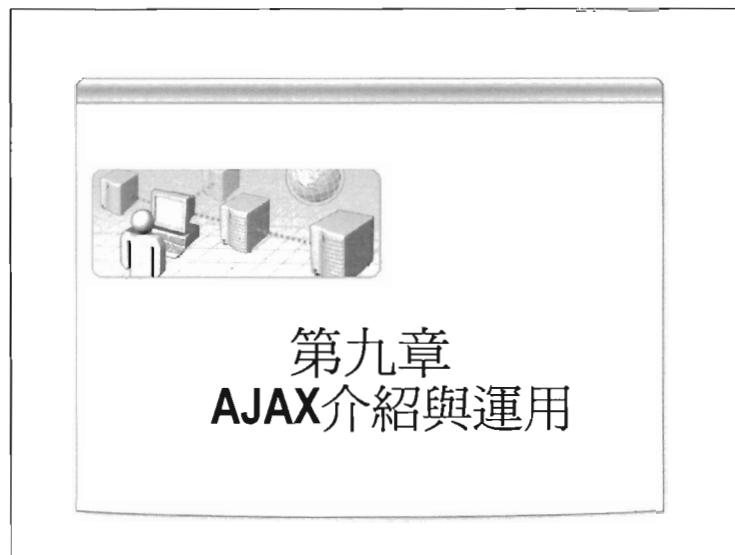
本章大綱

課程大綱.....	3
什麼是 AJAX ?	4
ASP.NET AJAX 架構.....	6
使用 ASP.NET AJAX 啟服器控制項.....	9
ScriptManager 控制項.....	10
了解網頁部分更新	11
練習 9.1 : 使用 UpdatePanel 控制項設計非同步更新 ...	13
使用 Timer 控制項的定期執行程式碼.....	17
練習 9.2 : 使用 Timer 控制項的定期執行程式碼	18
使用 UpdateProgress 控制項提供進度回饋	22
練習 9.3 : 使用 UpdateProgress 控制項提供進度回饋 ...	23
ASP.NET AJAX Control Toolkit 簡介	27
使用 ASP.NET AJAX Control Toolkit.....	28
練習 9.4 : 使用 AJAX Control Toolkit 中的 CalendarExtender 控制項	30

作者 :

許薰宇





課程大綱

- 什麼是AJAX？
- ASP.NET AJAX架構
- 使用ASP.NET AJAX伺服器控制項
- 了解網頁部分更新
- ASP.NET AJAX Control Toolkit 簡介

課程大綱

在這個章節中將介紹 Asynchronous JavaScript and XML (Ajax) 技術，讓你建立回應效率更佳的網站應用程式。Microsoft® ASP.NET AJAX 整合 AJAX 特性，提供諸如部分更新功能，和其他伺服器控制項，能幫助使用者快速開發 AJAX 網頁。

本章介紹以下主題：

- 什麼是 AJAX？
- ASP.NET AJAX 架構
- 使用 ASP.NET AJAX 伺服器控制項
- 了解網頁部分更新
- ASP.NET AJAX Control Toolkit 簡介

什麼是AJAX？

- **Asynchronous JavaScript And XML**
 - Asynchronous—非同步呼叫
 - JavaScript—大量使用 Client Script
 - XML—結合 XML 方式傳輸資料
- 連結瀏覽器與伺服器的技術
- 為何需要AJAX?
 - Browser 和 Web Server 之間是兩個世界
 - AJAX提供兩端的通訊能力
 - AJAX使用非同步技術

什麼是 AJAX？

AJAX 為「Asynchronous JavaScript And XML」全名的縮寫，也就是使用大量的 JavaScript 與 XML 技術來達到非同步的網頁傳輸。而 AJAX 也並不是一項跨時代的全新技術，而是將目前網路上的相關標準與主流技術做了整合之後所演進的一項動態網頁更新技術。其主要目的為提高網頁的互動性與使用性，並以求達到如同 Windows 應用程式一般流暢的使用感為最高目標。

AJAX 其實在很早之前就已經在研發了，只是到了近年來的技術被相關網站運用得當，被大量的應用於 Google 的網頁之後，如 Gmail、Google Maps、和 Google Suggest 等等，而使得此技術因此聲名大噪。

而 AJAX 本身也並不是一門單獨的網路技術，其所用到的基本技術如下所示：

1. XHTML、DHTML、HTML，與 CSS 來做為其網頁的動態呈現。

2. 利用 DOM 與大量的 JavaScript 來處裡由伺服器所取回的資料 (XML)
3. 利用 XMLHttpRequest 物件，以非同步的模式來與遠端的伺服器溝通。

非同步技術

AJAX 所達成的非同步技術，能夠達成網頁上的以下功能：

1. 各自觸發更新處理程序，不需排隊輪流觸發。
2. 各自取得伺服器端所傳回的處理結果。
3. 將各自取回的結果各自更新回所對應到的控制項或是網頁區塊中。

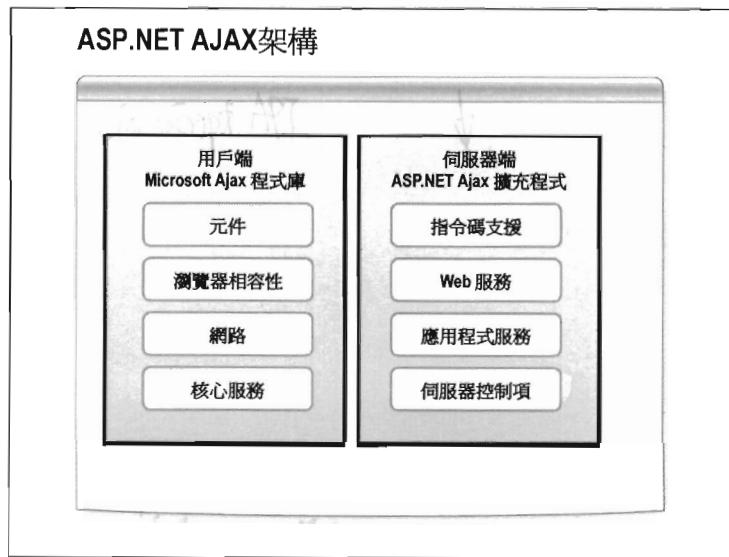
也由於它所提供的非同步技術的獨特性，讓 AJAX 有在現今的網路世界中，如此的炙手可熱。而上所述的幾項功能，也分別達到了如下的優點：

- 降低網路流量

而 AJAX 的局部更新功能，正好可以解決網路流量問題；透過局部的更新，來避免掉重新傳送整個網頁，如此可以大幅的改善使用者與伺服器端的網頁傳送流量，相對的可以提高處理效能與節省擴充相關設備的開銷。

- 即時反應與提升使用者觀感

AJAX 的局部更新功能，就可以達到即時反應的效果。並且讓使用者在使用該網頁之時，感覺該網頁不會在每次更新之後都重新刷新頁面，進而與使用 Windows 應用程式的使用觀感重疊。



ASP.NET AJAX 架構

在 ASP.NET 3.5 推出後一年左右，微軟將 ASP.NET 與 AJAX 緊密整合，稱之為 ASP.NET AJAX。程式設計師可以使用 Microsoft Visual Studio 直接開發 AJAX 網頁，不需再安裝其它套件。

ASP.NET AJAX 使用戶端指令碼程式庫 (Client-Script Library) 和許多伺服器控制項，結合 JavaScript、HTML、DHTML 技術。讓您整合 ASP.NET 伺服器架構開發平台，使用回應速度更佳的使用者介面項目，快速建立包含豐富使用者經驗的 Web 網頁。

由於此為非同步更新的技術，所以不僅僅是伺服端要能夠支援，瀏覽器端也有其非同步技術的支援限制，以微軟的 Internet Explorer 來說，如要能夠瀏覽具有 AJAX 的網站，就必須是要版本 5.01 之後的才能夠支援，且網頁的安全設定方面，需能執行 JavaScript 指令碼。

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：

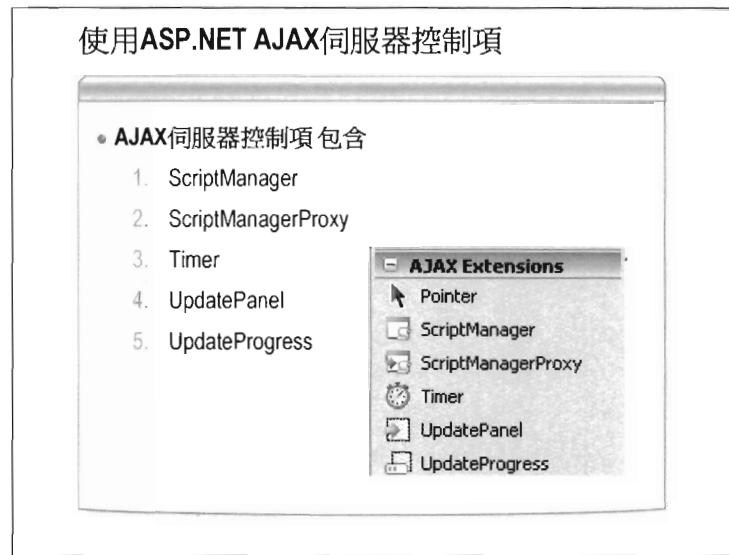
- 指令碼支援：ASP.NET 會由伺服端送出指令碼到瀏覽器，以便建立啓用 AJAX 的網頁，以支援 JavaScript 擴充功能，如物件導向、介面...等，以及網頁局部生成(Partial- Page Rendering) 功能。ASP.NET 指令碼包含發行版和偵錯版。
- Web 服務：您可以使用 JavaScript 來呼叫 ASP.NET Web 服務 (.asmx) 或者是 Windows Communication Foundation (WCF) 服務 (.svc)，以設計服務導向應用程式。

- 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。

使用 AJAX 擴充程式設計網頁

AJAX 擴充程式 (AJAX Extension) 包含許多伺服端的核心元件，可用來擴充 ASP.NET 伺服器控制項的功能。這些功能都是實作在 System.Web.Configuration 與 System.Web.UI 組件之中，常用的命名空間 (Namespace) 包含：

- System.Web.Configuration 命名空間：提供 ASP.NET 組態設定相關的類別。
- System.Web.Handlers 命名空間：提供 HTTP Handler 類別處理 HTTP Request。
- System.Web.Script.Serialization 命名空間：包含 JavaScript Object Notation (JSON) 序列化和還原序列化能力，或允許自訂序列化。
- System.Web.Script.Services 命名空間：提供客製化 Web 服務功能。
- System.Web.UI 命名空間：提供設計 ASP.NET Web 網頁使用者介面。
- System.Web.UI.Design 命名空間：擴充 ASP.NET Web 網頁和 Web 伺服器控制項的開發工具設計階段支援。



使用 ASP.NET AJAX 伺服器控制項

ASP.NET 提供的 AJAX 伺服器控制項包含如下：

- ScriptManager：管理用戶端指令碼，提供網頁部分更新 (Partial Update)、當地語系化、全球化..等功能。
- UpdatePanel：可提供網頁部分更新功能。
- UpdateProgress：提供有關 UpdatePanel 控制項中網頁網頁部分更新的進度資訊。
- Timer：以定義的間隔和伺服端溝通。可以搭配 UpdatePanel 控制項一起使用，以定期執行網頁部分更新動作。

當您在 Visual Studio 開啓 ASP.NET 網站時，從 Toolbox 就可以看到這些控制項。



ScriptManager控制項

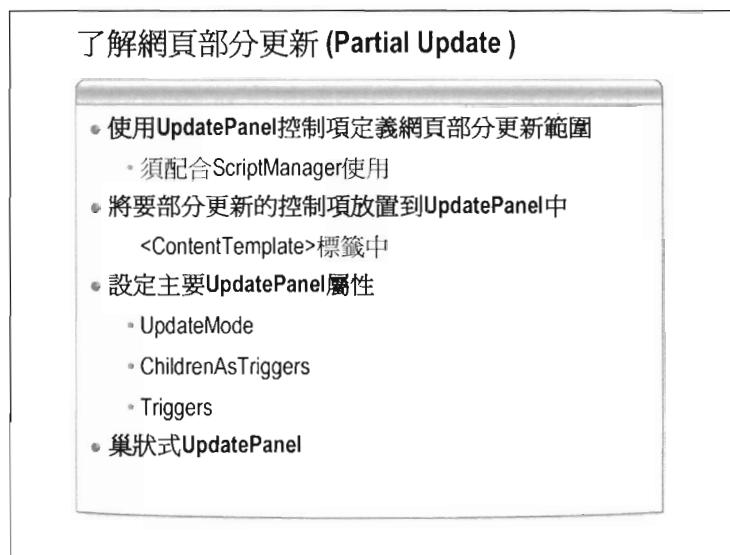
- **ScriptManager控制項主要功能**
 - 1. 管理ASP.NET AJAX 資源。
 - 2. 下載Microsoft AJAX Library 指令碼到瀏覽器。
 - 3. 整合UpdatePanel 控制項以完成網頁局部更新動作。
 - 4. 註冊指令碼。
 - 5. 指定要送到瀏覽器的指令碼為發行版或偵錯版。
 - 6. 註冊要使用的Web 服務。
 - 7. 註冊要使用的ASP.NET 應用程式服務，如驗證、角色、設定檔。
 - 8. 設定具備文化特性能力的指令碼。

ScriptManager 控制項

ScriptManager 控制項主要是應用在管理用戶端指令碼，提供網頁部分更新 (Partial Update)、當地語系化、全球化..等功能。此控制項是定義在 System.Web.Extensions.dll 組件中，它是 ASP.NET AJAX 的核心功能。

ScriptManager 控制項主要的工作包含：

- 管理網頁上的所有 ASP.NET AJAX 資源。
- 下載 Microsoft AJAX Library 指令碼到瀏覽器。
- 整合 UpdatePanel 控制項以完成網頁局部更新動作。
- 註冊指令碼。
- 指定要送到瀏覽器的指令碼為發行版或偵錯版。
- 註冊要使用的 Web 服務。
- 註冊要使用的 ASP.NET 應用程式服務，如驗證、角色、設定檔。
- 設定具備文化特性能力的指令碼。



了解網頁部分更新

網頁部分更新機制能夠在 Postback 後不重新整理整個網頁，而是只更新網頁中部分的區域。這樣的好處是使用者與網頁的互動性會更佳。您可以使用使用 UpdatePanel 控制項定義網頁部分更新範圍，而不需要撰寫用戶端指令碼。

使用 UpdatePanel 控制項

UpdatePanel 控制項，須配合 ScriptManager 使用才能夠進行網頁部分更新動作。UpdatePanel 控制項的內容可以是 HTML 或其他 ASP.NET 伺服器控制項。這些控制項必需出現在 UpdatePanel 控制項的 <ContentTemplate> 標籤中。

若 UpdatePanel 並非存在於另一個 UpdatePanel 之中，則 UpdatePanel 是根據 UpdateMode 與 ChildrenAsTriggers 屬性與 Trigger 的設定來決定是否進行部分更新的動作。

UpdateMode 可以設定為：

- Always：網頁只要 postback，則 UpdatePanel 控制項的內容便會更新，包含非同步 postback。
- Conditional：根據不同的條件，如 Trigger 來決定是否進行更新。

ChildrenAsTriggers 可以設定為 true，或 false。若設為 true 則 UpdatePanel 中的控制項只要有進行 postback 動作，都會觸發 UpdatePanel 部分更新。若設為 false，則不觸發部分更新，除非使用 Trigger 或程式碼要求更新。

UpdatePanel 的 UpdateMode 屬性，預設值為 Always，若 UpdateMode 屬性設定為 Always，則 UpdatePanel 中的控制項進行 postback 動作時，會觸發 UpdatePanel 的更新動作。而 UpdatePanel 外的控制項進行 postback 動作時，也會觸發 UpdatePanel 的更新動作。

若 UpdatePanel 之中還有 UpdatePanel，成巢狀方式出現，則父 UpdatePanel 進行部分更新時，也會更新子 UpdatePanel 中的內容。若 UpdatePanel 之中又包含 UpdatePanel，子 UpdatePanel 的 UpdateMode 若為 Conditional，則父 UpdatePanel 會視以下情況更新：

- 明確呼叫 Update 方法。
- 使用 Trigger 觸發。



練習 9.1 : 使用 UpdatePanel 控制項設計非同步更新

目的：

這個練習主要是讓您熟悉 UpdatePanel 控制項的使用，透過它的 ChildernAsTrigger，以及 Trigger 屬性，來新增兩個 Trigger：AsyncPostBackTriggerTrigger 與 PostBackTriggerTrigger 來測試非同步更新的動作。

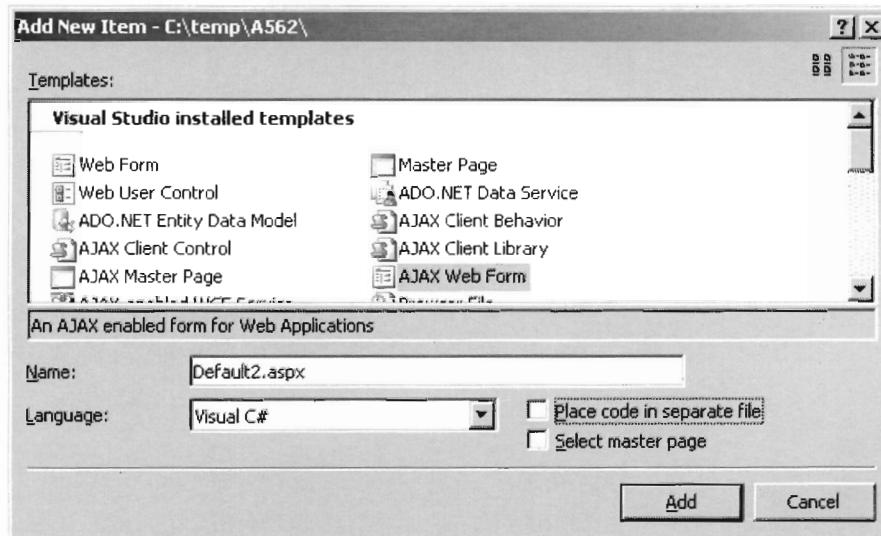
功能描述：

這個練習會在網頁中使用一個 UpdatePanel，UpdatePanel 外放一個 Button；UpdatePanel 內外放兩個 Button，當按鈕一與按鈕二被按下時，會進行非同步更新，只更新 UpdatePanel 中的 Label 上的時間。當按鈕三被按下時，會進行同步更新，更新 UpdatePanel 內與外的兩個 Label 上的時間。

預估實作時間：10 分鐘

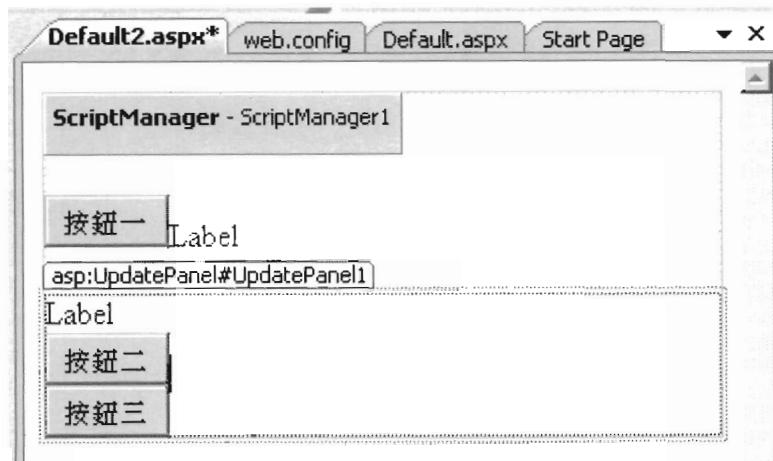
實作步驟：

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「\U9543\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「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。



預設網頁中會包含一個 ScriptManager 控制項。

4. 從「Toolbox」工具箱中拖拉一個 Button 控制項，一個 Label 控制項，一個 UpdatePanel 控制項到網頁中 ScriptManager 控制項下方。將 Button 控制項的 Text 屬性設定為「按鈕一」。
5. 在 UpdatePanel 控制項之中，加入兩個 Button 控制項，一個 Label 控制項。將 Button 控制項的 Text 屬性分別設定為「按鈕二」與「按鈕三」。您的畫面大致如下：

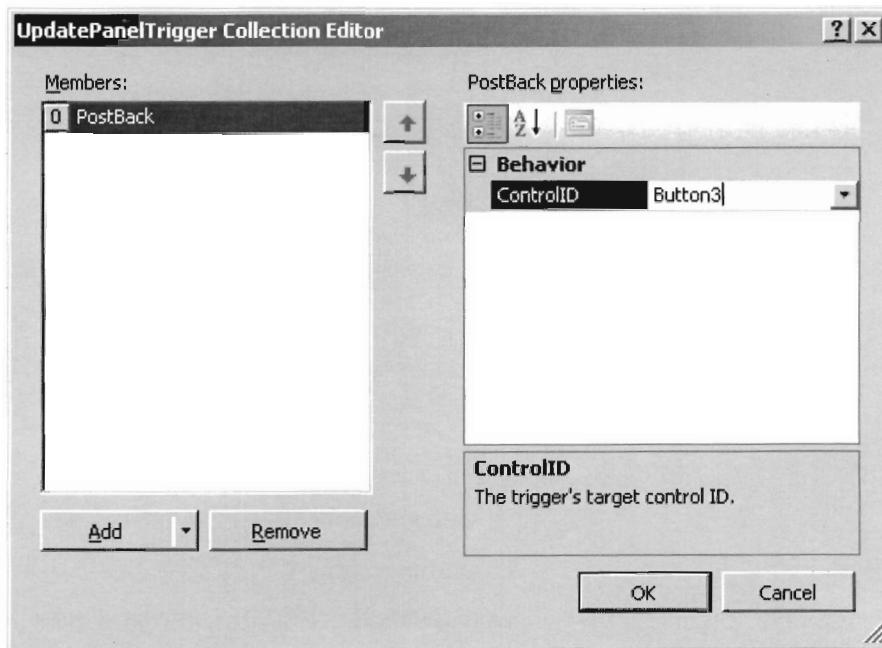


6. 設定 UpdatePanel 控制項的 UpdateMode 屬性。從「Properties」視窗上方的下拉式清單方塊，選取「UpdatePanel1」控制項，然後將 UpdateMode 屬性設為「Conditional」。另外，請注意預設 UpdatePanel 控制項的 ChildrenAsTrigger 屬性設定為 true。
7. 在 Page_Load 事件處理常式中，加入以下程式碼，將系統時間顯示在 Label1 與 Label2 控制項上：

```
Visual Basic
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Label1.Text = System.DateTime.Now.ToString()
    Label2.Text = System.DateTime.Now.ToString()
End Sub
```

```
C#
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = System.DateTime.Now.ToString();
    Label2.Text = System.DateTime.Now.ToString();
}
</script>
```

8. 設定 UpdatePanel 的 Triggers 屬性，新增一個PostBackTrigger，設定 ControlID 為 Button3，按鈕三會進行完整 PostBack。



9. 設定 UpdatePanel 的 Triggers 屬性，再新增一個
AsyncPostBackTrigger，設定 ControlID 為 Button1，按鈕一會
進行非同步 PostBack。
10. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠
右鍵→選「View In Browser」。當「按鈕一」與「按鈕二」
被按下時，會進行非同步更新，只更新 UpdatePanel 中的
Label 上的時間。當「按鈕三」被按下時，會進行同步更新，
更新 UpdatePanel 內與外的兩個 Label 上的時間。



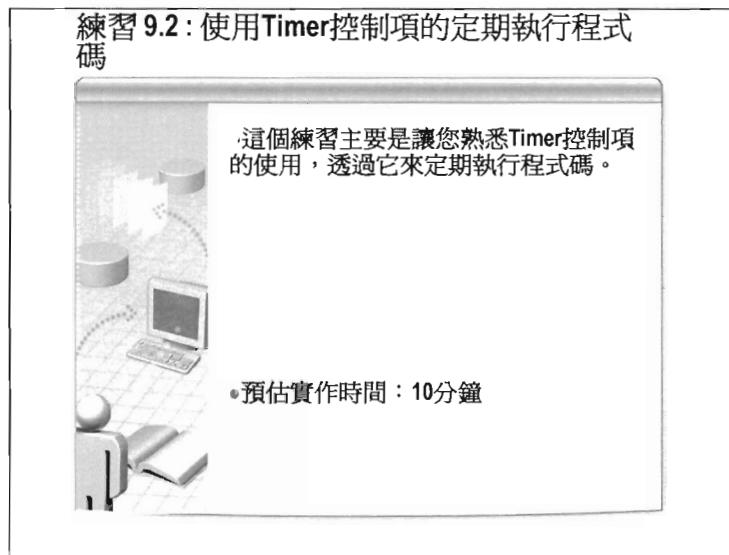
使用 Timer 控制項的定期執行程式碼

ASP.NET AJAX Extension 提供一個 Timer 控制項允許定期執行程式碼，您可以設定 Timer 控制項的 Interval 屬性來決定定期的時間間隔。Interval 屬性會以毫秒為單位，預設值為 60,000 毫秒 (60 秒)。另外有一個 Enabled 屬性，決定是否啓用 Timer。

若在 UpdatePanel 控制項之中使用 Timer 控制項，則 UpdatePanel 控制項就可以透過非同步更新機制來更新網頁部分的畫面。您也可以將 Timer 控制項置於 UpdatePanel 控制項之外，然後設定 Trigger 來進行非同步更新。Timer 控制項也可以不搭配 UpdatePanel 使用，在網頁中加入 Timer 控制項透過完整 PostBack 來定期執行程式碼。

注意事項，請勿將 Interval 屬性的值設太小，以免和 Web 伺服器溝通的動作過於頻繁，而降低效能。如非必要，勿過度使用 Timer 控制項來重新整理網頁。

您可以在 Tick 事件處理常式撰寫定期要執行的程式碼。



練習 9.2 : 使用 Timer 控制項的定期執行程式碼

目的：

這個練習主要是讓您熟悉 Timer 控制項的使用，透過它來定期執行程式碼。

功能描述：

這個練習會在網頁中使用一個 UpdatePanel、Timer 控制項，當網頁執行時，定期在伺服端產生 1~10 之間的亂數資料與系統時間，然後以非同步更新方式，將亂數資料與系統時間更新在 UpdatePanel 中 Label 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「\U9543\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_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. 在 UpdatePanel 控制項之中，加入兩個 Label 控制項，一個 Timer 控制項，從「Properties」視窗上方的下拉式清單方塊，選取「Label1」，然後將 Text 屬性設為空白。重複將 Label2 控制項的 Text 屬性設為空白。您的畫面大致如下。



UpdatePanel 的標籤目前看起來如下：

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
<ContentTemplate>
<asp:Timer ID="Timer1" runat="server" Interval="1000" >
</asp:Timer>
<br />
系統時間：
<asp:Label ID="Label1" runat="server"></asp:Label> <br />
亂數：
<asp:Label ID="Label2" runat="server"></asp:Label>
</ContentTemplate>
</asp:UpdatePanel>
```

6. 設定 Timer 控制項 Interval 屬性。從「Properties」視窗上方的下拉式清單方塊，選取「Timer1」，然後將 Interval 屬性設「1000」。

7. 點選「Properties」視窗上方的 Events 按鈕(閃電圖示)，雙擊 Tick 事件，產生 Tick 事件處理常式。在事件處理常式加入以下程式碼：

Visual Basic

```
Protected Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs)
    Label1.Text = DateTime.Now.ToString("yyyy/MM/dd HH:mm:ss")
    Label2.Text = GetData()
End Sub
```

C#

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    Label1.Text = DateTime.Now.ToString("yyyy/MM/dd HH:mm:ss");
    Label2.Text = GetData();
}
```

8. 在程式碼視窗加入以下函式：

Visual Basic

```
Private Function GetData() As String
    Dim r As Integer = New Random().Next(1, 11)
    Return r
End Function
```

C#

```
private string GetData()
{
    int r = new Random().Next(1, 11);
    return r.ToString();
}
```

9. 在 Page_Load 事件處理常式，加入以下程式：

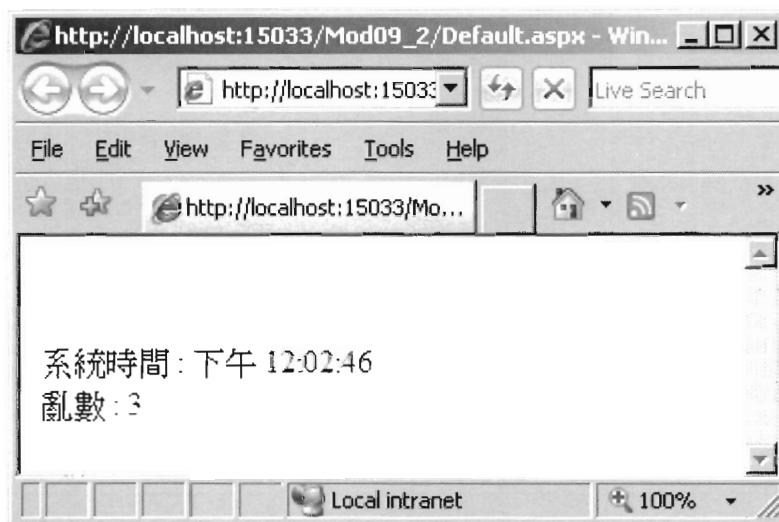
Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Label1.Text = DateTime.Now.ToString("yyyy/MM/dd HH:mm:ss")
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = DateTime.Now.ToString("yyyy/MM/dd HH:mm:ss");
}
```

10. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。每隔一秒鐘，就將系統時間取出，並產生亂數，更新到網頁畫面上，參考執行畫面如下。



使用UpdateProgress控制項提供進度回饋

- 部分更新網頁的過程中提供視覺化的進度回饋
- 一個網頁中，可包含多個
- 藉由AssociatedUpdatePanelID屬性和網頁中的UpdatePanel控制項關聯

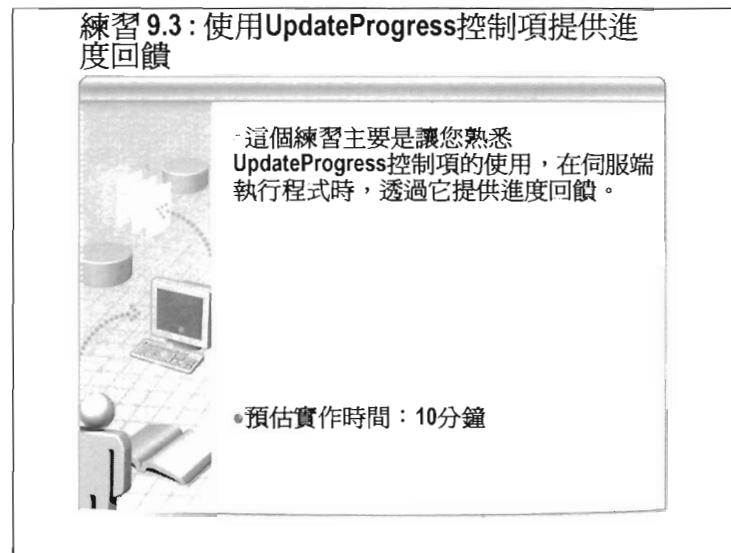
使用 **UpdateProgress** 控制項提供進度回饋

UpdateProgress 控制項可以在部分更新網頁的過程中，提供視覺化的進度回饋。在進行部分更新網頁的動作之前，或部分更新網頁的動作完成之後 UpdateProgress 控制項是隱藏的，不可視的。

您可以在 UpdateProgress 控制項中加入 ProgressTemplate 來設定要顯示的內容，您可以在其中使用文字、HTML 標籤、圖片、動畫檔，或控制項來進行設計。

一個網頁中，可以包含多個 UpdateProgress 控制項。UpdateProgress 控制項可以藉由 AssociatedUpdatePanelID 屬性和網頁中的 UpdatePanel 控制項關聯在一起。若未設定 UpdateProgress 控制項的 AssociatedUpdatePanelID 屬性，則網頁中 UpdatePanel 進行非同步更新動作時，就會顯示 UpdateProgress 控制項。

若設定 UpdateProgress 控制項的 AssociatedUpdatePanelID 屬性，將其和 UpdatePanel 控制項關聯在一起，那麼關聯的 UpdatePanel 控制項進行非同步更新時，才會顯示 UpdateProgress 控制項。



練習 9.3 : 使用 UpdateProgress 控制項提供進度回饋

目的：

這個練習主要是讓您熟悉 UpdateProgress 控制項的使用，在伺服端執行程式時，透過它提供進度回饋。

功能描述：

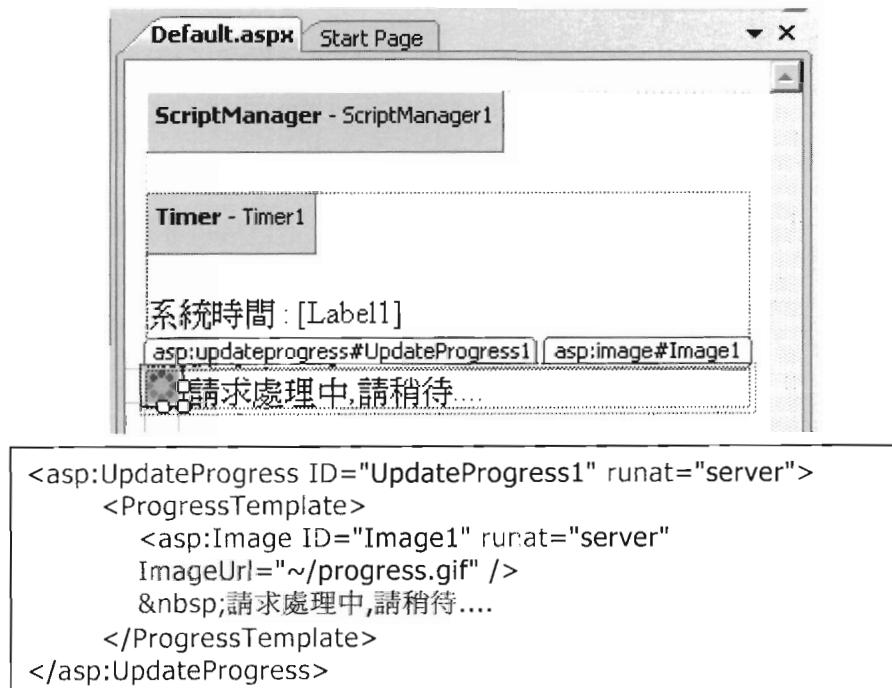
這個練習會在網頁中使用一個 UpdatePanel、Timer 控制項，當網頁執行時，定期在伺服端產生 1~10 之間的亂數資料與系統時間，然後以非同步更新方式，將亂數資料與系統時間更新在 UpdatePanel 中 Label 控制項中。

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. 開啓上一個練習完成的 Mod09_2 專案，或從「File」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9543\Practices\VB 或 CS\Mod09_2\Starter\Mod09_2」目錄，開啓 Mod09_2 專案。
3. 從「Solution Explorer」→點選專案名稱，按滑鼠右鍵，選取「Add Existing Item」，將「\U9543\Practices\VB 或 CS\Mod09_2\Starter」目錄下的 progress.gif 檔案加到專案。
4. 在網頁中，UpdatePanel 控制項下方加入一個 UpdateProgress 控制項。
5. 從「Toolbox」工具箱中拖拉一個 Image 控制項到網頁中 UpdateProgress 控制項之中。
6. 從「Properties」視窗上將 Image 控制項的「ImageUrl」屬性設為「~/progress.gif」。
7. 在 UpdateProgress 控制項內 Image 控制項之後加入以下文字：「請求處理中,請稍待....」，您的畫面看起來如下：



8. 在網頁</head>標籤上方，加入以下 CSS 樣式：

```
<style type="text/css">
#UpdateProgress1 {
    width: 100%; background-color: yellow;
    bottom: 0%; left: 0px; position: absolute;
}
</style>
```

9. 在 Timer1_Tick 事件處理常式中，第一行程式加上呼叫 Thread.Sleep 方法，讓程式暫停 3 秒執行：

Visual Basic

```
Protected Sub Timer1_Tick(ByVal sender As Object, ByVal e As System.EventArgs)
    System.Threading.Thread.Sleep(3000)
    Label1.Text = DateTime.Now.ToString("yyyy/MM/dd HH:mm:ss")
    Label2.Text = GetData()
End Sub
```

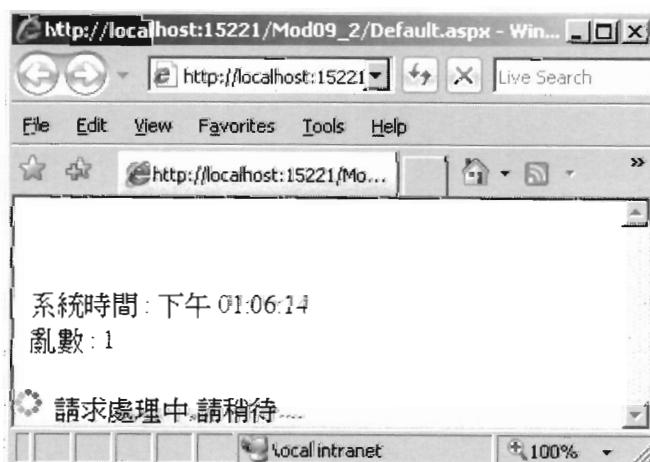
C#

```
protected void Timer1_Tick(object sender, System.EventArgs e)
{
    System.Threading.Thread.Sleep(3000);

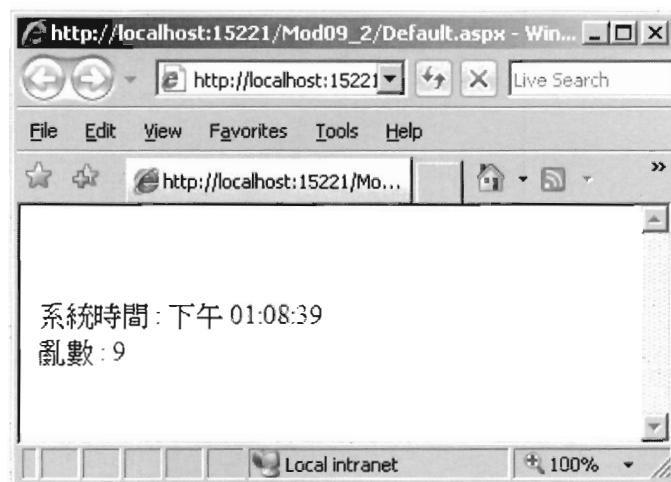
    Label1.Text = DateTime.Now.ToString("yyyy/MM/dd HH:mm:ss");
    Label2.Text = GetData();

}
```

10. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。每當取出系統時間與產生亂數的過程中，UpdateProgress 將會顯示進度提示：



而資料更新到畫面後，UpdateProgress 就會自動隱藏。

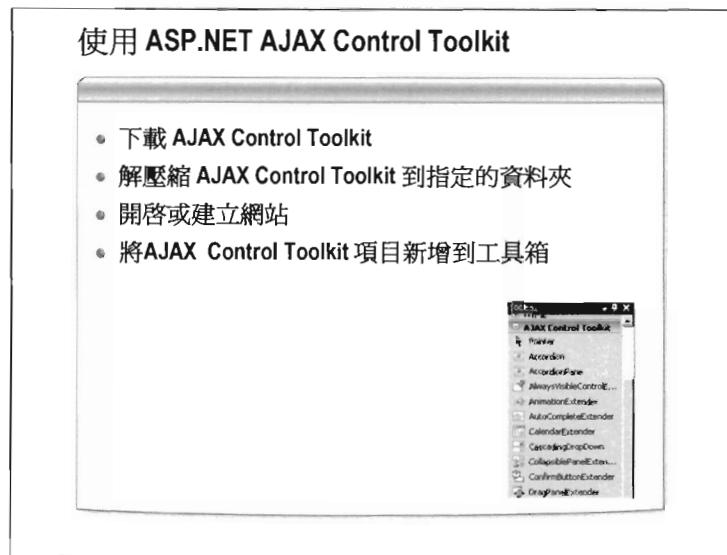


ASP.NET AJAX Control Toolkit 簡介

- 微軟的 ASP.NET AJAX 社群所分享的原始檔專案
- 目前可支援 ASP.NET 2.0 以及 ASP.NET 3.5
- 包含許多用戶端強大的功能
- 提供許多延伸的控制項以強化 ASP.NET AJAX
- 可從 <http://www.codeplex.com> 下載

ASP.NET AJAX Control Toolkit 簡介

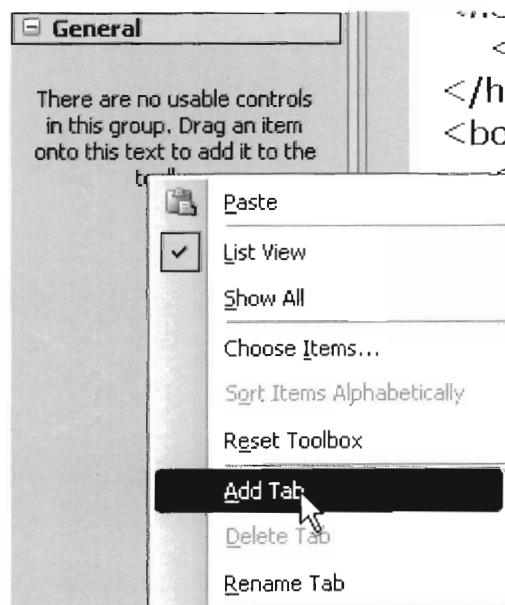
ASP.NET AJAX Control Toolkit 是一組以 ASP.NET AJAX 技術為基礎，所建構出來的 AJAX 延伸視覺控制項。也就是說，可以將 ASP.NET AJAX Control Toolkit 視為已經先做好的現成 AJAX 功能控制項，所以使用之前，也需要至以下網站下載：
<http://www.codeplex.com/AjaxControlToolkit>



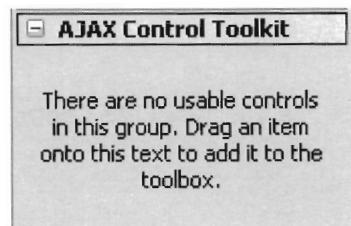
使用 ASP.NET AJAX Control Toolkit

您必需將 ASP.NET AJAX Control TookKit 的控制項加入至 Visual Studio 的 ToolBox 中 才能夠在網站中使用這些控件項。

首先開啓 Visual Studio，建立一個 ASP.NET Web Site。開啓其 Toolbox，並在 Toolbox 按下滑鼠右鍵，選擇「Add Tab」如下所示：



將新加入的 Tab 名稱設定為「AJAX Control Toolkit」：

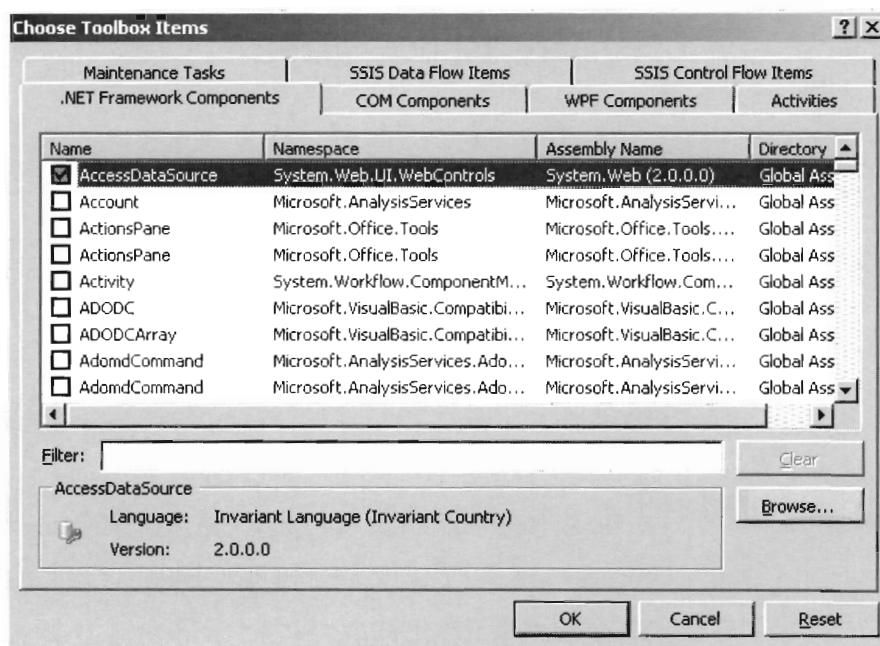


控制項加入步驟有兩種，第一種：

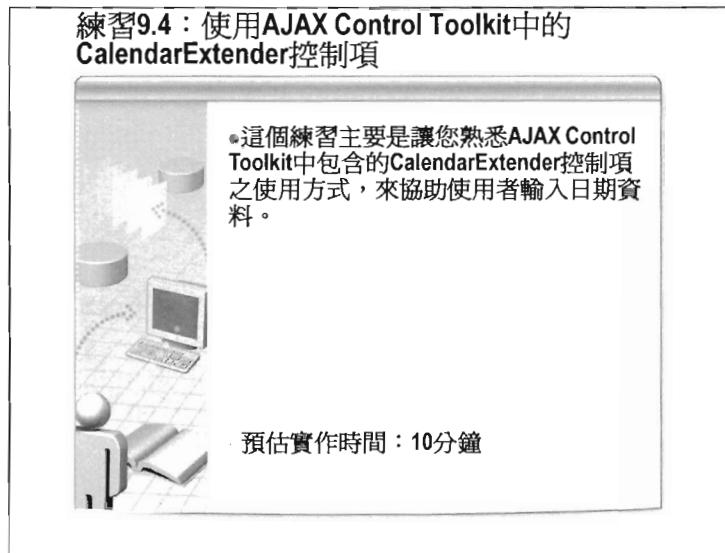
從檔案總管找到解壓縮的 ASP.NET AJAX Control Toolkit，找到 AjaxControlToolkit.dll 檔案。預設在解壓縮目錄下\SampleWebSite\Bin 子目錄中。從檔案總管，將 AjaxControlToolkit.dll 拖曳到 Visual Studio 工具箱中。

第二種方式如下：

在 Toolbox 視窗「AJAX Control Toolkit」頁，按下滑鼠右鍵，選擇「Choose Items...」將出現「Choose Toolbox Items」視窗。



選擇「.NET Framework Components」頁籤，再按下右下角的「Browse...」按鈕。選取「AjaxControlToolkit.dll」檔，按下「Open」按鈕。再按下「OK」按鈕。Toolbox 便會出現這些控制項。



練習 9.4 : 使用 AJAX Control Toolkit 中的 CalendarExtender 控制項

目的：

這個練習主要是讓您熟悉 AJAX Control Toolkit 中包含的
CalendarExtender 控制項之使用方式，來協助使用者輸入日期資
料。

功能描述：

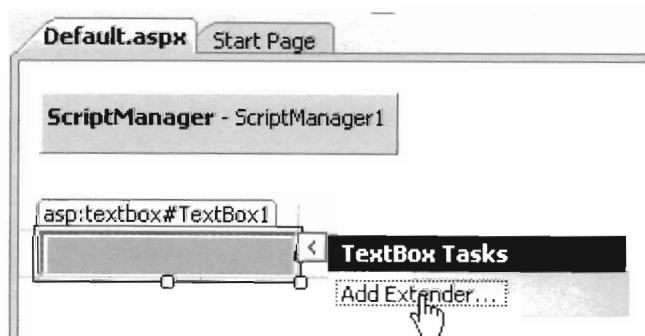
這個練習會在網頁中使用一個 CalendarExtender 控制項來擴充
TextBox 控制項的功能，使用者可以點選 TextBox 控制項旁的
Images 控制項，開啟月曆，再選取日期，以自動輸入日期資料。

預估實作時間：10 分鐘

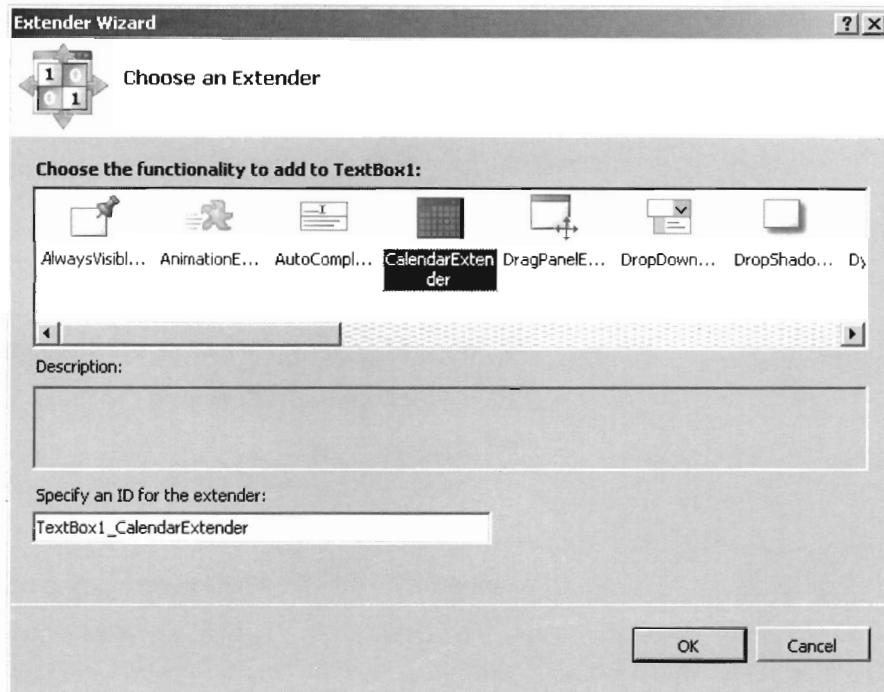
實作步驟：

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「\U9543\Practices\VB 或 CS\Mod09_4\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台收名為「Mod09_4」。
3. 從「Solution Explorer」→點選專案名稱，按滑鼠右鍵，選取「Add Existing Item」，將「\U9543\Practices\VB 或 CS\Mod09_4\Starter」目錄下的 Calendar_scheduleHS.png 檔案加到專案。
4. 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。
5. 從「Toolbox」工具箱中拖拉一個 TextBox 控制項到新網頁中 ScriptManager 控制項下方。在 TextBox 控制項加上一個 Image 控制項。
6. 使用「Properties」視窗，設定 Image 控制項的 ImageUrl 為「Calendar_scheduleHS.png」檔案。
7. 點選 TextBox 控制項的智慧型標籤，選取「Add Extender」：



8. 從 Extender Wizard 選取 CalendarExtender 控制項：



9. 使用「Properties」視窗，設定 CalendarExtender 控制項(預設工具產生的名稱為 TextBox1_CalendarExtender)的 PopupButtonID 為「Image1」(需手動輸入)。

您的標籤目前看起來如下：

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<cc1:CalendarExtender ID="TextBox1_CalendarExtender"
    runat="server"
    Enabled="True"
    TargetControlID="TextBox1"
    PopupButtonID="Image1">
</cc1:CalendarExtender>
<asp:Image ID="Image1" runat="server"
    ImageUrl="~/Calendar_scheduleHS.png" />
```

10. 執行網頁測試。在「Solution Explorer」→點選網頁→按滑鼠右鍵→選「View In Browser」。點選畫面中的月曆圖式，便會自動顯示月曆，選取月曆中的日期後，日期會自動填入 TextBox 控制項中。



總結

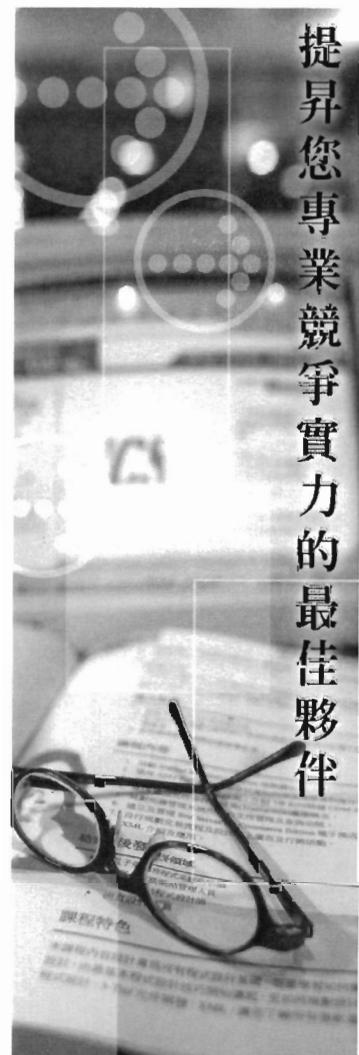
- **AJAX**開發架構可以設計局部更新的網頁
- **AJAX**提供更佳的互動式效果
- **ASP.NET AJAX Extension**提供多種伺服器控制項來協助開發網頁
- **AJAX Control Toolkit**提供多種擴充控制項來擴充原有控制項的功能

第十章：實用控制項運用

本章大綱

課程大綱.....	3
ASP.NET 實用控制項介紹	4
MultiView 與 View	5
FileUpload	7
練習 10.1：使用 MultiView 與 FileUpload 控制項	9
ImageMap.....	14
練習 10.2：使用 ImageMap	16
Wizard	19
Wizard 控制項使用簡介	21
WizardStep 設計方式.....	23
練習 10.3：建立 Web.sitemap 檔案.....	26
ASP.NET Mobile Web 應用程式簡介.....	30
Mobile Web Form	31
ASP.NET Mobile 控制項簡介.....	33
練習 10.4：建立 Mobile Web 應用程式.....	35
ObjectList.....	38

作者：
周季賢





c

大綱

- ASP.NET 實用控制項介紹
- ASP.NET Mobile Web 應用程式簡介
- 總結

課程大綱

在這個章節中將介紹 ASP.NET Web 應用程式在開發時所會用到的實用控制項與 ASP.NET Mobile 應用程式簡介。

本章介紹以下主題：

- ASP.NET 實用控制項介紹
- ASP.NET Mobile Web 應用程式簡介



ASP.NET 實用控制項介紹

開發應用程式時，總會有各種大大小小不同的需求出現，如何達到需求，除了要有高超的程式能力之外，控制項更是不可或缺的一項；本小節介紹四種不同功能的實用控制項，可幫助您在開發 Web 應用程式時，達到事半功倍的效果，本小節介紹內容如下：

- MultiView 與 View
- FileUpload
- ImageMap
- Wizard
- Wizard 控制項使用簡介
- WizardStep 設計方式

MultiView與View

- 顯示多重頁面，並且可自訂條件變更顯示
- View為其它控制項的容器
- MultiView為View控制項的容器
- 頁面切換方式
 - 指定MultiView的ActiveViewIndex
 - 按鈕控制項設定CommandName

MultiView 與 View

MultiView 是用來顯示多重頁面的一種控制項，利用 MultiView 可輕易的將多張因邏輯或判斷而關聯在一起的網頁濃縮，在一張網頁內即可達到多畫面的切換，且因為所需資訊與控制項均在同一頁，所以在資訊傳遞上亦比傳統的多網頁切換方便。

設計方式

MultiView 使用了 View 控制項來當作其子成員，每一個 View 控制項即代表其中一頁的檢視內容；所以如果要利用 MultiView 開發多重頁面，首先必須要在 MultiView 內加入自行決定數量的 View 控制項，接著便是在 View 控制項內置入該頁欲顯示的控制項或內容。

頁面切換方法

MultiView 一次只能顯示一個 View 控制項的內容，至於頁面的切換方式則沒有一定規則，可依執行時期的任何邏輯判斷來決定欲切換頁面，以下介紹兩種切換方式：

- 指定 MultiView 的 ActiveViewIndex

可利用程式來設定 MultiView 的 ActiveViewIndex 屬性，該屬性必須提供 View 控制項位於 MultiView 的索引值。

- 按鈕控制項設定 CommandName

MultiView 也支援利用按鈕控制項指定 CommandName 與 CommandArgument 來做切換，切換的設定方式如下：

CommandName	CommandArgument	效果
NextView	無	下一個 View
PrevView	無	上一個 View
SwitchViewByID	View 控制項 ID	指定 ID 的 View
SwitchViewByIndex	View 控制項索引值	指定索引值的 View

FileUpload

- 可讓使用者將檔案上傳至伺服器上。
- 可以限制上傳檔案的大小
- 可在伺服器端檢查上傳檔案的附檔名

```
Visual Basic
If FileUpload1.HasFile Then
    FileUpload1.SaveAs(Server.MapPath(FileUpload1.FileName))
End If
```

```
C#
if (FileUpload1.HasFile)
{
    FileUpload1.SaveAs(Server.MapPath(FileUpload1.FileName));
}
```

FileUpload

在各種的 Web 應用程式開發中，都免不了需要請使用者將資料傳送至伺服器的行為；ASP.NET 提供了 FileUpload 控制項，能夠讓我們輕鬆的利用該控制項提供讓使用者將檔案上傳至 ASP.NET Web 伺服器上的功能。

上傳方式

上傳至伺服器的程式相當簡單，僅需要利用 FileUpload 的 SaveAs 方法即可完成儲存至伺服器上的動作，不過需要注意的是 SaveAs 方法必須提供的是位於伺服器上的絕對路徑，所以在設計上，通常會使用 Server.MapPath 的方法，並在該方法內使用 FileUpload 的 FileName 作為參數來當作 SaveAs 方法的參數，如下所示：

```
Visual Basic
FileUpload1.SaveAs(Server.MapPath(FileUpload1.FileName))
```

```
C#
FileUpload1.SaveAs(Server.MapPath(FileUpload1.FileName));
```

Server.MapPath 方法會取得目前所在網頁位於該伺服器的實際路徑，並組合所參數內所提供的檔案路徑名稱組合成一個新的實際路徑，利用該方式便能簡單的將檔案存於網站內的資料夾。

另外，也可以利用 FileUpload 的 HasFile 屬性來判斷使用者是否有選擇檔案上傳，再決定是否要執行存檔的程式碼。

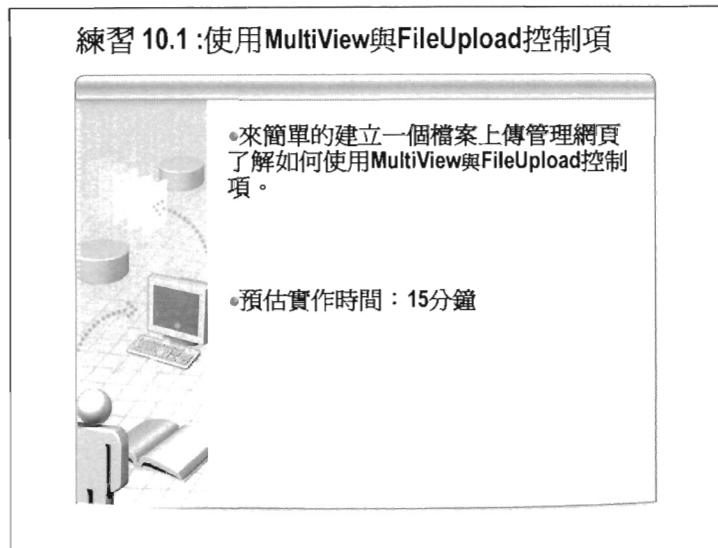
限制上傳檔案的大小

除了上傳能力之外，再使用 FileUpload 時也需要留心上傳檔案大小的控管，預設 ASP.NET Web 應用程式會限制使用者上傳檔案的大小為 4MB，如果允許上傳檔案大小超過 4MB，那麼就必須要設定 Web.Config 中的「`HttpRuntime`」項目的「`maxRequestLength`」屬性，該屬性為 Int32 的型別，以 KB 為單位，如下所示：

```
<system.web>
    <httpRuntime maxRequestLength="4096"/>
</system.web>
```

檢查上傳檔案的類型

最後，需注意 FileUpload 並不具備在使用者端檢查上傳檔案類型的能力；如需要檢查，必須要自行撰寫用戶端指定碼或伺服器端程式來做檔案類型或內容的判斷。



練習 10.1：使用 MultiView 與 FileUpload 控制項

目的：

建立一個檔案上傳管理網頁了解如何使用 MultiView 與 FileUpload 控制項。

功能描述：

在這個練習中，將加入一個 View 控制項至準備好的 MultiView 控制項內，在該控制項內加入 FileUpload 控制項來作為上傳用途，並且還會利用 RadioButtonList 控制項來切換功能頁面。

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「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod10_1\Starter\」

Mod10_1\」目錄，完成後，按下「Open」按鈕即可開啓網站。

3. 在「testWebHD.aspx」網頁內，從「Toolbox」工具箱中拖拉一個 RadioButtonList 控制項至網頁最上層，將「AutoPostBack」屬性設為 true，「RepeatDirection」設為 Horizontal。並且點選「Items」屬性開啟成員編輯視窗，依序加入兩個 ListItem 成員，成員的屬性如下所示：

成員	屬性	屬性值
成員一	Text	檢視內容
	Selected	True
成員二	Text	上傳

4. 完成之後，從「Toolbox」工具箱中拖拉一個 RadioButtonList 控制項至 MultiView 控制項的內的 ID 為 View1 的 View 控制項下，並且在內依序加入 Label、FileUpload 與 Button 控制項，控制項的屬性如下所示：

控制項	屬性	屬性值
Label	ID	Label1
	Text	請選擇檔案
FileUpload	ID	FileUpload1
Button	ID	Button1
	Text	確認上傳
	CommandName	PrevView

5. 完成之後，畫面應如下所示：



6. 註冊 RadioButtonList1 的 SelectedIndexChanged 事件處理常式，在程式碼區塊中加入下面的程式碼：

Visual Basic

```
Protected Sub RadioButtonList1_SelectedIndexChanged(ByVal sender As Object, ByVal e As System.EventArgs)
    MultiView1.ActiveViewIndex = RadioButtonList1.SelectedIndex
End Sub
```

C#

```
protected void RadioButtonList1_SelectedIndexChanged(object sender, EventArgs e)
{
    MultiView1.ActiveViewIndex = RadioButtonList1.SelectedIndex;
}
```

7. 註冊 Button1 的 Click 事件處理常式，在程式碼區塊中加入下面的程式碼：

Visual Basic

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    If FileUpload1.HasFile Then
```

```
FileUpload1.SaveAs(Server.MapPath("files") &
End If
End Sub
```

C#

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (FileUpload1.HasFile)
    {
        FileUpload1.SaveAs(Server.MapPath(@"files\" +
                                         FileUpload1.FileName));
    }
}
```

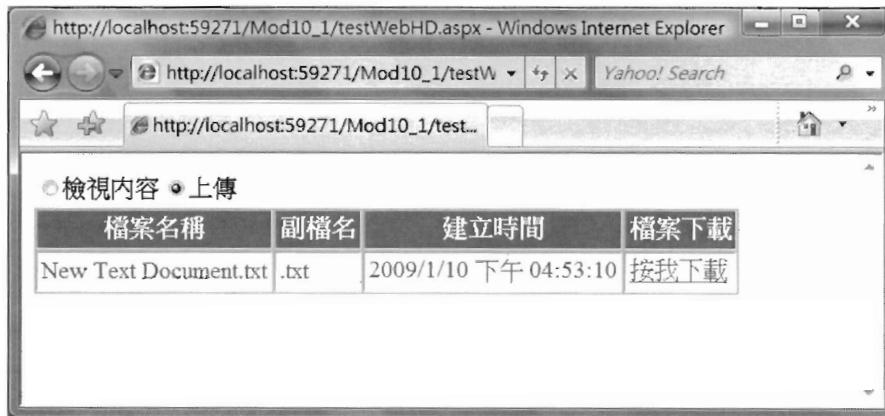
8. 在 Solution Explorer 中的「testWebHD.aspx」網頁上點右鍵，選擇「View in Browser」。執行結果如下

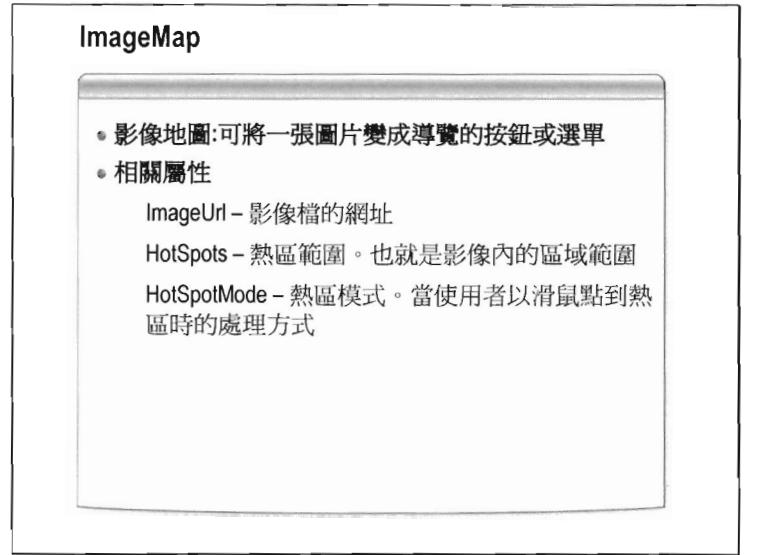


9. 點選「上傳」的 radioButton 之後，切換到上傳頁面，畫面應如下所示：



10. 選擇檔案後，按下確認上傳的按鈕，檔案應會順利上傳至伺服器，並且會切換至檢視內容頁面顯示伺服器檔案資訊與下載功能。畫面應如下所示：





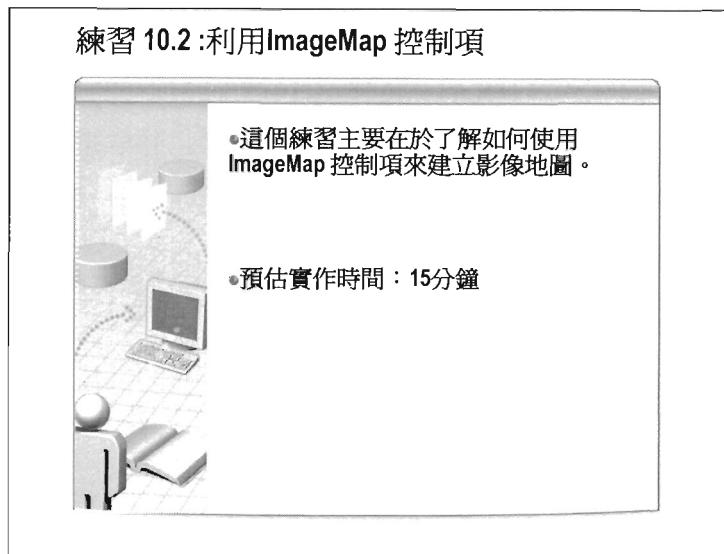
ImageMap

ImageMap 控制項可將一張圖片變成導覽的按鈕或選單。其關鍵屬性如下：

- **ImageUrl**
影像檔的網址。
- **HotSpots**
定義熱區範圍。也就是影像內的區域範圍。熱區範圍可以是一塊矩形、圓形、或者多邊形區域。
- **HotSpotMode**
熱區模式。當使用者以滑鼠點到某個熱區時的處理方式。預設為 NotSet，你可以參考下表：

HotSpotMode	說明
Inactive	當使用者點到熱區時，不做任何處理。
NotSet	不指定
Navigate	瀏覽到指定的網址。該網址由各個熱區物件的 NavigateUrl 指定，若未指定，則瀏覽到網站的根目錄。
PostBack	產生 PostBack 到伺服器端，你可以在伺服器端處理 Click 事件，並從事件的 ImageMapEventArgs 參數判斷是哪塊熱區被按下了。

注意 ImagMap 和 HotSpots 集合屬性中的每個熱區物件都有 HotSpotMode 屬性，如果 HotSpots 集合中的所有熱區物件的 HotSpotMode 都是 NotSet，那麼所有的熱區物件就都會採用 ImageMap 的 HotSpotMode，如果 ImagMap 的 HotSpotMode 也是 NotSet，那麼當使用者點到某個熱區時，就會瀏覽到網站的根目錄。



練習 10.2：使用 ImageMap

目的：

這個練習主要是讓您熟悉如何使用 ImageMap 控制項。

功能描述：

利用一個現有的影像檔製作地圖導覽，此影像檔包含三個區塊的圖形，分別用來代表「上一頁」、「下一頁」、以及「加入我的最愛」。每當使用者點選到其中一個區塊，就會在網頁上顯示使用者點選的區塊名稱。

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「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod10_2\Starter\」

Mod10_2\」目錄，完成後，按下「Open」按鈕即可開啟網站。

3. 在 Solution Explorer 中找到 ImageMapDemo.aspx，雙擊此項目以編輯此網頁，然後在編輯視窗中切換至 Design View。
4. 從「Toolbox」工具箱中拖拉一個 ImageMap 控制項到網頁上，接著到屬性視窗中設定 ImageUrl 屬性。點擊此屬性右邊的 [...] 按鈕以開啟選擇影像的對話窗。接著選擇 Map.jpg，按 OK 鈕。
5. 設定 HotSpotMode 屬性為 PostBack。
6. 點擊 HotSpots 屬性右邊的 [...] 按鈕，接著會開啟熱區編輯視窗。
7. 在視窗中點「Add」鈕旁邊的向下箭頭，選擇「RectangleHotSpot」，接著在右邊設定其 Right 和 Bottom 屬性為 63。設定PostBackValue 為 "上一頁"。
8. 重複上一個步驟，加入一個 RectangleHotSpot，設定 Bottom 屬性為 63，Left 為 64，Right 為 127。設定PostBackValue 為 "下一頁"。
9. 重複上一個步驟，加入一個 RectangleHotSpot，設定 Bottom 屬性為 63，Left 為 128，Right 為 191。設定PostBackValue 為 "加到我的最愛"。
10. 雙擊 ImageMap1 控制項以撰寫 Click 事件處理程序。參考以下程式碼：

Visual Basic

```
Protected Sub ImageMap1_Click(ByVal sender As Object,
    ByVal e As ImageMapEventArgs)
    Response.Write("你按了：" & e.PostBackValue)
End Sub
```

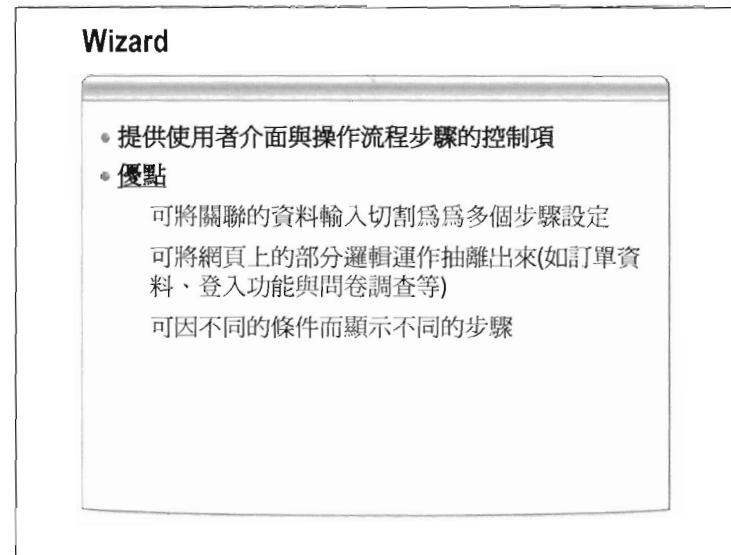
C#

```
protected void ImageMap1_Click(object sender,
    ImageMapEventArgs e)
{
    Response.Write("你按了: " + ePostBackValue);
}
```

11. 在 Solution Explorer 中的 ImageMapDemo.aspx 項目上點右鍵，選擇「View in Browser」。執行結果如下：

你按了: 下一頁





Wizard

一般來說，在網頁應用程式中要達到大量資料輸入的保存，或是有相關資料流程的運作，大部分會將資料在多個網頁中傳遞並使用狀態物件來保存，如：Session 物件、ViewState 物件等。

例如：今天想要做一份網頁問卷，然而假如問卷的問題太多，以至於必須分好幾個網頁讓使用者填寫，此時，跨不同網頁之間資料的保存就必須手動來撰寫程式控管了，然而在跨網頁應用程式時，使用狀態物件來保存資料不僅費時又難以設計。對程式設計師來說通常都是一個非常頭痛的問題。

在 ASP.NET 中，就提供了一組 Wizard 控制項，這組 Wizard 控制項不僅僅可以設計成多個步驟的資料流程運作，甚至提供很多的樣式和界面，讓開發者可以隨心所欲的設定，其使用者界面的客製化以及步驟流程之間的結合，實在是彈性非常大的一個控制項。

ASP.NET Wizard 控制項的特色

- 可將相關連的資料切割為多個步驟設定

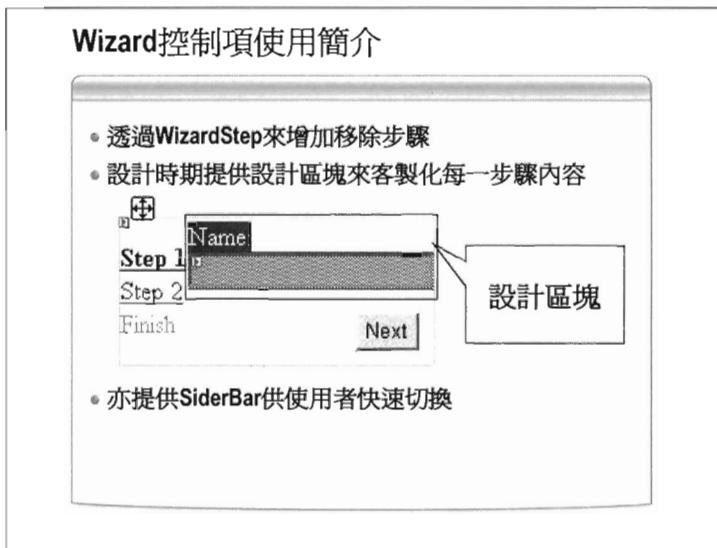
ASP.NET Wizard 最大的特色在於能夠設計多個步驟的使用者操作界面，一般來說，可以將相關聯的資料使用逐步執行的方式來記錄保存，而 ASP.NET Wizard 控制項正是可以讓開發者使用最少的程式碼達到最大效果的控制項物件。

- 可將網頁上的部份邏輯運作抽離出來

ASP.NET Wizard 控制項的另一個特色是可以將多重或是複雜的資料流程使用一步步執行的方式來保存資料。例：訂單資料，登入功能、問卷調查……等，都是 ASP.NET Wizard 控制項的普遍運用。

- 可設計因不同的滿足條件而有不同的步驟

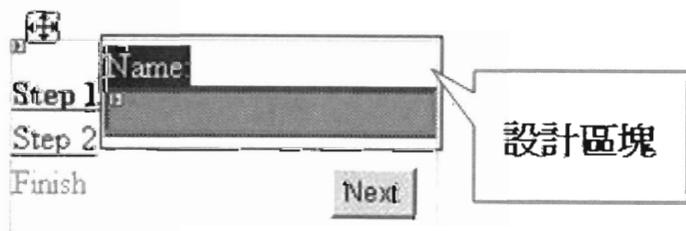
在 ASP.NET Wizard 中所設定的每一個步驟在運作時，都能隨時根據不同的條件而有不同步驟的變化，並非一貫不變的逐步執行，這樣的彈性設計為 ASP.NET Wizard 控制項的運用加分不少。



Wizard 控制項使用簡介

在設計 Wizard 控制項時，必須先決定與使用者互動所需要的步驟，每一個步驟都是一個顯示畫面，而步驟的增減則是在設計時期可利用 Wizard 控制項的屬性或工具來設定 WizardStep。至於步驟的設計上，每個步驟的設計區塊都是獨立分開的，開發者在步驟一和步驟二所設計的畫面都可以是不同的，在執行的過程中，設計區塊中的畫面將會依照步驟的順序而有不同的展示。

在畫面的設計上，ASP.NET Wizard 控制項提供了直覺化的設計區塊，開發者可以直接在從「Toolbox」工具箱中將所想要使用的控制項直接拖拉進設計區塊中，也可以在設計區塊中直接編輯所想要顯示提供給使用者的資訊。



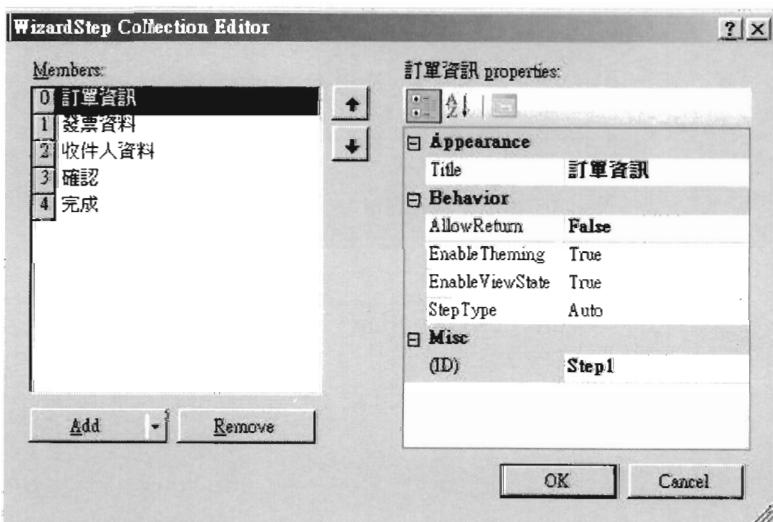
Wizard 控制項也可以讓開發者針對每一個步驟的使用者界面做客製化以及美觀設定，並且提供快速導覽步驟的功能讓使用者可以快速切換到想要執行的步驟，此區塊稱之為「sidebar」。





WizardStep 設計方式

透過智慧標籤(Smart Tag)來增減 Wizard 控制項的步驟等相關屬性是最直覺的設計方式，要使用智慧標籤(Smart Tag)可以在 Wizard 控制項上按滑鼠右鍵，選擇「Show Smart Tag」就會在控制項右上角出現一個小三角形，只需要點一下，就會出現「Wizard Tasks」。接著選擇「Add/Remove WizardSteps」就可以開啟「WizardSteps Collection Editor」工具。另外，點選 Wizard 控制項後在針對其屬性視窗的「WizardStep」亦可開啟「WizardStep Collection Editor」工具。



設計時必須利用 Wizard 的 Smart Tag 作 Step 的切換

在設定 Wizard 控制項的眾多屬性中最重要的就是設定其「Step」，因為「Step」是用來控管使用者操作的步驟以及畫面運作的步驟，在智慧標籤(Smart Tag)上設定 Step 的方式只需要先點到「Step」下拉選擇其中一個項目(如：Step1、Step2...)步驟就行了。等選到其中一個步驟之後，就可以直接在畫面上設定那個步驟的畫面了。



設定 StepType 決定步驟類型

每一個 Wizard 控制項中的 Step 都有各自的 StepType 屬性來決定每個步驟的類型，可能是開始的步驟(Start Step)或是結束的步驟(Complete Step)等等。以下用列表的方式，列出 StepType 所擁有的屬性以及其意義：

StepType	意義
WizardStepType.Auto	這個步驟會自動判斷目前所在的執行步驟位置應該是哪一類型，為預設值
WizardStepType.Start	這是第一個執行步驟，且並不會產生 Previous (上一步)按鈕
WizardStepType.Step	這個執行步驟是定義介於第一執行步驟和最後執行步驟之間，會有 Previous(上一步)按鈕和 Next(下一步)按鈕產生

WizardStepType.Finish	這是最後一個執行步驟，會將所有的使用者資料保存，會產生一個 Finish (完成)按鈕
WizardStepType.Complete	這是最後一個步驟畫面，並不會產生任何的按鈕，通常是用來展示所有保存的資料

WizardStep 切換方式

- ActiveStepIndex 屬性

只有在執行時期可設計，功能為動態去控制步驟之間的切換。只是這個屬性是使用索引的方式來設定。

```
Visual Basic
Wizard1.ActiveStepIndex = Wizard1.WizardSteps.IndexOf (Me.步驟名稱)
```

```
C#
Wizard1.ActiveStepIndex = Wizard1.WizardSteps.IndexOf (this.步驟名稱);
```

- MoveTo 方法

可以使用這個方法在執行時期動態去控制步驟之間的切換。

```
Visual Basic
Wizard1.MoveTo(Me.步驟名稱)
```

```
C#
Wizard1.MoveTo(this.步驟名稱);
```

練習 10.3 : 使用 Wizard 控制項

•了解如何使用 Wizard 控制項來建立使用者介面與操作流程步驟相關的網頁。

•預估實作時間：15分鐘

練習 10.3：使用 Wizard 控制項

目的：

這個練習主要是讓您了解如何使用 Wizard 控制項來建立使用者介面與操作流程步驟相關的網頁。

功能描述：

從工具箱中將 ASP.NET Wizard 控制項加入網頁中設定與使用，並了解 ASP.NET Wizard 控制項的基本功能與運作方式。模擬使用者在網頁上需要逐步輸入名字和 E-MAIL，最後的頁面將會顯示使用者在每個步驟中所輸入的資料。

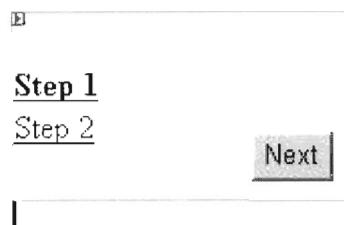
預估實作時間：15 分鐘

實作步驟：

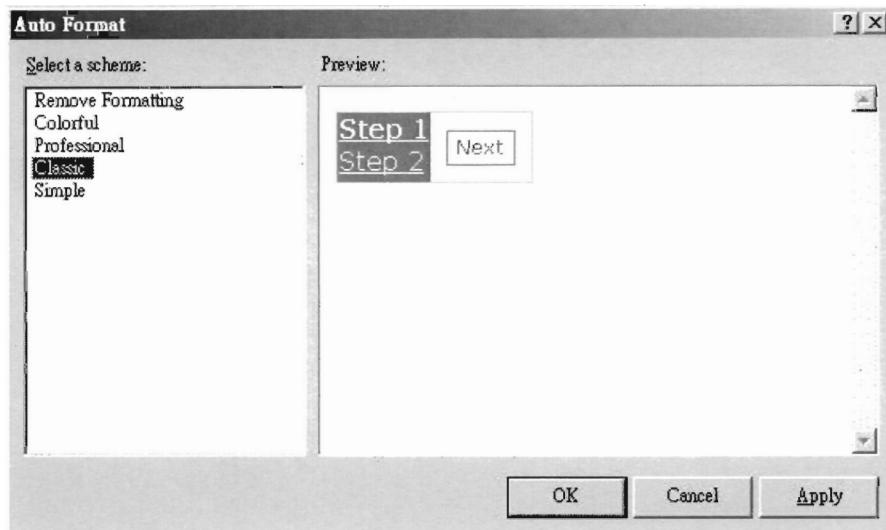
1. 從『Start』→『Program』→『Microsoft Visual Studio 2008』→『Microsoft Visual Studio 2008』，啓動 Visual Studio 2008 開發環境。
2. 從『File』→『New Web Site』→選取『Empty Web Site』→將『Location』設為『File System』並點選『Browse...』按鈕選

取「\U9543\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」，檔名命名為「DemoWizardStep.aspx」，清除「Place code in separate file」核取方塊，建立來源網頁。
4. 在「DemoWizardStep.aspx」網頁內，從「Toolbox」工具箱中拖拉一個 Wizard 控制項到「DemoWizardStep.aspx」中。



5. 點選 Wizard1 控制項 → 按滑鼠右鍵 → 選「Auto format」→ 選擇一種樣式，在此範例選擇「Classic」→ 按下「OK」按鈕。



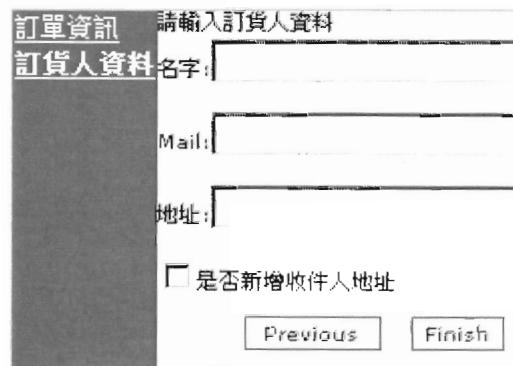
6. 點選 Wizard1 控制項 → 按滑鼠右鍵 → 選「Show Smart Tag」→ 選擇「Add/Remove WizardSteps」。
7. 在視窗「WizardStep Collection Editor」中選擇「Step1」→ 設定「Title」為「訂單資訊」→ 設定「StepType」為「Start」→ 按下「OK」按鈕 → 關閉「WizardStep Collection Editor」視窗。

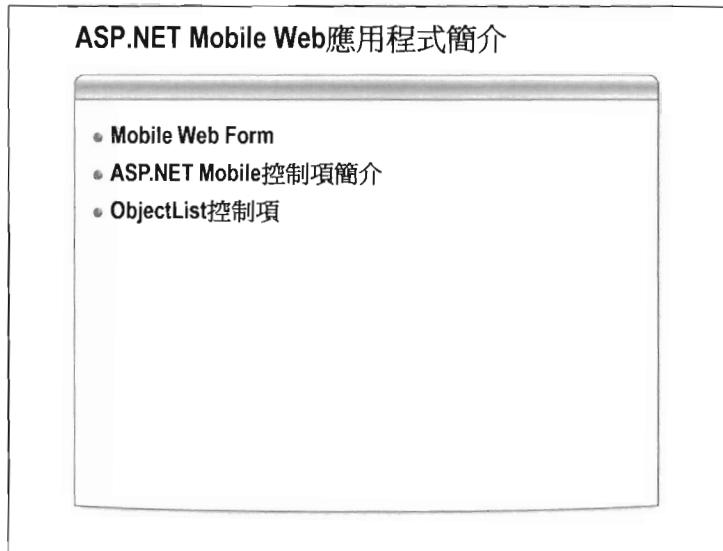


8. 在 Wizard1 可設計的區塊中，先打上「請依照步驟完成訂單資料」字串。
9. 緊接著設定訂貨人資訊步驟。點選 Wizard1 控制項 → 按滑鼠右鍵 → 選「Show Smart Tag」 → 選擇「Add/Remove WizardSteps」 → 點選「Step2」 → 設定「Title」為「訂貨人資訊」 → 按下「OK」按鈕。
10. 點選 Wizard1 控制項 → 按滑鼠右鍵 → 選「Show Smart Tag」 → 選擇「訂貨人資訊」。
11. 在 Wizard1 可設計的區塊中 先輸入「名字」 → 從「Toolbox」工具箱中拖拉一個 TextBox 控制項到 Wizard1 中並將其命名為「txtName」。重覆此步驟分別設定「Mail」、「地址」，接著再拖拉一個 CheckBox 控制項到 Wizard1 中並將其命名為「cb」，設定 Text 屬性為「是否新增收件人地址」，畫面如下：



12. 點選 Wizard1 上的訂單資訊，在「Solution Explorer」→點選 DemoWizardStep.aspx 網頁→按滑鼠右鍵→選「View In Browser」。
13. 點一下「訂貨人資料」切換步驟，結束程式執行，關閉瀏覽器。





ASP.NET Mobile Web 應用程式簡介

Mobile Web 應用程式是 ASP.NET 用來開發行動裝置的網頁技術，該中開發技術在設計上與一般的 ASP.NET Web 應用程式相差不遠。

本小節將介紹以下內容：

- Mobile Web Form
- ASP.NET Mobile 控制項簡介
- ObjectList 控制項

Mobile Web Form

- 針對行動裝置所開發的ASP.NET網頁
- 一般Browser亦可瀏覽
- 與ASP.NET Web Form相同之處
 - 副檔名也是aspx
 - 網頁結構亦可用Single File與Code-Separation
- 與ASP.NET Web Form不同之處
 - 僅能使用Mobile控制項
 - 一個Mobile Web Form中能使用多個form控制項
 - Visual Studio 2008並無提供建立用範本
 - 可設定DeviceFilter來辨識瀏覽該網站用的裝置

Mobile Web Form

Mobile Web Form 是用來開發行動裝置的網頁技術，行動裝置受限於螢幕大小以及硬體規格不盡相同，開發者必須開發盡量適合所有行動裝置微型瀏覽器可以使用的網頁程式。

與 ASP.NET 相同之處

Mobile Web Form 副檔名也是 aspx，所使用的控制項也是伺服器端控
制項。

在開發工具中也是分為兩種方式，Single File 的開發法會產生下列標
籤：

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="Mobile_Default" %>
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls" Assembly="System.Web.Mobile" %>
```

Code-Separation 開發法則產生下列標籤：

```
<%@ Page Language="VB" Inherits="Mobile_Default" %>
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.
```

```
MobileControls" Assembly="System.Web.Mobile" %>
```

與 ASP.NET 不同之處

雖然 Mobile Web Form 所使用的控制項也是伺服器端控制項，但是並非唯一般的伺服器控制項，必須為 Mobile Web Control，也就是控制項標籤必須要是以「`<Mobile:」` 開頭，並非為「`<ASP:」`。

由於行動裝置的操作畫面比較小，所以必須依靠多張表單來切割步驟或是展現內容，所以在 Mobile Web Form 在設計上，是可以支援多個 Mobile Form 控制項的加入。

因為 Visual Studio 2008 設計上的問題，在開發 Mobile Web 應用程式時，無法支援在設計時使用 Design 頁面來編輯，必須要用 Source 頁面。另外 Visual Studio 2008 預設並沒有提供 Mobile Web 應用程式的建立範本，如果需要，可至 Visual Web Developer Team Blog 下載。
網址為：<http://blogs.msdn.com/webdevtools/archive/2007/09/17/tip-trick-asp-net-mobile-development-with-visual-studio-2008.aspx>。下載之後，只需要將該壓縮檔解開，再分別將所需要的範本壓縮當複製到 [My Documents]\Visual Studio 2008\Templates\ItemTemplates\Visual Web Developer 目錄內即可。

就如同 ASP.NET Web 應用程式能夠因為 Browser 的不同而決定呈現不同的 Html 內容，Mobile Web 應用程式也具備此能力，只要設定好 Web.config 檔案中利用「`deviceFilters`」區段來指定裝置的相關參數，屆時即可讓 Mobile Web 應用程式辨識出該裝置，並提供不同的呈現方式。

ASP.NET Mobile 控制項簡介

- 大部分控制項名稱與ASP.NET控制項相同，使用方法亦相當相似
- **Form**
mobile控制項的容器，一Mobile Web Form可有多個
- **Command**
似Button控制項，也具有Click事件
- **Label**
與ASP.NET控制項的Label相同，但有StyleReference屬性可設定風格

ASP.NET Mobile 控制項簡介

ASP.NET Mobile 控制項有許多都與一般網頁的控制項類似，大部分功能都可以不需重新學習，不過部分控制項功能會略為縮減。以下介紹幾個簡單常用的控制項。

Form 控制項

由於行動裝置的操作畫面比較小，所以必須依靠多張表單來切換步驟或是展現內容，所以在網頁放置多個 Mobile Form 控制項，再將所需的其他控制項放入 Mobile Form 控制項。

設計時期，控制 Mobile Web Form 的切換有幾種方式，第一種是設定 Link 控制項的 NavigateUrl 屬性，如以下標籤：

```
<mobile:Link ID="Link1" Runat="server"
NavigateUrl="#Form2">Link</mobile:Link>
```

第二種方法是撰寫程式指定 ActiveForm 屬性，範例如下：

```
Visual Basic
Protected Sub Command1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Me.ActiveForm = Form2
End Sub
```

```
End Sub
```

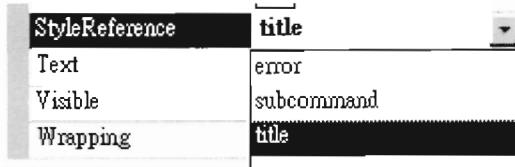
```
C#
protected void Command1_Click(System.Object sender, System.EventArgs e) {
    this.ActiveForm = Form2;
}
```

Command 控制項

該控制項可視為 Mobile Web Form 的 Button 控制項，使用方式均與 Button 控制項相同，也都是利用其 Click 事件來撰寫程式。

Label 控制項

此控制項與 ASP.NET Web 應用程式的 Label 控制項相同，均是用來顯示文字，不過略有不同的是 Label 控制項就有可以透過 StyleReference 屬性指定 Mobile 控制項的樣式集，如下圖：



練習 10.4 :建立 Mobile Web 應用程式



•了解如何練習建立 Mobile Web 應用程式，以及 Mobile Web 應用程式的控制項使用方式。

•預估實作時間：15分鐘

練習 10.4：建立 Mobile Web 應用程式

目的：

這個練習主要是讓您熟悉如何建立 Mobile Web 應用程式，以及 Mobile Web 應用程式的控制項使用方式。

功能描述：

透過使用者輸入姓名，讓視窗能夠切換後並且對使用者打招呼。

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「File System」按鈕並選取「\U9543\Practices\VB 或 CS\Mod10_4\Starter\Mod10_4\」目錄，完成後，按下「Open」按鈕即可開啓網站。

3. 在「Solution Explorer」視窗對 HelloUser.aspx 雙擊左鍵來開啟 Source 編輯頁面。

4. 在「<body>」區段內加入以下內容：

```
<mobile:form id="form1" runat="server">
    <mobile:Label ID="Label1" Runat="server">
        請輸入姓名:
    </mobile:Label>
    <mobile:TextBox ID="TextBox1" Runat="server">
    </mobile:TextBox>
    <mobile:Command ID="Command1" Runat="server"
        OnClick="Command1_Click">確定</mobile:Command>

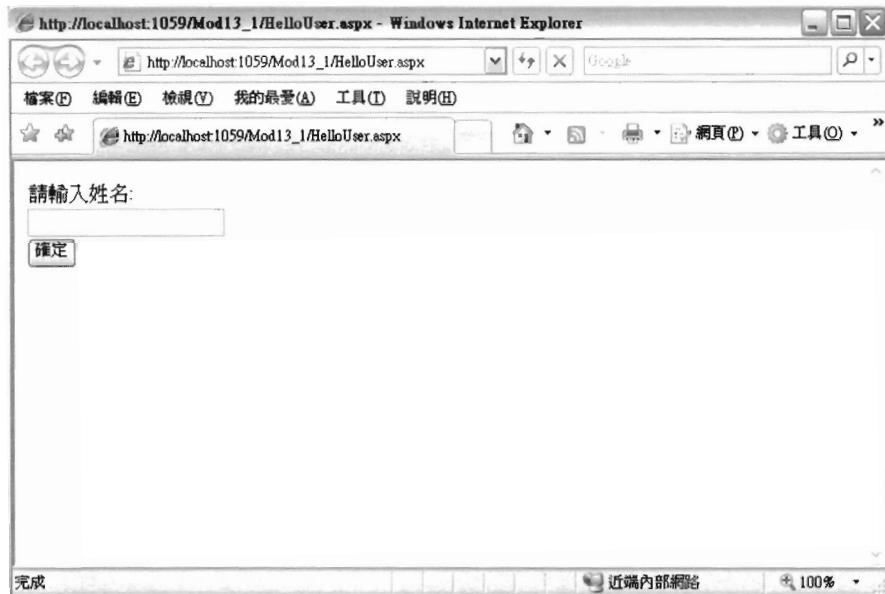
</mobile:form>
<mobile:Form ID="Form2" Runat="server">
    <mobile:Label ID="Label2" Runat="server">Label</m
obile:Label>
</mobile:Form>
```

5. 再「<script runat="server">」程式碼區塊中加入下面的程式碼：

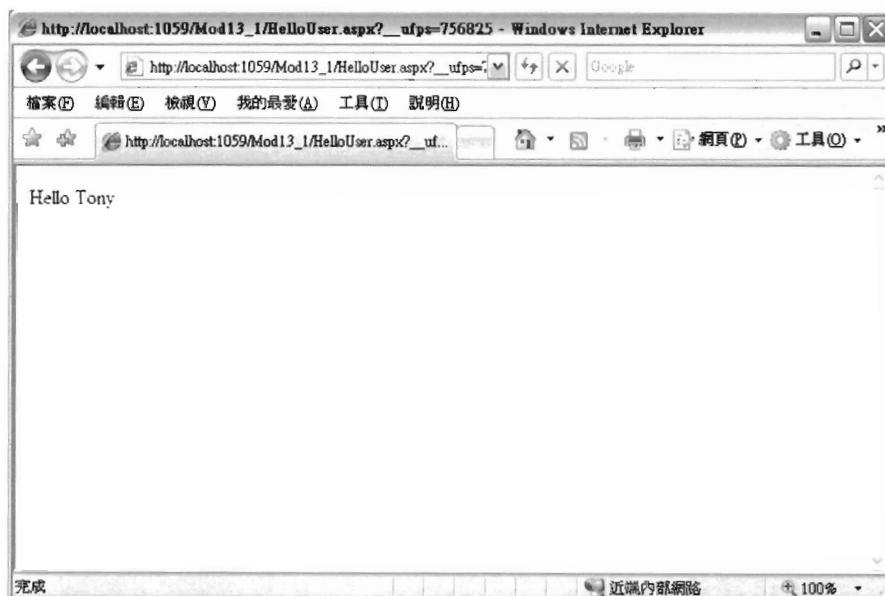
```
Visual Basic
Protected Sub Command1_Click(ByVal sender As Object, ByVal e A
s
System.EventArgs)
    Me.Label2.Text = "Hello " + Me.TextBox1.Text
    Me.ActiveForm = Me.Form2
End Sub
```

```
C#
protected void Command1_Click(object sender, EventArgs e)
{
    this.Label2.Text = "Hello " + this.TextBox1.Text;
    this.ActiveForm = this.Form2;
}
```

6. 在「Solution Explorer」視窗對 HelloUser.aspx 按下右鍵選擇「View in Browser」來開啟測試網頁。開啟畫面如下：



7. 在 TextBox 中輸入「Tony」後，按下確定；視窗將會導向 Form2，並顯示「Hello Tony」的文字，畫面如下：





ObjectList

ObjectList 算是 Mobile Web 應用程式中，較為複雜的控制項，該控制項能夠以多重清單的方式提供複雜物件內容的檢視。

該控制項在設計上只需要使用 DataSource 屬性指定複雜物件之後，再呼叫 DataBind 方法即可完成資料繫結。不過需要注意的是物件必須為 DataView、DataSet 或是實作 IEnumerable 介面物件才可。

在使用上除了繫結之外還可以利用以下的方式客製化屆時顯示的內容。

自訂顯示欄位

- TableFields

設定為欲顯示在第一頁清單內的資料欄位，該屬性可設定為以分號隔開的字串內容。

- LabelField

在針對每一筆資料檢視時，位於檢視頁內容上的識別用欄位，該屬性可用字串的方式來指定欲顯示的欄位名稱。

自訂功能事件

- Commands 屬性

該屬性可開啓 Command List 的編輯視窗，在視窗內可新增或移除要在檢視每一筆資料時所提供的功能命令。

- ItemCommand 事件

會在使用者點選該控制項的功能命令時觸發，並且在觸發時，會提供一個型別為 ObjectListCommandEventArgs 的參數，該參數內可以利用 CommandName 屬性得知是由哪個功能命令被點選後所觸發的，也可由此決定該執行的程式區段。

總結

- **ASP.NET實用控制項介紹**
- **MultiView與View**
- **FileUpload**
- **ImageMap**
- **Wizard**
- **ASP.NET Mobile Web應用程式簡介**
- **Mobile Web Form**
- **ASP.NET Mobile控制項簡介**
- **ObjectList控制項**

第十一章：網站錯誤客製化

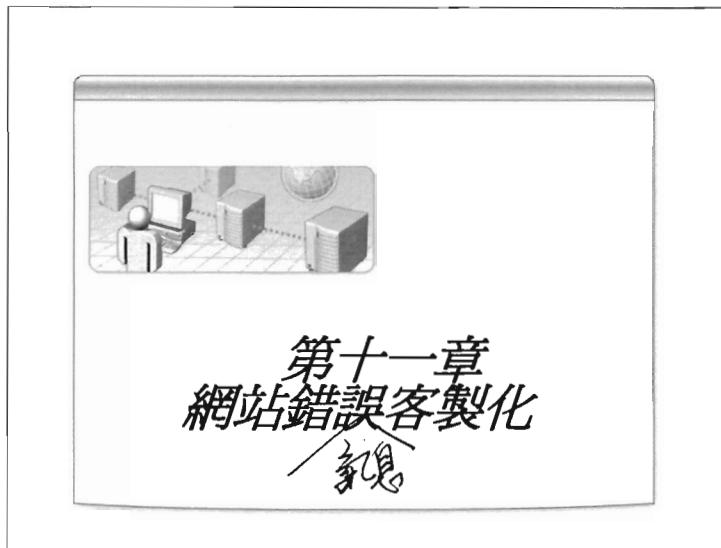
本章大綱

課程大綱.....	3
網頁常見錯誤.....	4
執行時期錯誤處理.....	5
預設錯誤頁面.....	7
啓用除錯設定.....	8
本機與遠端錯誤頁面.....	9
例外錯誤處理流程.....	11
頁面層級錯誤.....	12
練習 11.1：頁面層級錯誤處理.....	14
應用程式層級錯誤.....	18
練習 11.2：應用程式層級錯誤處理.....	20
自訂錯誤頁面.....	23
練習 11.3：使用 ErrorPage 設定網頁自訂錯誤.....	25
練習 11.4：應用程式層級與自訂錯誤.....	29
通用 HTTP 錯誤處理.....	33

作者：

許薰尹



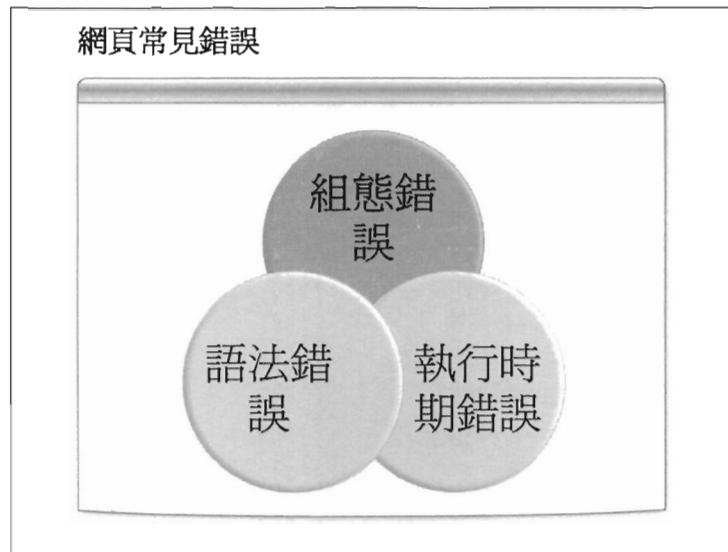




課程大綱

這個章節主要介紹如何處理未使用 Try .. Catch 捕捉例外錯誤處理的網頁應該如何進行頁面層級、或應用程式層級的自訂錯誤處理。本章將介紹以下主題：

- 網頁常見錯誤
- 執行時期錯誤
- 預設錯誤頁面
- 本機與遠端錯誤頁面
- 例外錯誤處理流程
- 頁面層級錯誤
- 應用程式層級錯誤
- 自訂錯誤頁面
- 通用 HTTP 錯誤處理



網頁常見錯誤

當網頁執行時，ASP.NET 會將錯誤資訊傳送給使用者。通常網頁發生的錯誤可分為幾大類：

- 組態錯誤：網站的 Web.Config 設定錯誤，可能是設定某個組態的位置錯誤。
- 語法錯誤：程式語法不正確導致。
- 執行時期錯誤：在網頁執行過程中發生，這些錯誤通常無法在設計階段處理。

執行時期錯誤處理

- 通常利用程式除錯
 使用 Try..Catch 例外錯誤處理常式
- 範例

Visual Basic <pre>Try '可能發生錯誤的程式碼 Catch ex as Exception '例外狀況處理程式 End Try</pre>	C# <pre>try { //可能發生錯誤的程式碼 } catch (Exception ex) { //例外狀況處理程式 }</pre>
---	--

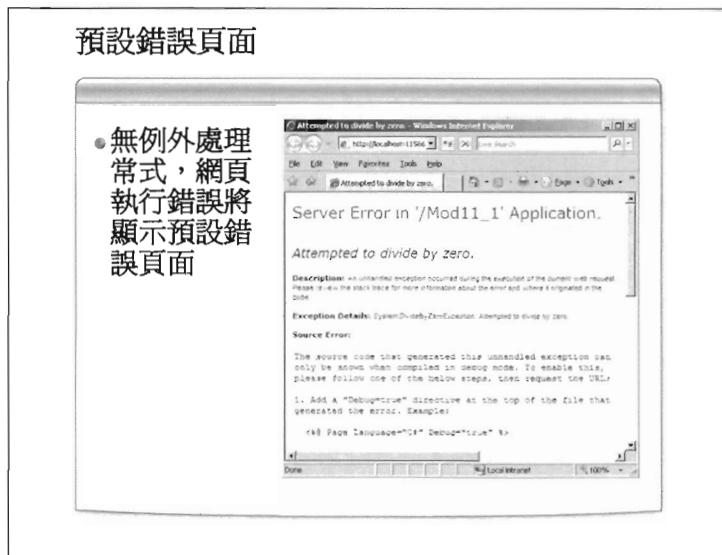
執行時期錯誤處理

ASP.NET 應用程式和一般的.NET 程式一樣，程式在執行時期，若讀取使用者所輸入的資料進行運算，或是存取檔案、資料庫等外部資源時，使用者輸入了錯誤的資料，或是因為權限不足等原因，導致程式在執行時期無法正常的執行而導致執行時期錯誤，ASP.NET 執行環境會產生 `Exception` 物件。您可以利用 `Try...Catch` 錯誤處理常式來，捕捉程式中所發生的 `Exception`，並且使用自訂的例外狀況邏輯處理：

```
Visual Basic
Try
    '可能發生錯誤的程式碼
Catch ex as Exception
    '例外狀況處理程式
End Try
```

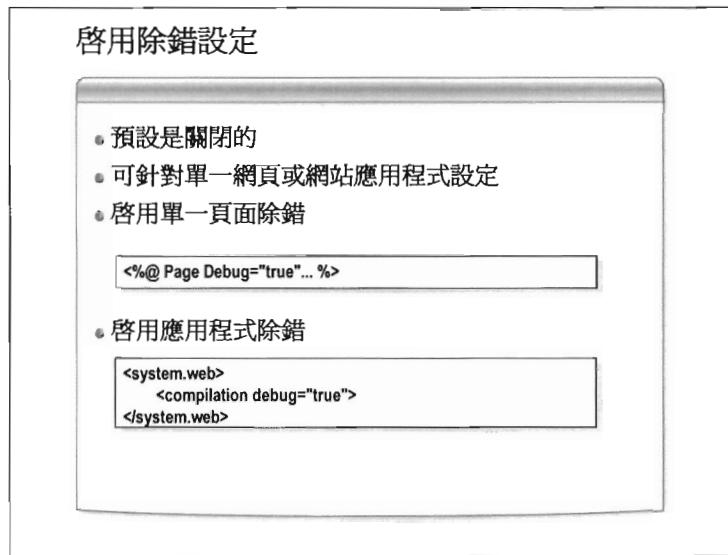
```
C#
try
{
    //可能發生錯誤的程式碼
}
catch (Exception ex) {
    //例外狀況處理程式
}
```

若是在程式中，有未被捕捉到的 Exception，則 ASP.NET 會觸發 Page 物件的 Error 事件，你也可以利用這一個事件執行頁面層級的錯誤處理。若是在 Page 中沒有處理，則該狀況就會被傳遞到應用程式執行環境之中，觸發 Application 的 Error 事件。如果在應用程式層級，還是沒有去捕捉這一個例外狀況，則 ASP.NET 執行環境就會產生預設的錯誤訊息。



預設錯誤頁面

若 ASP.NET 網頁執行時發生錯誤，而您沒有撰寫例外處理常式，則網頁將無法執行，ASP.NET 會顯示一個黃色頁面描述錯誤資訊，如上圖所示。



啓用除錯設定

若在執行時期發生例外錯誤，您可以在網站或網頁啓用除錯，以便於透過錯誤物件來檢視錯誤訊息，或呼叫堆疊的資訊，幫助修訂這些問題。預設除錯是關閉的，您可以針對單一網頁或網站應用程式設定除錯。

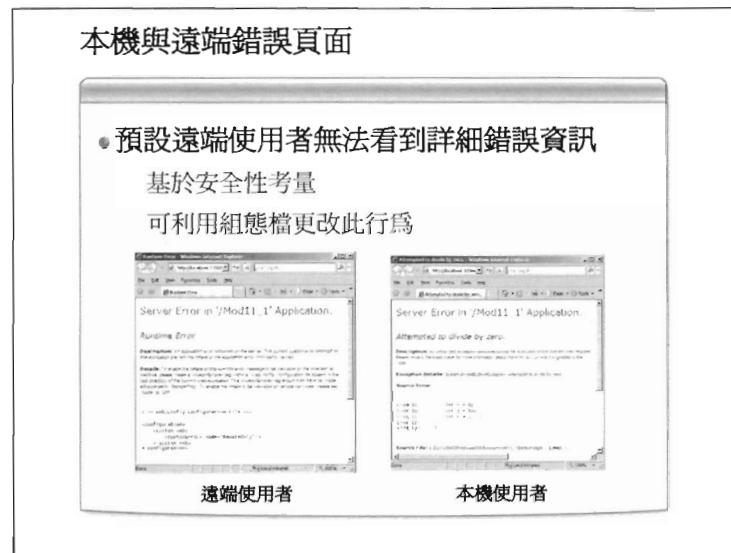
若要除錯一個網頁，可以將 Page 指示詞的 debug 設為 true：

```
<%@ Page Debug="true" ... %>
```

若要除錯整個網站中所有網頁，可以在 Web.Config 檔進行統一的設定：

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

將 debug 設定為 true 會影響效能，所以預設它的值是 false，只有在開發期間想透過它進行除錯時，才需要將這個值設定為 true。

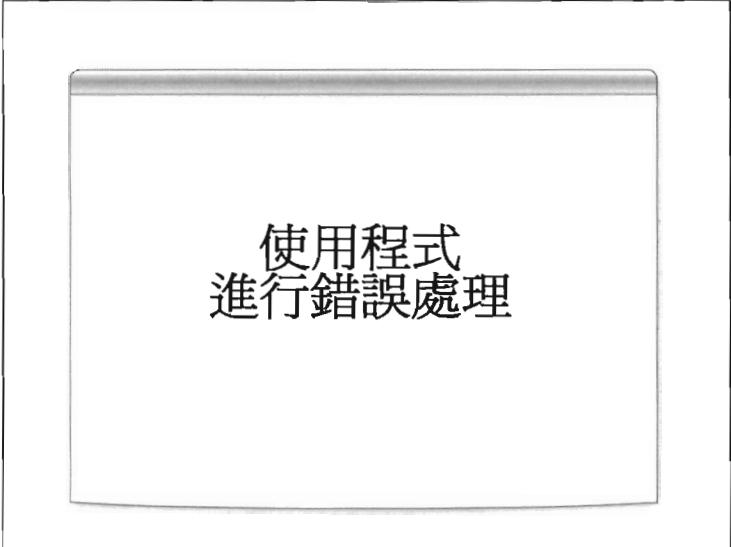


本機與遠端錯誤頁面

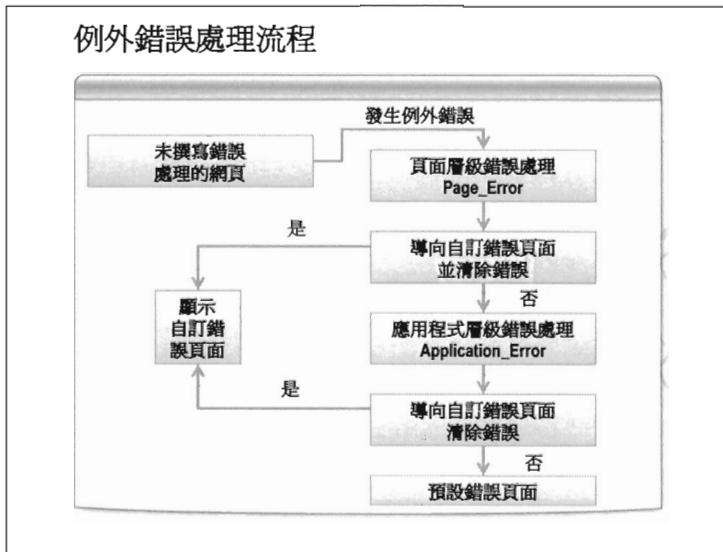
若啓用了除錯設定，預設本機使用者可以看到預設的錯誤頁面，顯示錯誤訊息、例外錯誤與追蹤堆疊等資訊，如上方右圖。

若沒有看到這個錯誤頁面，請檢視 Web.Config 檔案，是否有將 <system.web> 區段下<compilation>子區段的 debug 設定為 true，或將 Page 指示詞的 debug 設定為 true，這樣在發生例外錯誤時，ASP.NET 會將除錯符號插入編譯過的頁面，以利除錯。

但遠端與本機的使用者所看到的 ASP.NET 預設錯誤頁面是不相同的。本機使用者看到的是如上圖的錯誤頁面；而因為安全性的理由，遠端使用者看不到記錄詳細錯誤的頁面，如上方左圖。您可以利用組態檔更改此行為，本章稍後再說明。



使用程式
進行錯誤處理



例外錯誤處理流程

當網頁沒有撰寫錯誤處理常式的情況下執行發生錯誤時，ASP.NET 會先檢視是否撰寫了頁面層級的錯誤處理常式(Page_Error)，若沒有則檢視應用程式層級的錯誤處理常式(Application_Error)。若網頁並沒有撰寫這些例外錯誤處理程式，最終將會顯示預設錯誤頁面。

您可以在頁面等級或應用程式等級的 Error 事件處理常式中，進行錯誤處理，然後叫用 Server.ClearError 方法清除錯誤，您可以利用 Server.Transfer 方法導向自訂的錯誤頁面來呈現錯誤訊息。上圖是通用的例外錯誤處理流程。

頁面層級錯誤

- 擷取 Page 物件的 Error 事件
- 針對一個網頁統一進行錯誤處理
- 不需在網頁中撰寫 Try...Catch 程式區段捕捉錯誤

Visual Basic

```
Protected Sub Page_Error(ByVal sender As Object, ByVal e As EventArgs)
    Server.Transfer("GenErr.aspx?err=" & Server.GetLastError().Message)
    Server.ClearError()
End Sub
```

C#

```
protected void Page_Error(object sender, EventArgs e) {
    Server.Transfer("GenErr.aspx?err=" + Server.GetLastError().Message);
    Server.ClearError();
}
```

頁面層級錯誤

若要針對某一個網頁的所有程式進行統一的錯誤處理，您可以省略在程式中撰寫 Try...Catch 程式區段捕捉錯誤。當網頁程式執行發生例外錯誤時，則會觸發 Page 物件的 Error 事件。您可以在網頁中撰寫錯誤處理常式來處理這些例外。

在網頁 Error 事件的 Event Handler，統一處理表單中的例外狀況的範例如下：

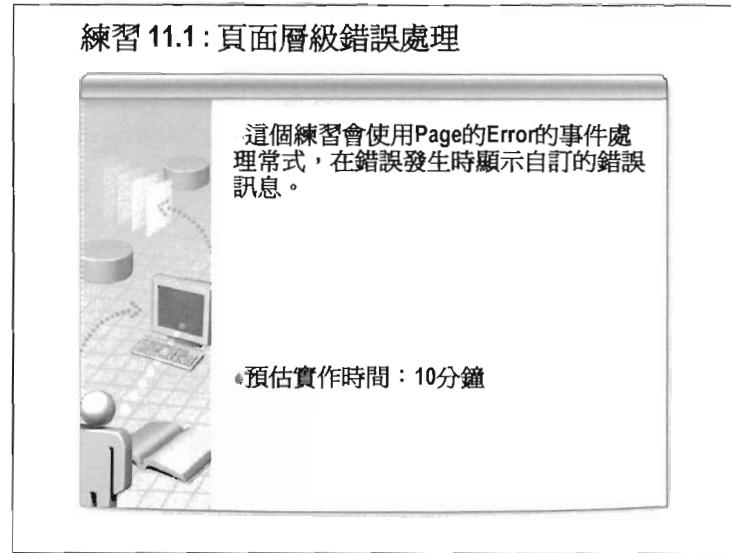
Visual Basic:

```
Protected Sub Page_Error(ByVal sender As Object, ByVal e As EventArgs)
    Server.Transfer("GenErr.aspx?err=" & Server.GetLastError().Message)
    Server.ClearError()
End Sub
```

C#:

```
protected void Page_Error(object sender, EventArgs e)
{
    Server.Transfer("GenErr.aspx?err=" + Server.GetLastError().Message);
    Server.ClearError();
}
```

您可以利用 Server 物件的 GetLastErrors 方法取得例外錯誤物件，在事件處理常式中，您可以利用 Server.Transfer 方法將使用者導向其它自訂的錯誤訊息網頁。一旦錯誤處理完畢，你應該呼叫 Server.ClearError 方法清除錯誤。



練習 11.1：頁面層級錯誤處理

目的：

這個練習主要是讓您熟悉如何針對網站中單一網頁進行例外錯誤處理，當網頁發生例外錯誤時，顯示錯誤資訊。

功能描述：

這個練習會使用 Page 的 Error 的事件處理常式，在錯誤發生時顯示自訂的錯誤訊息。

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「\U9543\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. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，新增一個網頁，命名為 testError.aspx。
4. 在 testError.aspx 的 Page_Load 事件處理常式輸入以下程式碼，將錯誤的日期資訊強制轉型成 DateTime 型別：

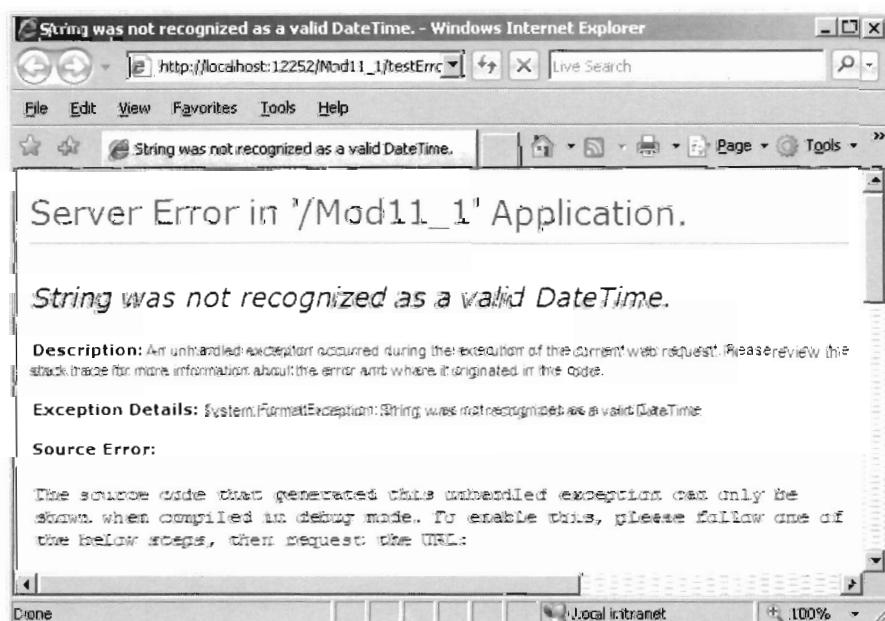
Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim s As String = "2009/2/30"
    Dim d As DateTime = DateTime.Parse(s)
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    String s = "2009/2/30";
    DateTime d = DateTime.Parse(s);
}
```

5. 在「Solution Explorer」→點選 UsePageError.aspx 網頁→按滑鼠右鍵→選「View In Browser」，當網頁執行時，會發現 ASP.NET 預設的錯誤頁面：



6. 關閉瀏覽器。回到程式視窗。
7. 自主選單「Web Site」下「Add New Item...」，選取「HTML Page」，新增一個 HTML 網頁，命名為 Err.htm。
8. 在 Err.htm< body>區塊下，加入錯誤訊息如下：

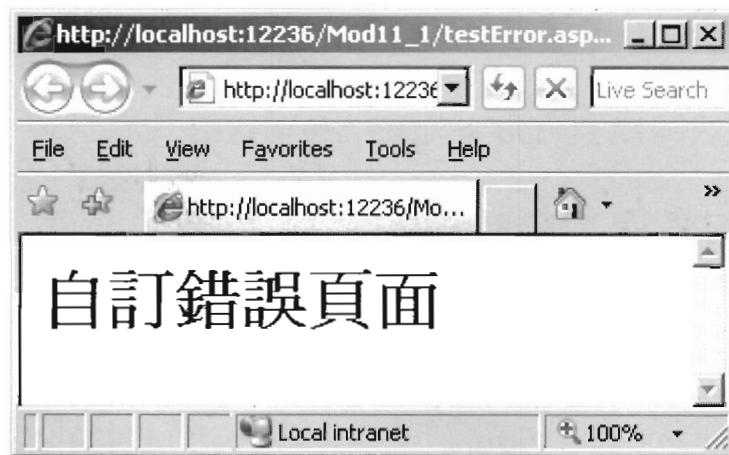
```
<h1>自訂錯誤頁面</h1>
```

9. 在程式碼視窗中，從「Server Objects & Events」下拉選單中，選擇 Page 物件，再從右邊的事件下拉選單中，選取 Error 事件，加入 Error 事件的錯誤處理常式，然後輸入以下程式碼，發生錯誤後，轉向 Err.htm 顯示錯誤訊息，並清除錯誤：

```
Visual Basic
Protected Sub Page_Error(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim ex As Exception
    ex = Server.GetLastError()
    Server.Transfer("Err.htm")
    Server.ClearError()
End Sub
```

```
C#
protected void Page_Error(object sender, EventArgs e)
{
    Exception ex = default(Exception);
    ex = Server.GetLastError();
    Server.Transfer("Err.htm");
    Server.ClearError();
}
```

10. 執行網頁測試。在「Solution Explorer」→點選 TestError1.aspx 網頁→按滑鼠右鍵→選「View In Browser」。當程式執行發生錯誤時，將顯示 Err.htm。



應用程式層級錯誤

- 擷取 Global.asax 檔中的 Application_Error 事件
- 多個網頁發生例外錯誤可共用事件處理常式進行錯誤處理

```

Visual Basic
Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
    Dim ex As Exception = Server.GetLastError()
    Response.Write(ex.Message)
    Server.ClearError()
End Sub

C#
void Application_Error(object sender, EventArgs e) {
    Exception ex = Server.GetLastError();
    Response.Write(ex.Message);
    Server.ClearError();
}

```

應用程式層級錯誤

Page 的 Error 事件只能針對特定的網頁來處理錯誤，若網站中有多個網頁發生例外錯誤而想共用事件處理常式進行錯誤處理時，您可以選擇使用應用程式層級錯誤來解決。

當在 ASP.NET 應用程式裡的任何一個表單，有未被處理的例外狀況時，這個例外狀況最後會被 Application 物件擷取，觸發 Application 物件的 Error 事件。這樣就不用在每一個網頁程式中加入錯誤處理的程式碼，便可以利用這一個事件處理常式來處理所有程式中的例外狀況。

Global.asax 檔案

Application 物件的 Error 事件處理常式需撰寫在 Global.asax 檔案。預設在 Visual Studio 建立 ASP.NET 網站並不會建立 Global.asax 檔案，您需要手動加入它。然後建立 Application_Error 事件處理常式，接著使用 Server 物件，取出導致 Web Request 物件中止執行的 Exception，並透過自訂的邏輯處理錯誤：

Visual Basic

```
Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
    Dim ex As Exception = Server.GetLastError()
    Response.Write(ex.Message)
    Server.ClearError()
End Sub
```

```
C#
void Application_Error(object sender, EventArgs e)
{
    Exception ex = Server.GetLastError();
    Response.Write(ex.Message);
    Server.ClearError();
}
```

需要注意的是，當你在 Application 物件的 Error 事件中處理過例外狀況後，仍然呼叫 Server.ClearError 指令，清除 Exception，否則 ASP.NET 執行環境仍會中斷程式執行。



練習 11.2: 應用程式層級錯誤處理

目的：

了解如何處理應用程式層級的錯誤，以便針對整個網站程式的例外錯誤做統一處理。

功能描述：

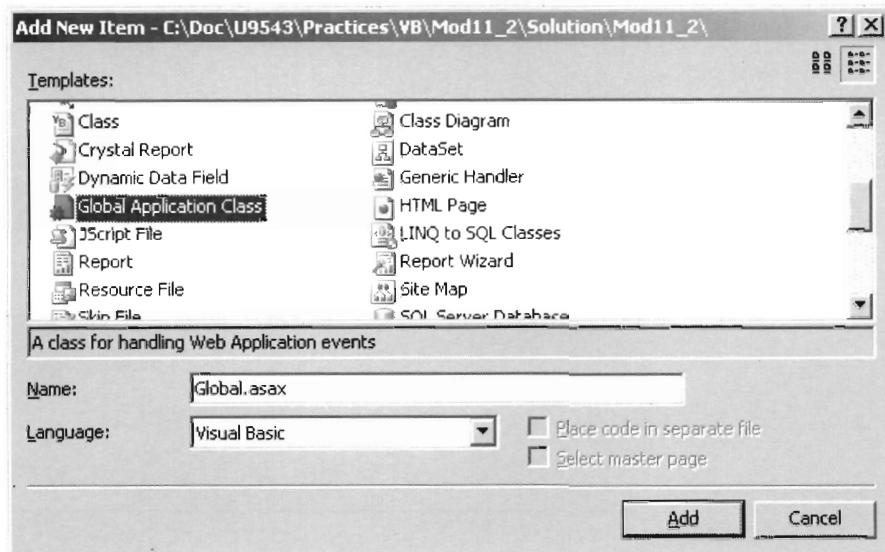
這個練習將會在 Global.asax 檔案中撰寫 Application 的 Error 的事件處理常式，並在錯誤發生時導向 Err.htm 以顯示自訂的錯誤訊息。

預估實作時間：5分鐘

實作步驟：

11. 延續上一個練習開啓上一個練習 11.1 完成的網站，或從 Visual Studio 「File」 → 「New Web Site」 → 選取「ASP.NET Web Site」 → 將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9543\Practices\VB 或 CS\Mod11_1\Starter\Mod11_1」目錄。
12. 在「Solution Explorer」視窗中 → 點選 WebSite → 按滑鼠右鍵 → 選「Add New Item」。

13. 自主選單「Web Site」下「Add New Item...」，選取「Global Application Class」，新增一個 Global.asax 檔案。



14. 將前一個練習中，testError.aspx 中 Page_Error 的事件處理常式中的程式碼移動到，Global.asax 檔案 Application_Error 區段，利用 Server.GetLastError 取得發生的例外錯誤，加入自訂的錯誤處理程式，程式碼看起來如：

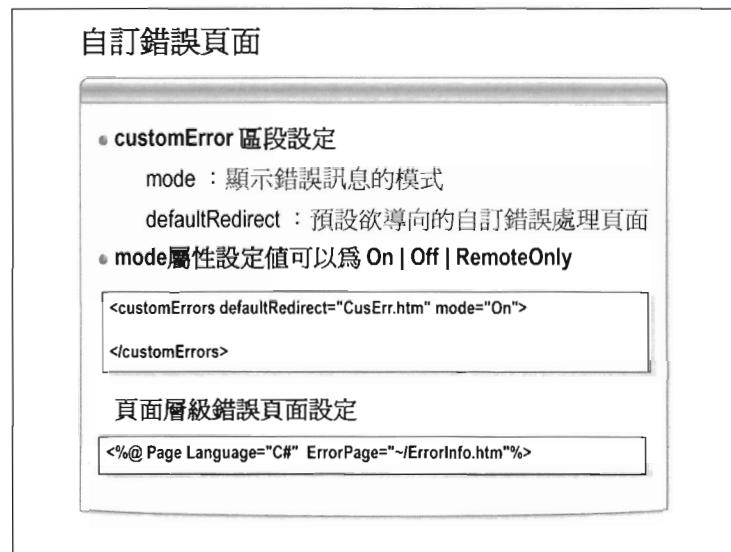
```
Visual Basic
Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs)
    Dim ex As Exception
    ex = Server.GetLastError()
    Server.Transfer("Err.htm")
    Server.ClearError()
End Sub
```

```
C#
void Application_Error(object sender, EventArgs e)
{
    Exception ex = default(Exception);
    ex = Server.GetLastError();
    Server.Transfer("Err.htm");
    Server.ClearError();

}
```

15. 測試網頁執行結果，此時瀏覽器中應顯示自訂的錯誤訊息。

使用組態設定
進行錯誤處理



自訂錯誤頁面

當 ASP.NET 網頁未撰寫例外處理程式時，ASP.NET 預設會產生一個黃底的錯誤頁面，若不想讓使用者檢視到此頁面，您可以在網站中的 Web.config 檔使用`<customErrors>`標籤指定發生錯誤時要導向哪一個頁面來顯示錯誤資訊。

`customError` 標籤提供兩個屬性，可設定錯誤處理的模式：

- mode : 必要屬性。用來設定顯示錯誤訊息的模式，可設為以下值：
 - ◆ On : 在本機或其他機器瀏覽器測試網頁出現錯誤時，會展現自訂導向的錯誤處理頁面或是預設的錯誤頁面。
 - ◆ Off : 在本機或其他機器瀏覽器測試網頁出現錯誤時，會展現詳細錯誤說明的錯誤頁面。
 - ◆ RemoteOnly : 如果在本機瀏覽器測試，網頁出現錯誤時會展現詳細錯誤說明的錯誤頁面；在其他機器瀏覽器測試時會展現自訂導向的錯誤處理頁面或是預設的錯誤頁面。

- `defaultRedirect`：非必要屬性，用來指定預設欲導向的自訂錯誤處理頁面。

如果在應用程式中，有一個自訂的錯誤頁面名為 `CusErr.htm`，你就可以在下面的設定將其設為預設的錯誤頁面：

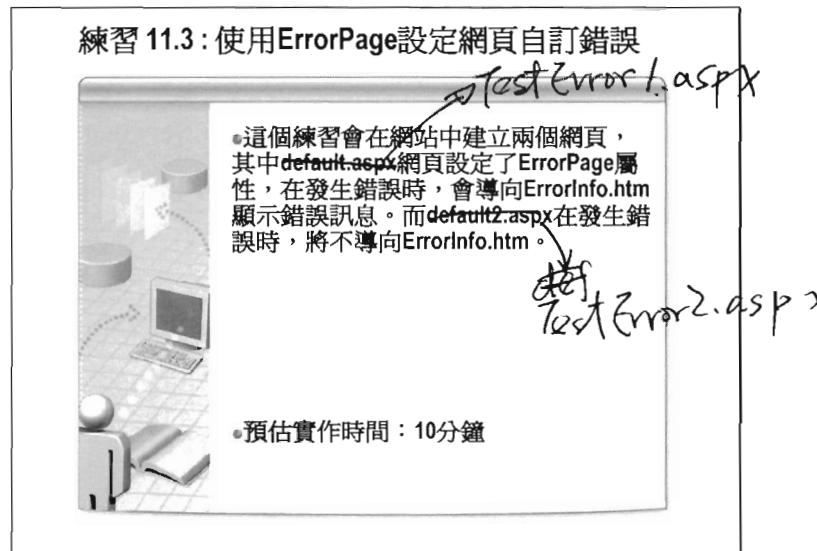
```
<customErrors defaultRedirect="CusErr.htm" mode="On">  
</customErrors>
```

一般情況下自訂的錯誤頁面都是使用 HTML，以避免錯誤頁面中的程式又發生錯誤而導致遞迴的錯誤。當你使用到 `defaultRedirect` 自訂錯誤頁面時，瀏覽器會收到一個 HTTP 302 狀態碼，以便重新發出一個 Request 轉向自訂錯誤頁面。這代表的含意是，原始網頁發生的例外錯誤資訊將會遺失，若使用 `Server.GetLastError` 都會取回 Null (或 Nothing)。

若自訂的錯誤頁面是一個 ASP.NET 網頁，您可以在網頁程式碼中，使用 `AspxErrorPath` 查詢字串得知是發生錯誤的網頁檔名：

```
Visual Basic  
Dim errPath As Object = Request.QueryString("AspxErrorPath")  
Label1.Text = errPath.ToString()
```

```
C#  
object errPath = Request.QueryString["AspxErrorPath"];  
Label1.Text = errPath.ToString();
```



練習 11.3: 使用 ErrorPage 設定網頁自訂錯誤

目的：

這個練習主要是讓您熟悉如何針對一個特定網頁的例外錯誤進行處理，當特定網頁生生例外錯誤時，將導向自訂的錯誤頁面顯示資訊。

功能描述：

這個練習會在網站中建立兩個網頁，其中 default.aspx 網頁設定了 ErrorPage 屬性，在發生錯誤時，會導向 ErrorInfo.htm 顯示錯誤訊息。而 default2.aspx 在發生錯誤時，將不導向 ErrorInfo.htm。

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」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選

取「\U9543\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「Mod11_3」。

3. 自主選單「Web Site」下「Add New Item...」，選取「HTML Page」，新增一個 HTML 網頁，命名為 ErrorInfo.htm。
4. 在 ErrorInfo.htm <body> 區塊下，加入錯誤訊息如下：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>
</head>
<body>
<h1>自訂錯誤頁面</h1>
網站發生錯誤!
</body>
</html>
```

5. 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 Webform 網頁，命名為 TestError1.aspx。
6. 在 TestError1.aspx 網頁 Page_Load 事件處理常式中，加入以下程式碼，產生例外錯誤：

Visual Basic

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim i As Integer = 0
    Dim j As Integer = 10
    Dim k As Integer = j / i
End Sub
```

C#

```
protected void Page_Load(object sender, EventArgs e)
{
    int i = 0;
    int j = 10;
    int k = j / i;
}
```

7. 在 TestError1.aspx 網頁 Page 指示詞中，設定 ErrorCode 屬性，指向 ErrorInfo.htm。

Visual Basic

```
<%@ Page Language="VB" ErrorPage("~/ErrorInfo.htm") %>
```

C#

```
<%@ Page Language="C#" ErrorPage "~/ErrorInfo.htm" %>
```

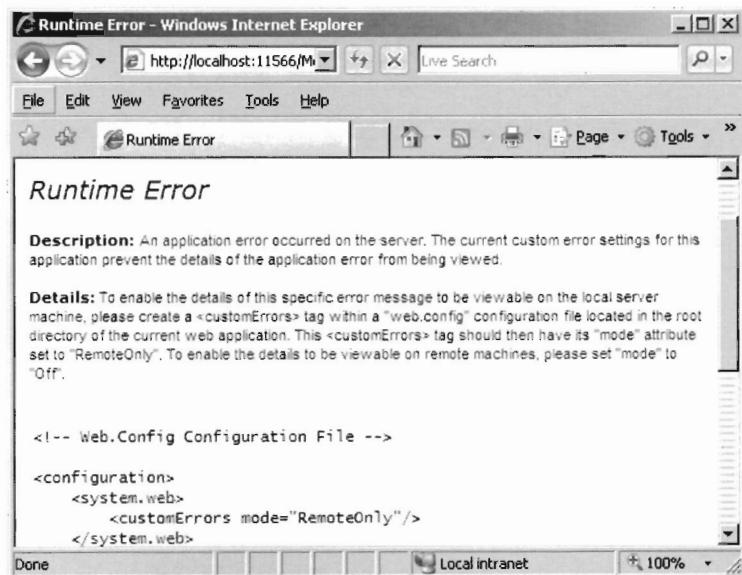
8. 在 Web.config 檔案中，<system.web> 區段中設定 customErrors 的 mode 屬性為 On :

```
<system.web>
<customErrors mode="On" />
</system.web>
```

9. 執行網頁測試。在「Solution Explorer」→點選 TestError1.aspx 網頁 → 按滑鼠右鍵 → 選「View In Browser」。當程式執行發生錯誤時，將顯示 ErrorInfo.htm :



10. 在網站中加入第二個網頁。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 Webform 網頁，命名為 TestError2.aspx。
11. 複製 TestError1.aspx 網頁 Page_Load 事件處理常式中的程式碼到 TestError2.aspx，以產生例外錯誤。
12. 執行網頁測試。在「Solution Explorer」→點選 TestError2.aspx 網頁 → 按滑鼠右鍵 → 選「View In Browser」。當程式執行發生錯誤時，將顯示預設錯誤頁面。



練習 11.4: 應用程式層級與自訂錯誤



練習 11.4: 應用程式層級與自訂錯誤

目的：

這個練習主要是讓您熟悉如何針對網站應用程式的所有網頁統一印行例外錯誤處理，當網頁產生例外錯誤時，將導向自訂的錯誤頁面顯示資訊，並將錯誤發生時間與訊息記錄到文字檔中。

功能描述：

這個練習會使用 Web.config 檔案設定啓用自訂錯誤，以及預設要導向的錯誤頁面(ErrInfo.aspx 網頁)。並搭配 Application_Error 事件，利用 FileStream 物件與 StreamWriter 將錯誤發生時間與訊息記錄到 log.txt 文字檔。

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」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「File System」並點選「Browse...」按鈕選取「\U9543\Practices\VB 或 CS\Mod11_4\Starter」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod11_4」。

3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，新增一個網頁，命名為 ErrInfo.aspx。

4. 在 ErrInfo.aspx 網頁<form>標籤下，加入錯誤訊息如下：

```
<form id="form1" runat="server">
<div>
<h1>Custom Error</h1>
發生自訂錯誤
</div>
</form>
```

5. 自主選單「Web Site」下「Add New Item...」，選取「Global Application Class」，新增一個 Global.asax 檔案。

6. 在 Application_Error 區段加入以下程式碼，利用 Server.GetLastError 取得發生的例外錯誤，然後將錯誤訊息利用 FileStream 與 StreamWriter 物件寫入 log.txt 檔案中：

<pre>Visual Basic Sub Application_Error(ByVal sender As Object, ByVal e As EventArgs) Dim ex As Exception = Server.GetLastError() Dim logPath = Server.MapPath("log.txt") Dim fs As System.IO.FileStream If Not System.IO.File.Exists(logPath) Then fs = System.IO.File.Create(logPath) End If Dim sw As New System.IO.StreamWriter(fs) sw.WriteLine(System.DateTime.Now.ToString() + "," + ex.InnerException.Message) sw.Close() fs.Close() End Sub C# void Application_Error(object sender, EventArgs e) { Exception ex = Server.GetLastError(); var logPath = Server.MapPath("log.txt"); System.IO.FileStream fs = default(System.IO.FileStream); if (!System.IO.File.Exists(logPath)) {</pre>
--

```

        fs = System.IO.File.Create(logPath);
    }
    System.IO.StreamWriter sw = new System.IO.StreamWriter
(fs);
    sw.WriteLine(System.DateTime.Now.ToString() + "," + ex.In
nerException.Message);
    sw.Close();
    fs.Close();

}

```

7. 在 Web.config 檔案中，<system.web>區段中設定 customErrors 的 mode 屬性為 On；並設定 defaultRedirect 屬性：

```
<customErrors mode="On" defaultRedirect="ErrInfo.aspx">
</customErrors>
```

8. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，新增一個網頁，命名為 TestError.aspx。並在 Page_Load 事件處理常式中，丟出自訂例外錯誤：

```

Visual Basic
Protected Sub Page_Load(ByVal sender As Object, ByVal e As Syst
em.EventArgs)
    Throw New Exception("Custom Error")
End Sub

```

```

C#
protected void Page_Load(object sender, EventArgs e)
{
    throw new Exception("Custom Error");
}

```

9. 執行網頁測試。在「Solution Explorer」→點選 TestError.aspx 網頁→按滑鼠右鍵→選「View In Browser」。當程式執行發生錯誤時，將顯示 ErrInfo.aspx：



10. 檢視 log.txt 檔案，參考執行結果如下：

2008/12/9 下午 12:29:33,Custom Error

通用HTTP錯誤處理

- 選擇根據不同錯誤所產生的錯誤碼，來轉向特定的錯誤頁面

<customError>區塊中，加入<error>標籤

- error項目屬性

statusCode : HTTP 狀態碼

redirect : 欲導向的網頁

```
<customErrors mode="RemoteOnly"
defaultRedirect="GenericErrorPage.htm">
<error statusCode="403" redirect="NoAccess.htm" />
<error statusCode="404" redirect="FileNotFound.htm" />
</customErrors>
```

通用 HTTP 錯誤處理

一個使用 defaultRedirect 來設定的通用錯誤處理網頁沒有辦法針對不同的錯誤來顯示錯誤資訊，這是因為你無法利用 Server.GetLastErrorMessage 來取得錯誤的關係。還好在<customErrors>區塊中，可以選擇根據不同錯誤所產生的錯誤碼，來轉向特定的錯誤頁面。如此，當程式發生不同種類的錯誤時，使用者就會看到不同的錯誤頁面的訊息。

要設定不同例外狀況的錯誤頁面，只需要再<customError>區塊中，加入<error>標籤，並且設定 error 標籤的 statusCode 和 redirect 屬性就可以了：

```
<customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
<error statusCode="403" redirect="NoAccess.htm" />
<error statusCode="404" redirect="FileNotFound.htm" />
</customErrors>
```

這樣將來當 403 錯誤發生時，使用者就會看到 NoAccess.htm 錯誤頁面中的訊息；當 404 錯誤發生時，使用者就會看到 FileNotFound.htm 錯誤頁面中的訊息。

總結

- 了解執行時期錯誤
- 網頁層級錯誤處理
- 應用程式層級錯誤處理
- 通用HTTP錯誤處理

第十二章: 網站發行與部 署

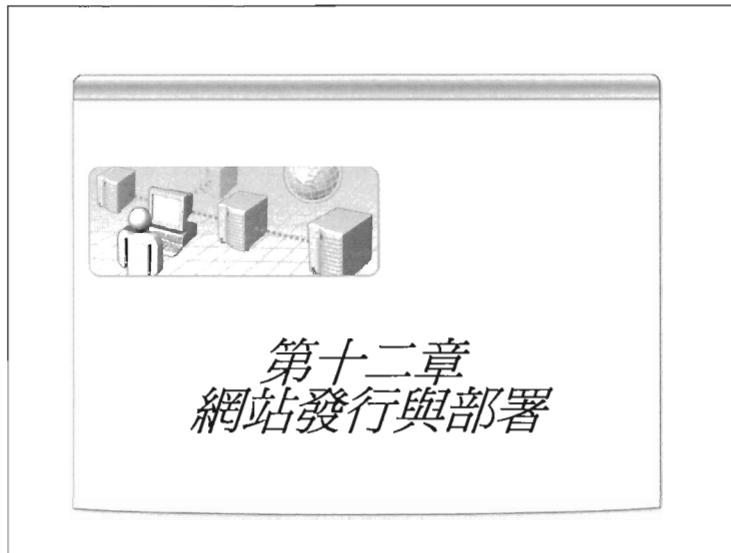
本章大綱

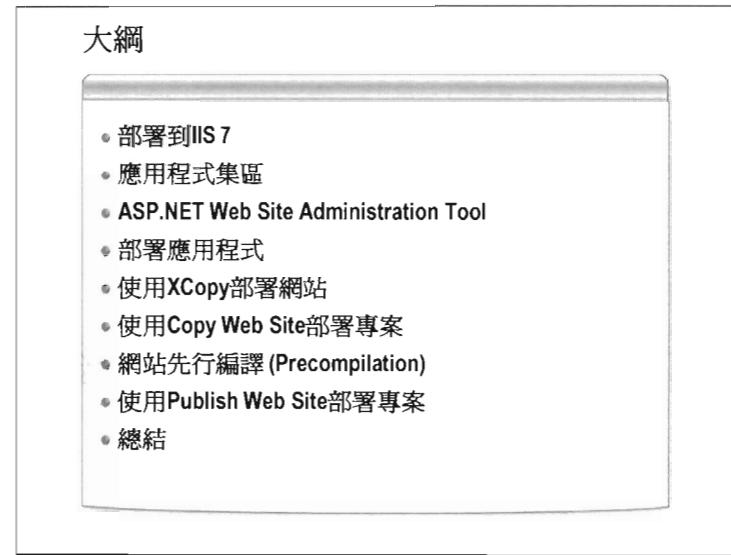
課程大綱.....	3
部署到 IIS 7	4
應用程式集區	7
網站組態檔階層.....	9
使用 IIS 管理工具修改 Web.config 檔.....	11
練習 12.1 : 使用 IIS 管理工具設定組態	15
ASP.NET Web Site Administration Tool.....	21
部署應用程式	22
使用 XCopy 部署網站	23
使用 Copy Web Site 部署專案	24
練習 12.2 : 使用 Copy Web Site 工具部署	26
網站先行編譯 (Precompilation).....	30
使用 Publish Web Site 部署專案	31
練習 12.3 : 使用 Publish Web Site 工具部署專案.....	33
建立應用程式安裝檔.....	37
練習 12.4 : 建立應用程式安裝檔.....	39

作者：

許薰尹



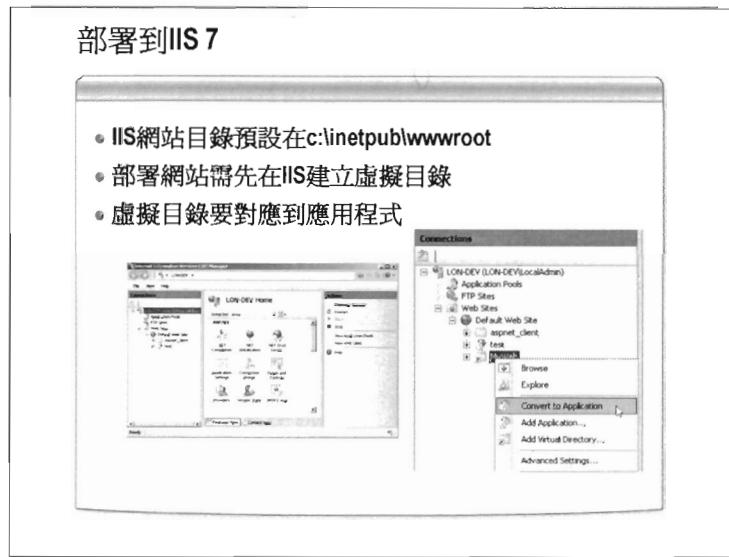




課程大綱

這個章節主要介紹該如何部署網站程式，以及部署的工具。本章將介紹以下主題：

- 部署到 IIS 7
- 應用程式集區
- ASP.NET Web Site Administration Tool
- 部署應用程式
- 使用 XCopy 部署網站
- 使用 Copy Web Site 部署專案
- 網站先行編譯 (Precompilation)
- 使用 Publish Web Site 部署專案

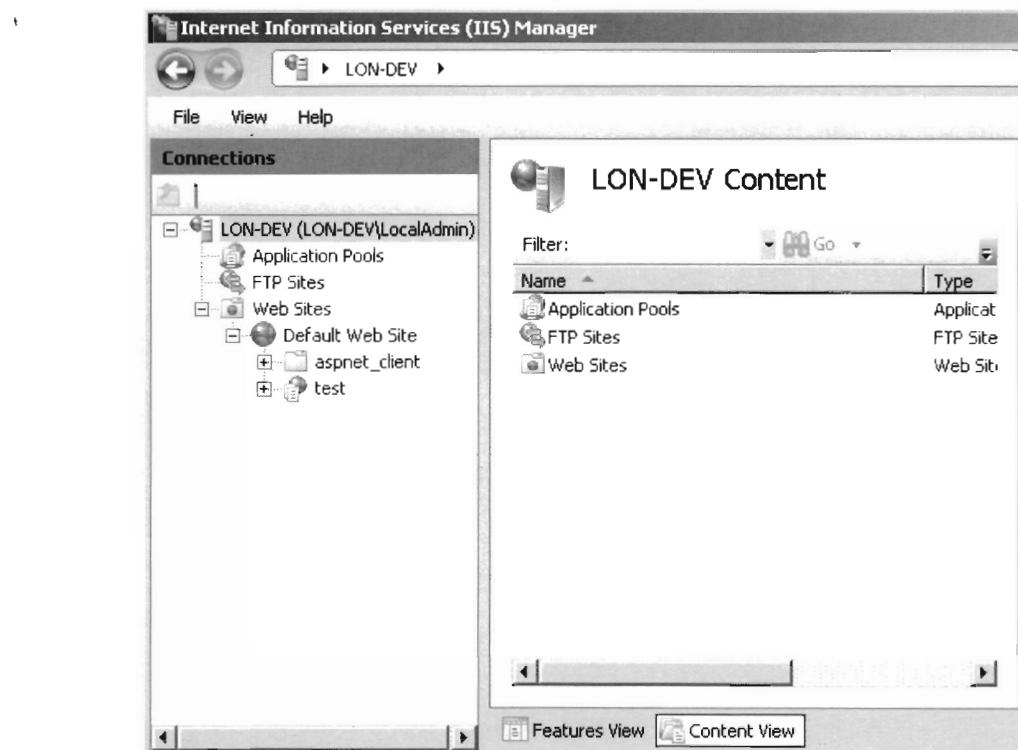


部署到 IIS 7

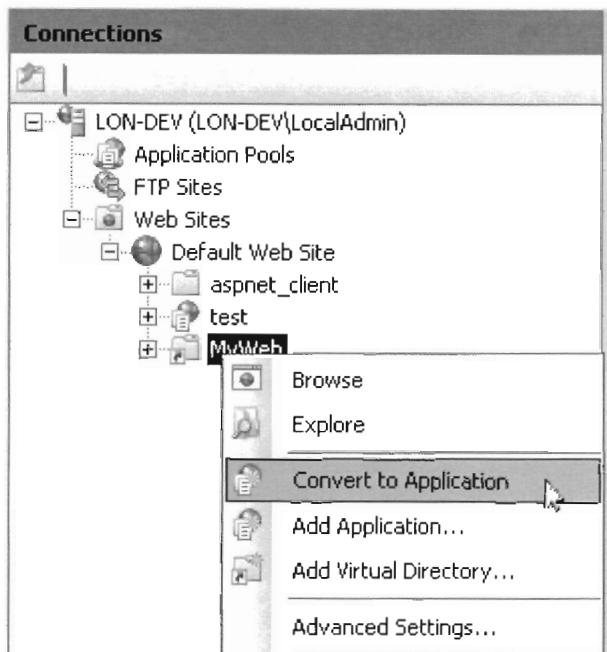
當主機安裝了 IIS，它預設會在 c:\inetpub\wwwroot 建立目錄，並產生一些檔案，這個目錄就是 IIS 預設網站的目錄。您可以在瀏覽器使用 <http://localhost/> 連結到 IIS 預設網站。

使用 IIS 7 管理工具

IIS7 管理工具介面如下圖，您可以從視窗下方點選「Content View 和 Features View」來進行檢視：



要將網站部署到 IIS 7 上，你需要先在 IIS 建立虛擬目錄。ASP.NET 網站應用程式的目錄可以是實體目錄也可以是虛擬目錄。通常利用 Visual Studio 在 IIS 上建立網站時，預設所建立的虛擬目錄就是應用程式目錄。你可以點選 IIS 7 管理工具 Connections 視窗下的虛擬目錄，然後按滑鼠右鍵，點選「Convert To Application」：



後續的章節中，將以應用程式(Application)通稱之。

若要在 ASP.NET 應用程式之中，判斷 IIS 的版本，可以利用如下程式碼：

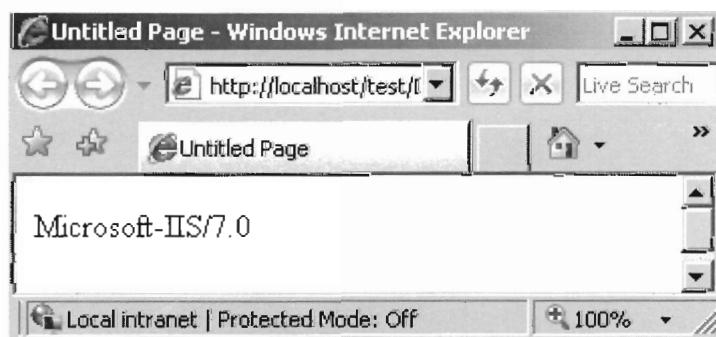
Visual Basic

```
Response.Write(Request.ServerVariables("SERVER_SOFTWARE"))
```

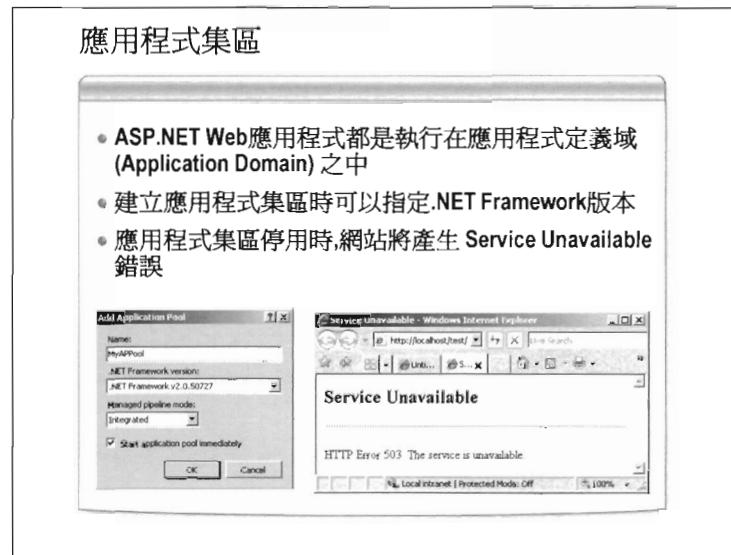
C#

```
Response.Write(Request.ServerVariables["SERVER_SOFTWARE"]);
```

例如下列是在 IIS 7 上的執行結果：



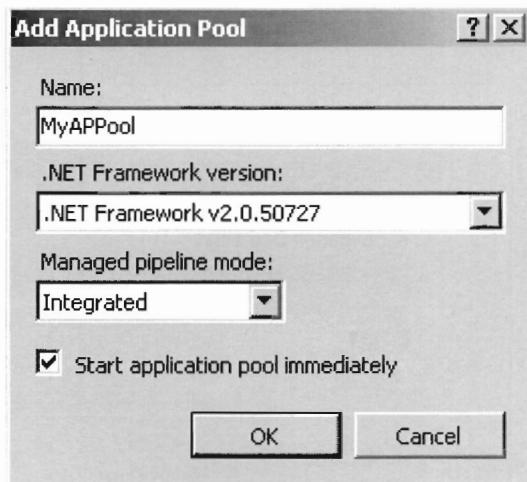
若使用 Visual Studio，以「File System」方式建立網站，利用測試的 Web 伺服器執行時，則會顯示空白。



應用程式集區

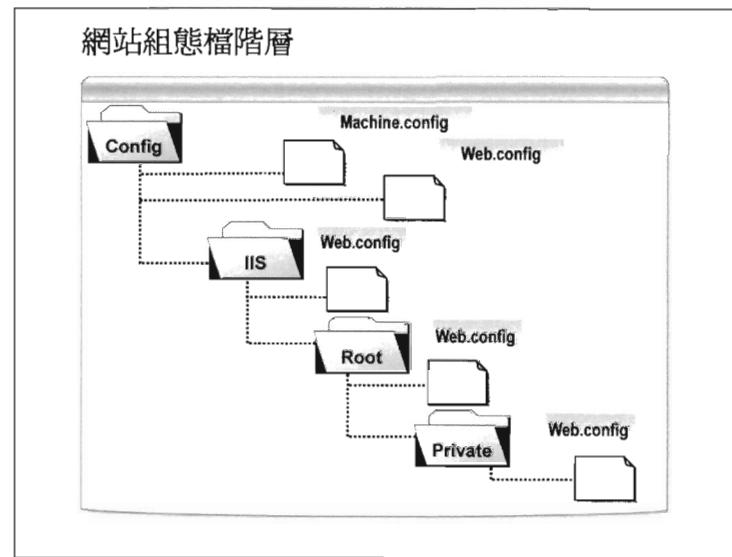
每個 ASP.NET Web 應用程式都是執行在自己的應用程式定義域 (Application Domain) 之中，而每個應用程式定義域都隸屬於某個應用程式集區 (Application Pool)，一個應用程式集區之中，可以包含多個 Web 應用程式。每個應用程式集區都相互隔離，其中一個執行發生錯誤，將不影響其它應用程式集區，以增加系統穩定性，但會影響到應用程式集區中的其它應用程式。

你可以在 IIS 7 中建立應用程式集區 (Application Pool)，您可以在 IIS 管理工具 Connection 區塊新增應用程式集區，然後設定要使用的.NET Framework 版本。



個別應用程式集區可以單獨啓動或停止，如果停止應用程式集區，又試圖去執行應用程式集區中的網站程式，則會得到一個 503 號錯誤，錯誤訊息為：「Service Unavailable」：





網站組態檔階層

ASP.NET 應用程式可以使用 XML 格式的組態檔 (*.config) 來設定網站的功能，如安全性或連接字串的設定。ASP.NET 可能會使用到以下幾個組態檔：

- 機器層級的 Machine.config：

Machine.config 的設定會套用到這台機器上所有的.NET 應用程式，Machine.config 檔位於
\Windows\Microsoft.NET\Framework\{version}\CONFIG 目錄路徑下。

- 機器層級的 Web.config：

機器層級 Web.config 的設定會套用到這台機器上面所有的 ASP.NET 網站應用程式，其檔案位於
\Windows\Microsoft.NET\Framework\{version}\CONFIG 目錄路徑下。

- 站台層級的 Web.config (選擇性)：

站台層級 Web.config 設定會套用至此站台下所有的 ASP.NET 網站應用程式。其組態檔位在每一個 Web Site 站台的根目錄下，例如，預設站台則存放於 C:\Inetpub\wwwroot 下。

- Web 應用程式層級的 Web.config(選擇性)：

Web 應用程式 Web.config 設定會套用到整個 Web 應用程式的檔案以及資料夾，其檔案位於 Web 應用程式的根目錄下。

- 資料夾層級的 Web.config(選擇性)：

此設定會套用到組態檔所在的資料夾下所有的網頁。

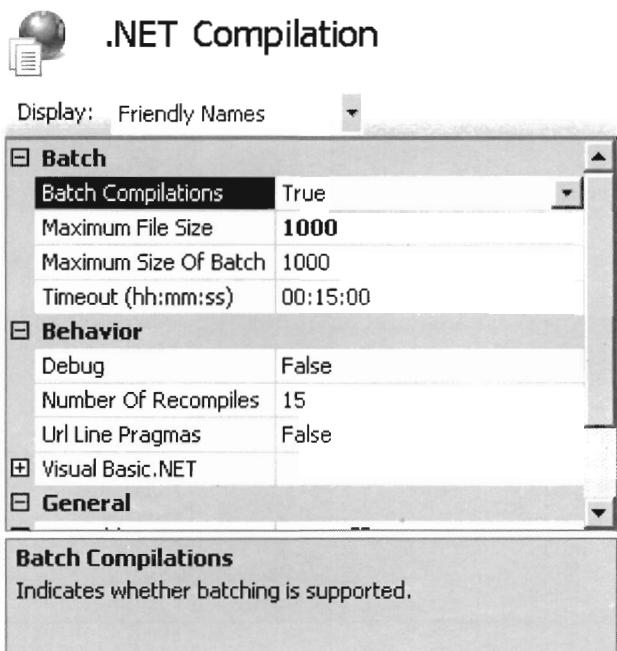


使用 IIS 管理工具修改 Web.config 檔

在 IIS 管理工具中，您可以利用 Features View 來變更網站 Web.config 檔案，或 machine.config 檔案的設定。您可以設定資料庫連接字串、編譯選項、應用程式設定、網站全球化...等資訊。以下介紹常用的設定。

Compilation 設定

在.NET Compilation 對話盒中所設定的資訊，會對應到組態檔案 `<compilation>` 區段的設定：



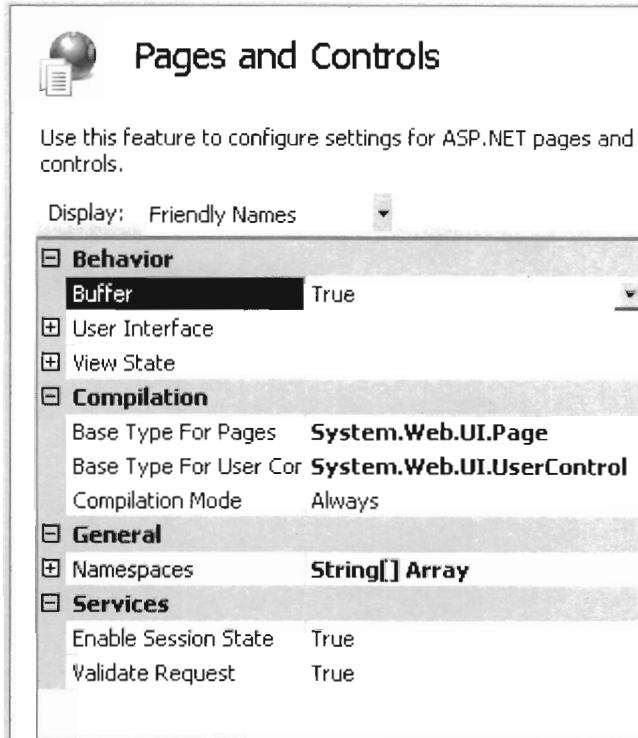
例如將 compilation 的 batch 設定為 true；將 debug 設定為 false，則在 Web.config 檔案中可以看到如下的設定：

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

batch 預設值為 false。若設定為 true，會將網站中所有尚未編譯過的檔案先進行編譯動作(Precompiled)。這樣的缺點是，當網站應用程式啓動時，啓動速度會變慢，但後續執行網站程式，就不會因進行編譯動作而有延遲現象發生。

Pages and Controls 設定

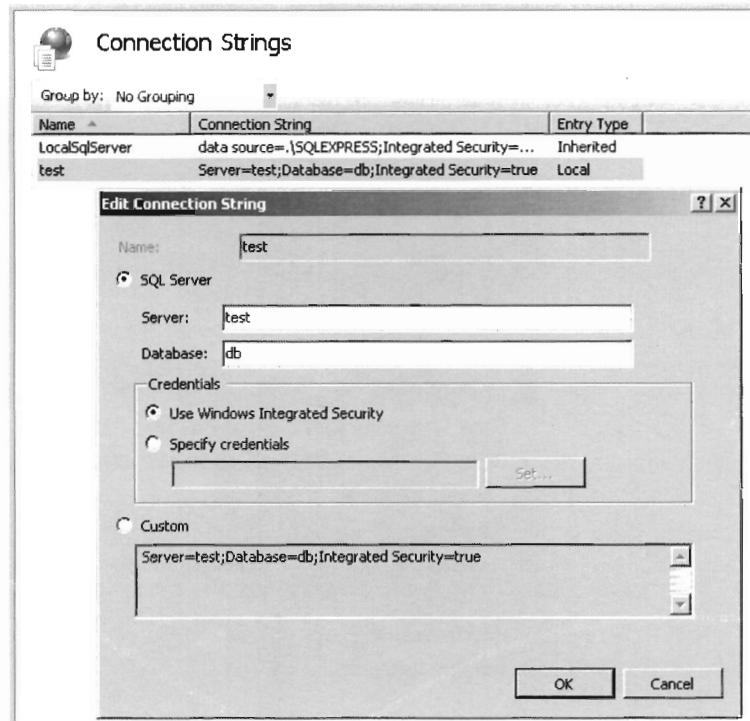
在 Pages and Controls 對話盒中所設定的資訊，會對應到組態檔案 <pages> 區段的設定。

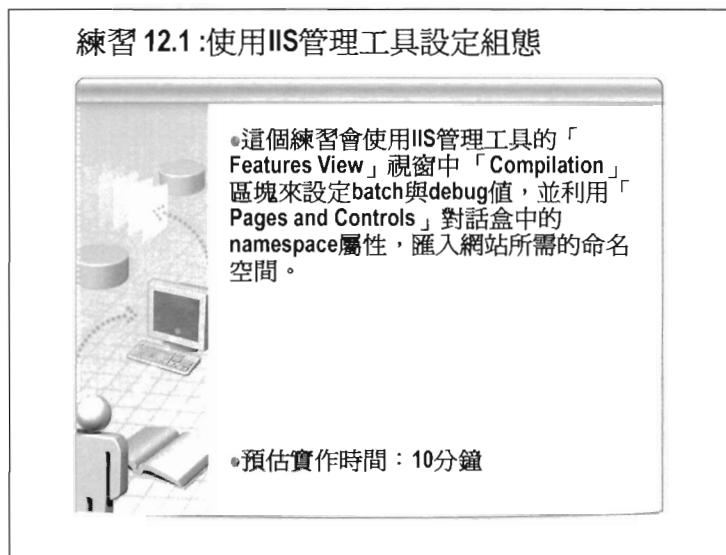


例如您可以編輯其中的 Namespace 區段，以設定網站程式中要匯入的命名空間。

Connection Strings 設定

在 Connection Strings 對話盒中所設定的資訊，會對應到組態檔案 <connectionStrings> 區段的設定，用來定義目前網站所可以使用到的資料庫連接字串：





練習 12.1 : 使用 IIS 管理工具設定組態

目的：

這個練習主要是讓您熟悉 IIS 管理工具的操作，利用管理工具來設定網站的組態檔案。

功能描述：

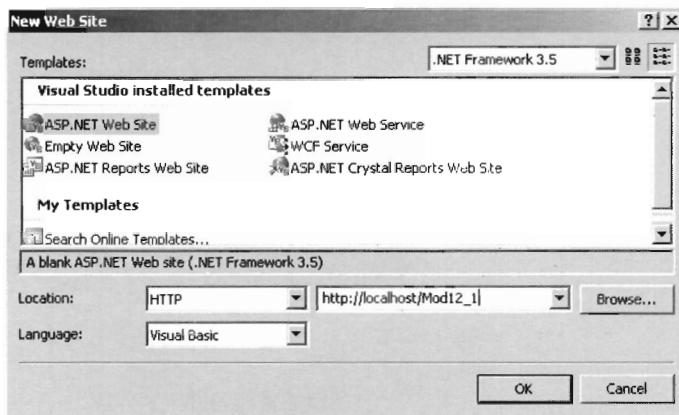
這個練習會使用 IIS 管理工具的「Features View」視窗中「Compilation」區塊來設定 batch 與 debug 值，並利用「Pages and Controls」對話盒中的 namespace 屬性，匯入網站所需的命名空間。

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『HTTP』，路徑設為

「http://localhost/Mod12_1」目錄，與使用的程式語言
(Language)，如 Visual Basic 或 C#。

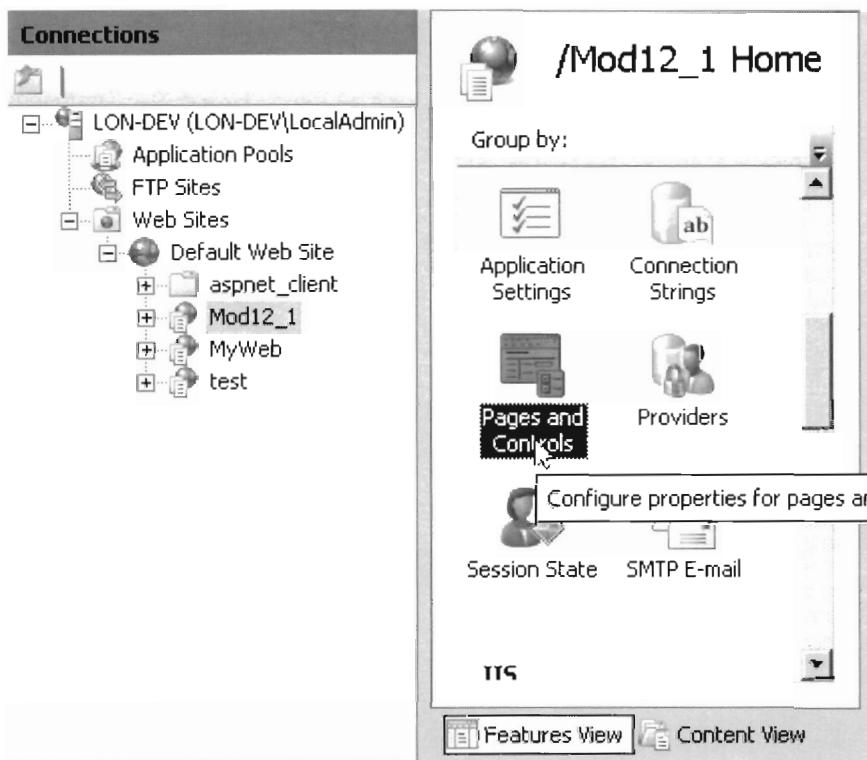


3. 自主選單「Web Site」下「Add New Item...」，選取「Web Form」，新增一個網頁，命名為 test.aspx。
4. 在 Page_Load 事件處理常式中加入以下程式碼：

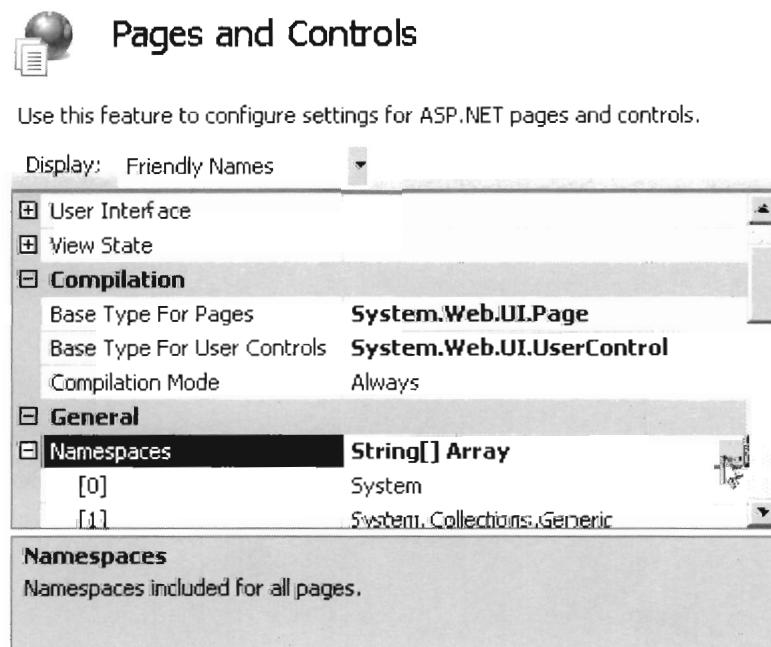
```
Visual Basic
Dim ar as new ArrayList
```

```
C#
ArrayList ar =new ArrayList();
```

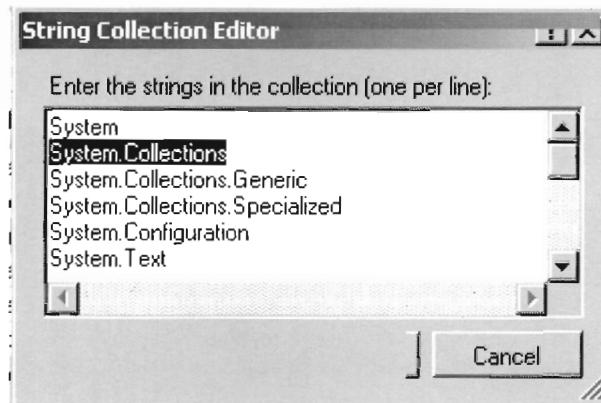
5. 在「Build」→點選「Build Page」，網頁可正確編譯，沒有錯誤訊息。
 6. 在作業系統命令提示字元輸入以下指令，啓動 IIS 管理工具：
- ```
inetmgr
```
7. 在「Connections」視窗中，找到 Mod12\_1 網站，雙擊中間視窗「Pages and Controls」圖示。



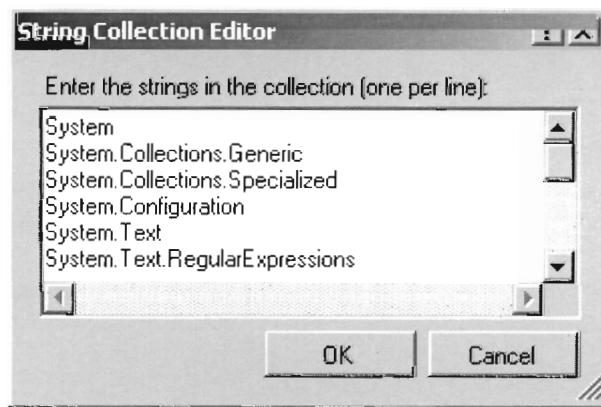
8. 點選 Namesapces 後方的按鈕，啓動「String Collection Editor」：



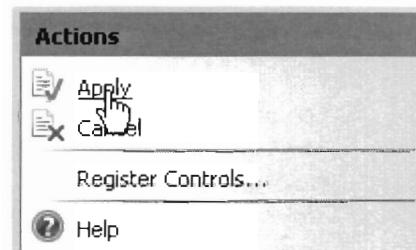
9. 找到 System.Collections 項目，將之刪除。



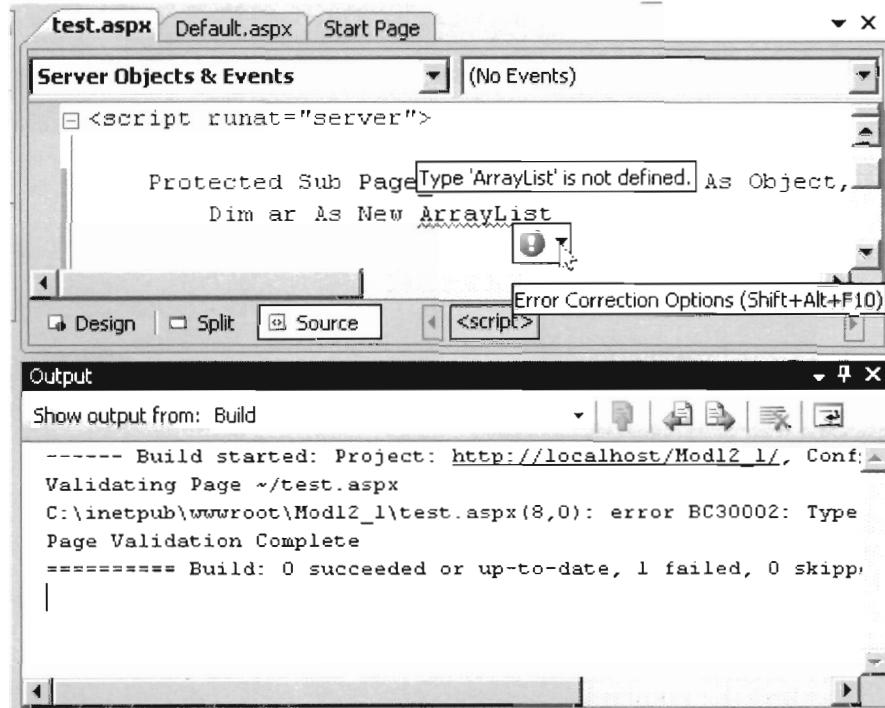
目前的畫面看起來如：



10. 按下「String Collection Editor」視窗的「OK」按鈕，然後在 IIS 管理工具「Action」視窗中點選「Apply」。



11. 回到 Visual Studio 工具。在「Build」→點選「Build Page」，網頁無法正確編譯，產生錯誤訊息。

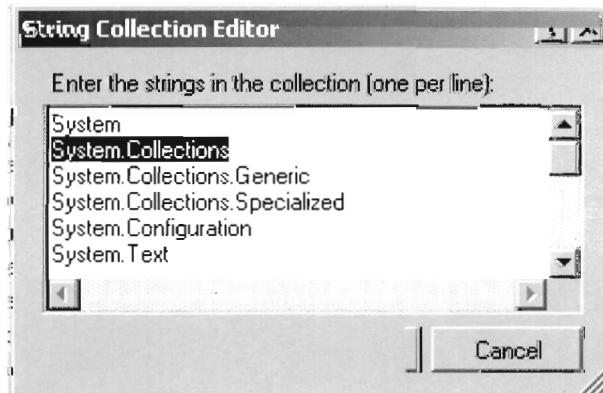


檢視 Web.config 檔案，pages 區段下將找不到以下粗體字這行  
匯入命名空間的設定：

```
<pages>
<namespaces>
<clear />
<add namespace="System" />
<add namespace="System.Collections" />
<add namespace="System.Collections.Generic" />
...

```

12. 參考步驟 8~11，重新啓動「String Collection Editor」，但這次要將 System.Collection 設定加回去後套用：



13. 回到 IIS 管理工具，雙擊 Mod12\_1 網站的「.NET Compilation」圖示將 Debug 設為 True；將 BatchCompilations 設為 false，然後在 IIS 管理工具「Action」視窗中點選「Apply」：



14. 回到 Visual Studio 工具。檢視 Web.config 檔案，compilation 區段如下：

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

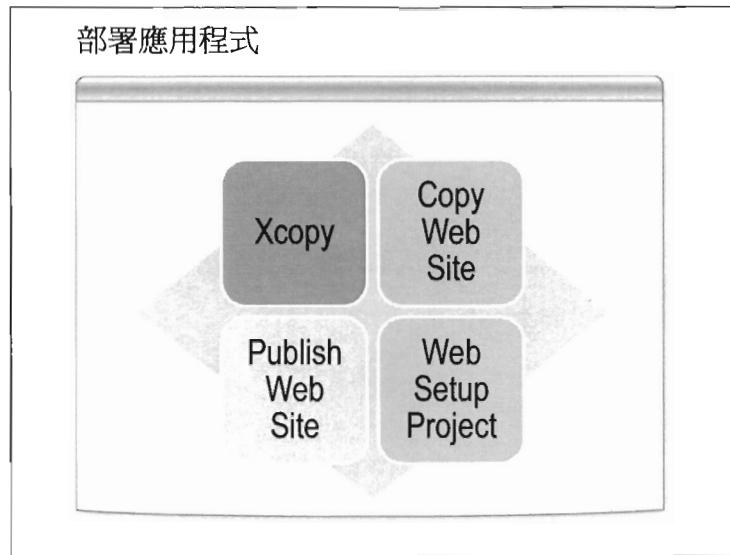


## ASP.NET Web Site Administration Tool

在 Visual Studio 開發工具中，另外提供了一個可以快速設定組態的工具—ASP.NET ASP.NET Web Site Administration Tool。你可以從 Visual Studio 開發工具的「Web Site」選單中，透過「ASP.NET Configuration」選項，開啓這一個工具。

ASP.NET Web Site Management Tool 提供 Web 操作介面，幫助使用者設定在 Web Site 根目錄底下的 Web.Config 檔的內容。所以你不需要記得 Web.Config 的標籤，也可以設定應用程式所需要的執行環境。這一個工具提供方便的操作介面幫助你設定：

- 安全性設定
- 使用者設定檔
- 應用程式設定
- Provider 設定



## 部署應用程式

當你在開發環境中將 ASP.NET 應用程式開發完成之後，最終要要將 ASP.NET 應用程式部署到 Web 伺服器上執行。在 Visual Studio 開發工具中，針對不同種類的 ASP.NET 應用程式提供了多種的部署方式的支援：

- Xcopy 指令
- Copy Web Site
- Publish Web Site
- Web Setup Project

你可以根據程式的需要選擇適當的方式部署應用程式。

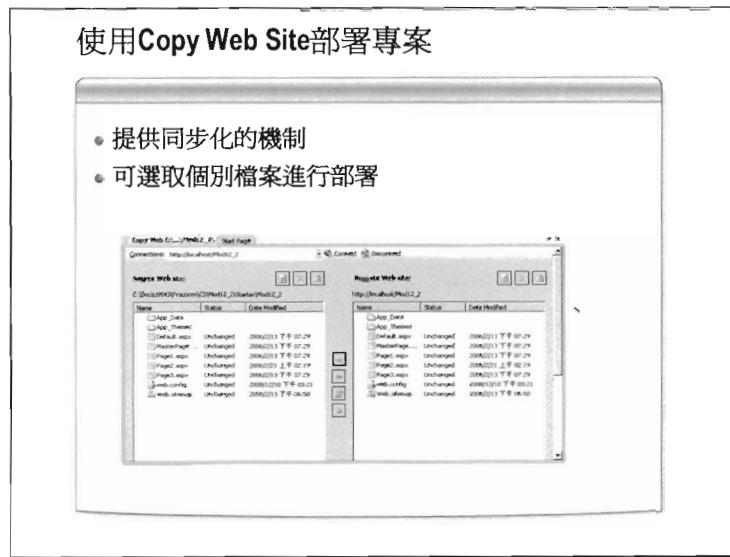
### 使用 XCopy 部署網站

- 用於簡單的應用程式部署
- 需要複製的檔案類型
  - aspx 檔案與 asmx 檔案
  - Global.asax
  - Web.config
  - bin 目錄下的 dll
  - 其他程式中會使用到的資源檔

### 使用 XCopy 部署網站

如果你所開發的 ASP.NET 應用程式中，並沒有需要特別的安裝共享元件或是資料庫等動作的話，你可以在開發完成後，直接將網站中的檔案到上線的 Web 伺服虛擬目錄中就可以了。

您可以使用 XCopy 指令來複製必要的檔案，包含\*.ASPX、Global.asax、文字檔、Web.config、bin 目錄下的 dll、圖片檔、聲音檔或網站使用到的樣式表。不過不是每一個檔案都是必需要複製的，像是網頁程式碼後置(Code-behind)的 \*.cs、\*.vb 檔，或是方案檔、資源檔等，就不是必要的。



## 使用 Copy Web Site 部署專案

當使用 Visual Studio 開發 ASP.NET 應用程式時，預先編譯與部署的動作就變得很容易，因為在 Visual Studio 開發工具中內建提供很多工具支援這些動作。在 Visual Studio 開發工具中，針對應用程式的部署提供了兩個工具：「Copy Web Site」與「Publish Web Site」。

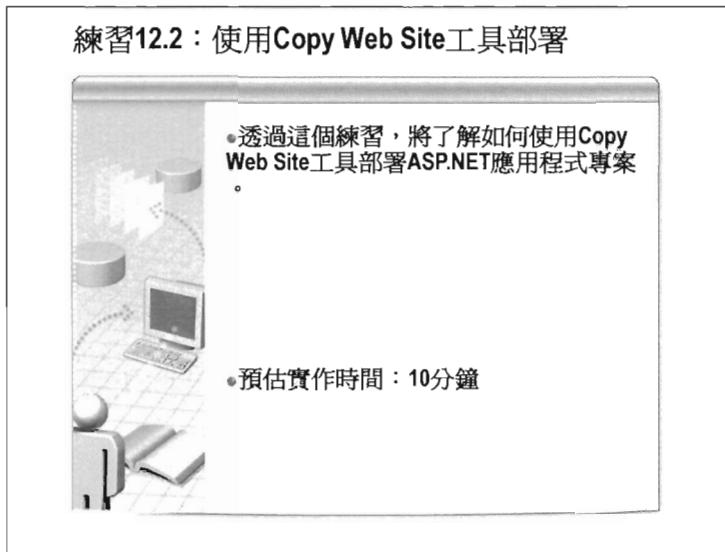
### Copy Web Site

「Copy Web Site」可以幫助你把 Visual Studio 專案中的檔案部署到指定的位置，你可以從「Web Site」選單中找到這一個工具。跟一般的 FTP 或是 XCopy 不一樣的地方就是有提供同步化的機制，會自動在複製時幫你檢查你所部署的資料是不是最新的版本：

同時，在工具裏也會顯示目前檔案的狀態，顯示在 Status 欄位之中：

- Unchanged：從上次更新到現在檔案沒修改過。
- Changed：從上次更新到現在檔案有異動過的紀錄。
- New：新加入的檔案。
- Deleted：已經移除的檔案。

這個工具還有一個有用的地方，就是當你無法使用 Visual Studio 開發工具直接修改遠端的應用程式時，可以先將遠端的 Web Site 複製回本機，修改完之後再複製回遠端的電腦中。您也可以選取個別檔案進行部署，而不用部署所有檔案。



## 練習 12.2：使用 Copy Web Site 工具部署

目的：

練習使用 Visual Studio 開發工具中的 Copy Web Site 功能，將網站應用程式部署到 IIS。

功能描述：

透過這個練習，將了解如何使用 Copy Web Site 工具部署 ASP.NET 應用程式專案。

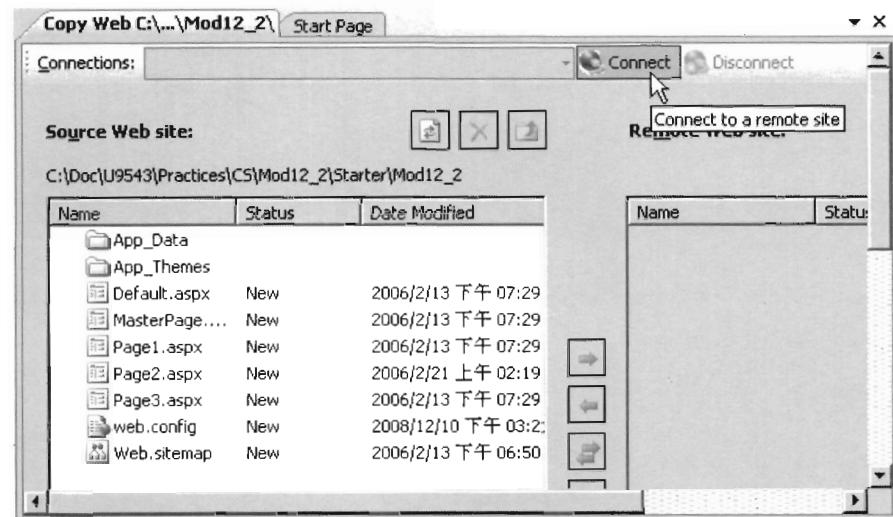
預估實作時間：10 分鐘

實作步驟：

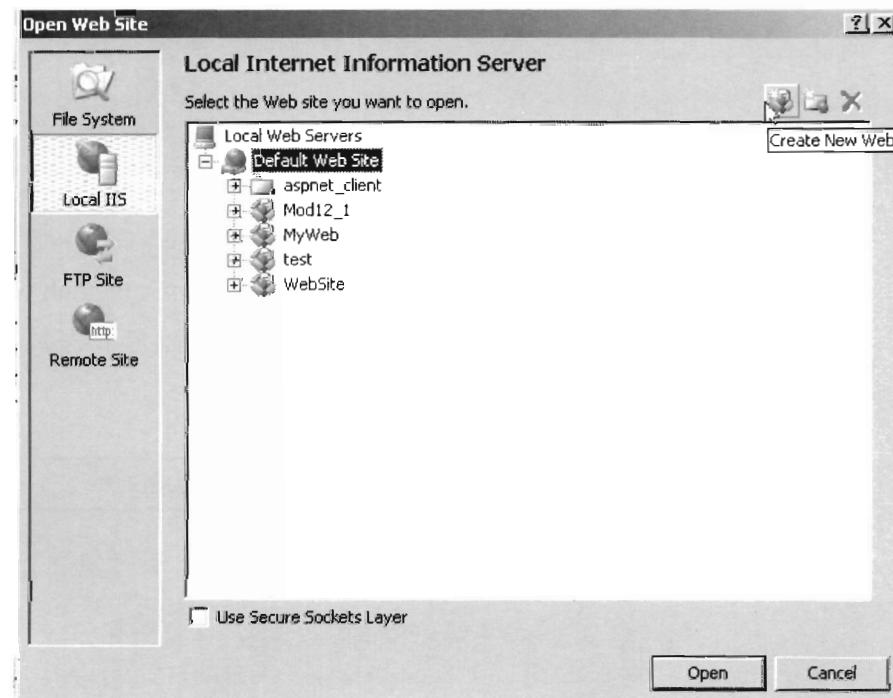
- 從「Start」→「Program」→「Microsoft Visual Studio 2008」→「Microsoft Visual Studio 2008」，啓動 Visual Studio 2008 開發環境。
- 從「File」→「Open Web Site」→選取「File System」→並選取「\U9543\Practices\VB 或 CSM\Mod12\_2\Starter\Mod12\_2」目錄開啓網站。

3. 在「Website」選單中，選取「Copy Web Site..」選項。

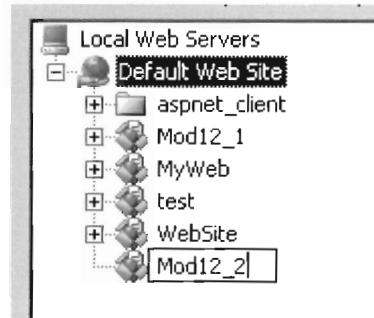
4. 從「Copy Web Site」工具中，選取「Connect」按鈕。



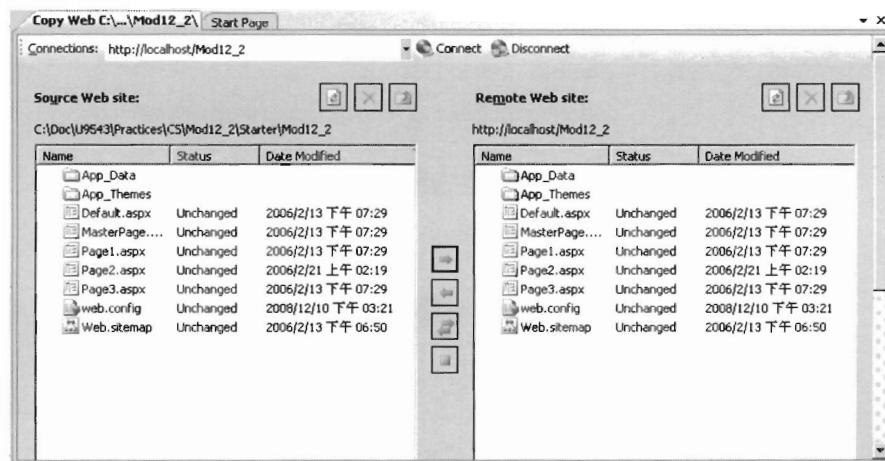
5. 在開啟的視窗中，選取「Local IIS」→「Default Web Site」→「Create New Web Application」。



6. 鍵入 Mod12\_2，按下「Open」按鈕。



7. 在「Copy Web Site」工具中，全選左視窗中所有檔案，按下 按鈕，將專案中所有檔案上載到 IIS 上的 Mod12\_2 應用程式目錄中。

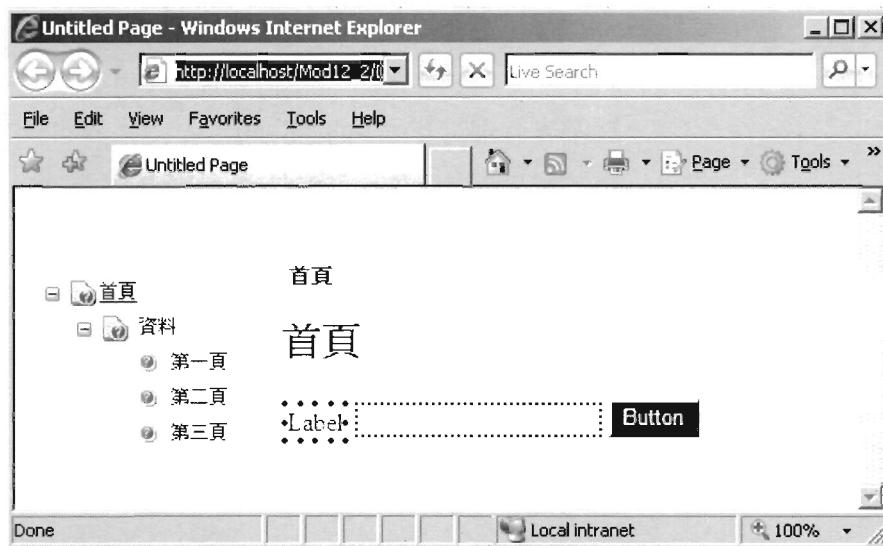


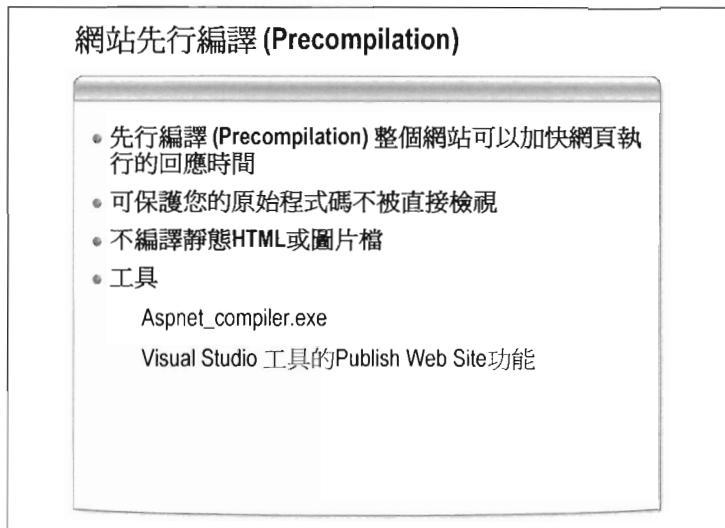
8. 完成後，關閉「Copy Web Site」工具。此時專案中所有檔案（含原始檔）就複製完成。檢視 C:\Inetpub\wwwroot 下是否正確建立 Mod12\_2 目錄，且所有檔案都正確複製到此目錄。

9. 開啓瀏覽器，在網址輸入

[http://localhost/Mod12\\_2/Default.aspx](http://localhost/Mod12_2/Default.aspx)

確認網站程式能夠正確執行，參考執行畫面如下：





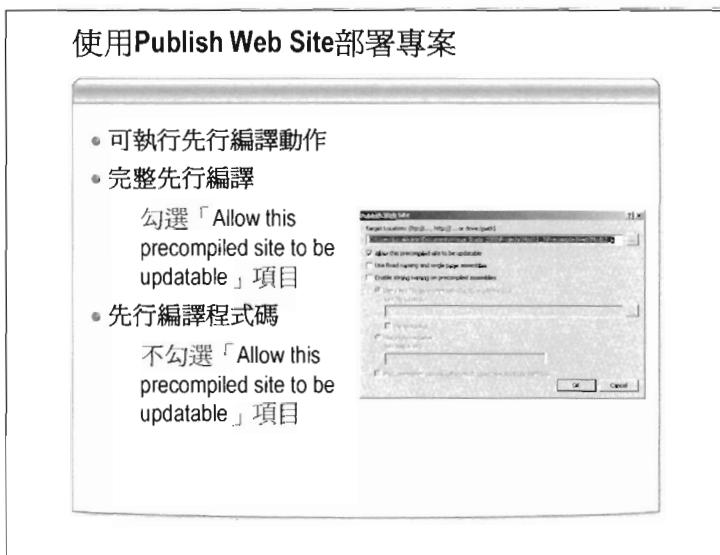
ASP.NET - merge .exe 將整個網站  
~~合併成一個 exe 檔~~  
 網站先行編譯 (Precompilation)

預設使用 Web Site 方式建立 ASP.NET 網站專案，網頁是在使用者第一次存取時，進行編譯動作，將編譯出的組件置入快取(Cache)。後續執行到相同網頁，就從快取中載入組件執行。

ASP.NET 提供先行編譯 (Precompilation) 整個網站的機制，以加快網頁執行的回應時間。因此網站要部署時，應該要先進行先行編譯的動作，以提升執行的效能。此外，也可以保護您的原始程式碼不被直接檢視。

先行編譯 (Precompilation)只會對網站中的變動的程式檔案(\*.vb、\*.cs 其他程式檔)以及資源檔有用，但不影響靜態 HTML 或圖片檔。每個 aspx 網頁會有一個對應的 .compiled 檔，其中包含網頁對應的組件檔資訊。

您可以在命令提示字元使用 Aspnet\_compiler.exe 工具，或透過 Visual Studio 工具 Publish Web Site 功能，執行網站先行編譯 動作。

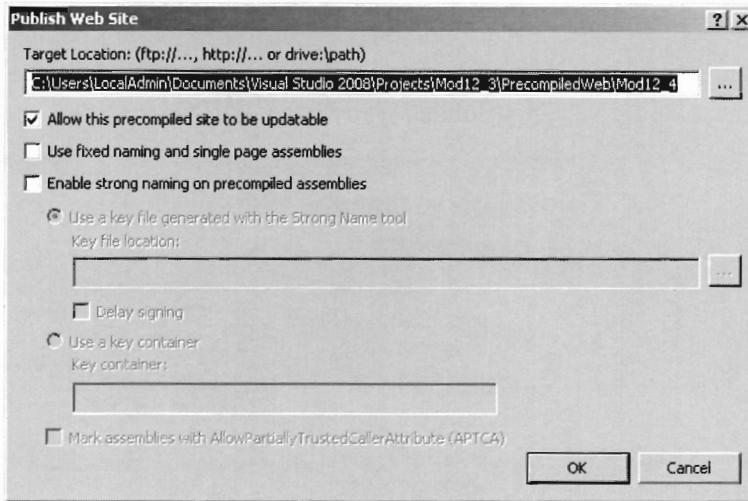


## 使用 Publish Web Site 部署專案

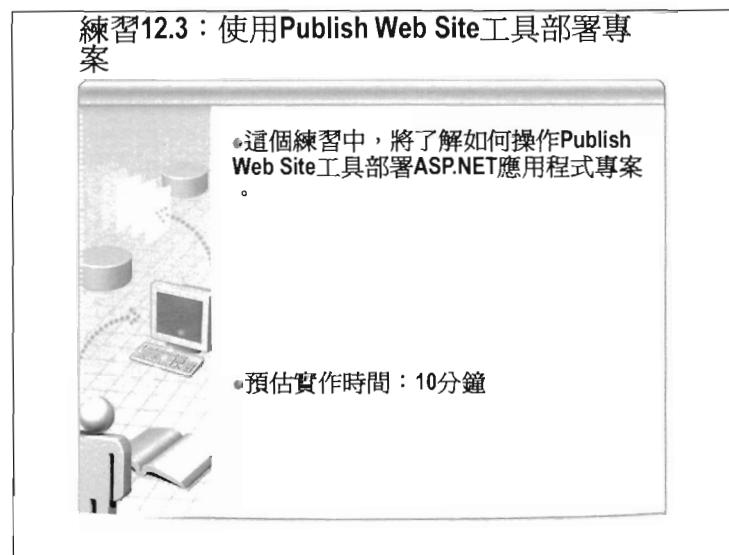
如果是要將 ASP.NET 應用程式預先編譯過，再進行部署的話，可以使用 Visual Studio 所提供的 - 「Publish Web Site」功能，利用圖型介面進行操作。

### Publish Web Site

「Publish Web Site」和「Copy Web Site」工具不一樣的地方，在於「Publish Web Site」會執行預先編譯動作之後，再根據你所指定的方式將編譯後的結果傳送到另一台伺服器；同時「Publish Web Site」不會執行同步化的檢查動作，換句話說，當部署時會直接將舊的資料覆寫掉。你可以從「Build」選單中找到這一個工具，再根據你的需要選擇傳送位置就可以了。



您可以在發行之前，設定「Allow this precompiled site to be updatable」項目，若清除此項設定，則網站的 ASPX 網頁在發行後，其中的資訊都會被編譯起來。若只想預先編譯程式碼，而將來部署後，可能會修改 ASPX 的內容，則可勾選這個設定。



## 練習 12.3 : 使用 Publish Web Site 工具部署專案

目的：

練習使用 Visual Studio 工具 Publish Web Site 功能，將網站程式進行預先編譯之後，再部署到 IIS。

功能描述：

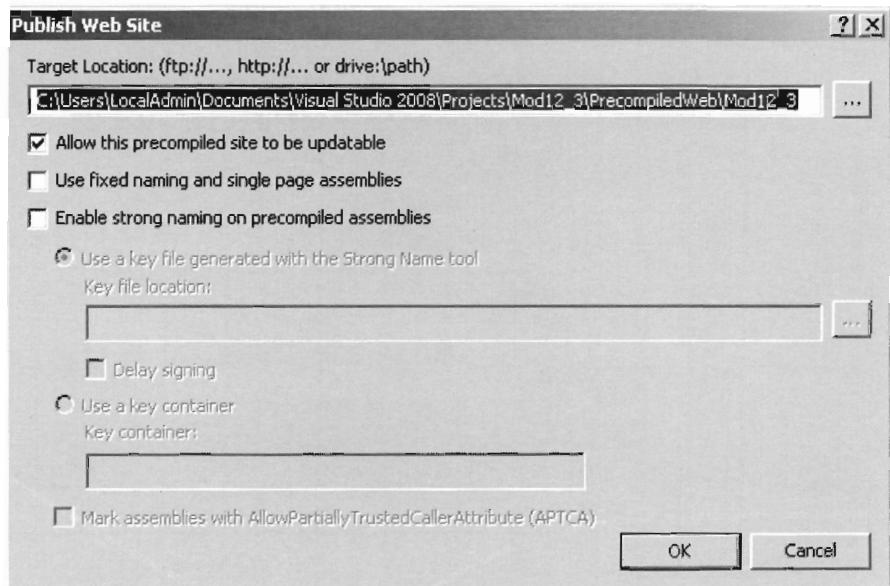
這個練習中，將了解如何操作 Publish Web Site 工具部署 ASP.NET 應用程式專案。

預估實作時間：10 分鐘

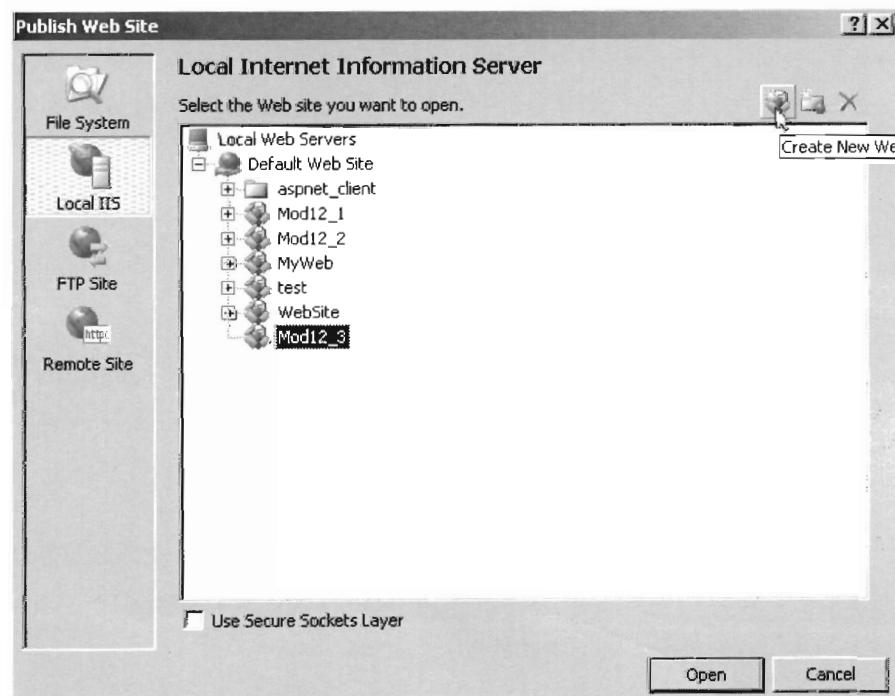
實作步驟：

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「File System」→並選取「\U9543\Practices\VB 或 CS\Mod12\_3\Starter\Mod12\_3」目錄開啓網站。

3. 在 Visual Studio 開發工具中，開啓「Build」選單，選擇「Publish Web Site」，開啓「Publish Web Site」工具視窗：

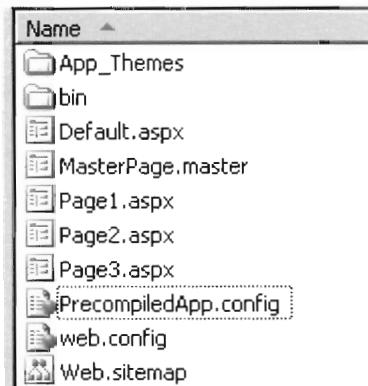


4. 點選「Target Location」旁的 [...] 按鈕，出現「Publish Web Site」路徑選擇畫面。選取「Local IIS」→「Default Web Site」→「Create New Web Application」 按鈕，建立新的應用程式目錄，命名為 Mod12\_3，按下「Open」按鈕，回到「Publish Web Site」工具視窗。



5. 確認「Allow this precompiled site to be updatable」已選取，按下「OK」按鈕，將網站發行。

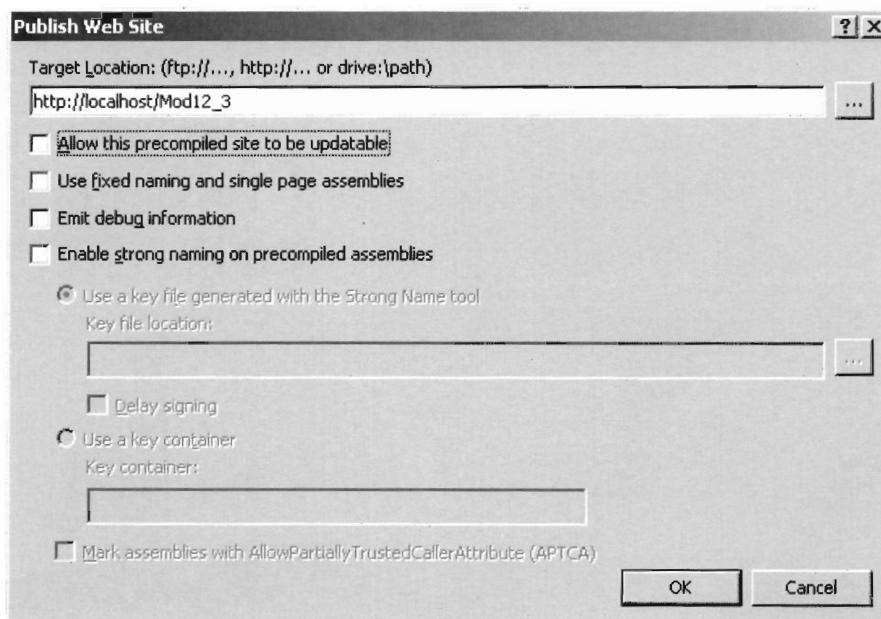
6. 檢視 C:\Inetpub\wwwroot 下是否正確建立 Mod12\_3 目錄，且所有檔案都正確複製到此目錄：



檢視目錄中產生的 PrecompiledApp.config 檔，其中設定如下組態

```
<precompiledApp version="2" updatable="true"/>
```

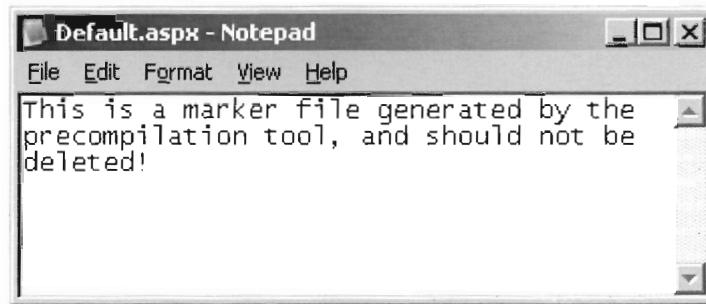
7. 在 Visual Studio 開發工具中，開啓「Build」選單，選擇「Publish Web Site」，開啓「Publish Web Site」工具視窗，清除選取「Allow this precompiled site to be updatable」，按下「OK」按鈕，將網站再度發行。

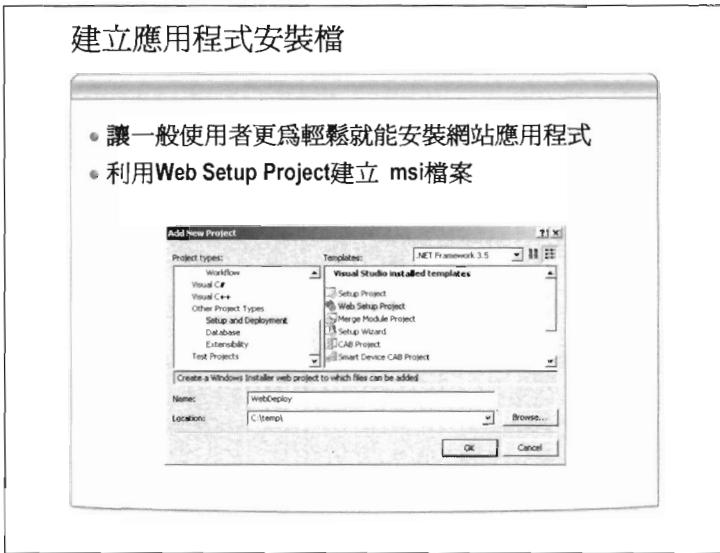


8. 檢視目錄中產生的 PrecompiledApp.config 檔，其中設定如下組態

```
<precompiledApp version="2" updatable="false"/>
```

利用 Notepad.exe 檢視目錄中產生的 default.aspx 網頁，可看到如下內容：





## 建立應用程式安裝檔

建立安裝檔可以讓一般使用者更為輕鬆就能安裝網站應用程式，也更容易攜帶、傳送、保存，經過適當的包裝就能產品化。

Windows 作業系統上有一個安裝技術，稱做 Windows Installer，從 Windows 2000 版開始，它便內建在作業系統之中。Windows Installer 提供安裝、移除、修復，以及管理應用程式的能力。安裝過程還可以提供交易保護動作，以及隨選安裝(installation on demand)的機制，讓使用者第一次使用到此應用程式時，再進行安裝動作。Windows Installer 會將應用程式相關檔案封裝在附檔名為 msi 的檔案中。

要建立安裝檔之前，請先確定網站 Web.config 的<compilation>區段的 Debug 屬性是否設定為 false，以避免網站將來會以除錯模式執行，此模式的執行速度較慢。

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

## 建立 Web Setup Project

Visual Studio 開發工具中，內建多種不同的專案範本。如果要將專案封裝成安裝檔再進行部署，你可以使用 Web Setup Project 開發專案的安裝程式。

### 加入專案項目

建立好 Web Setup Project 之後，接著就要加入要封裝的專案項目，將網站的「Project Output」或「Content Files」加入安裝專案。

### 封裝成安裝檔

當設定好 Web Setup Project 的內容之後，就可以把專案建置成安裝檔。只需要從「Solution Explorer」視窗中，點選安裝專案，按下滑鼠右鍵，從快捷選單中選取「Build」。當 Web Setup Project 建置完成，在專案的目錄下就會建立 msi 的安裝程式。

### 在目地電腦安裝

將來要部署應用程式時，只需要將 msi 檔複製到上線的機器執行就可以了。

### 練習 12.4：建立應用程式安裝檔

建立 ASP.NET 網站程式的應用程式安裝檔，將欲部署的網站檔案封裝在 msi 檔案之中，以便部署網站。

預估實作時間：10 分鐘

### 練習 12.4：建立應用程式安裝檔

#### 目的：

練習利用 Visual Studio 的 Web Setup 專案來建立應用程式安裝檔。並測試安裝與反安裝。

#### 功能描述：

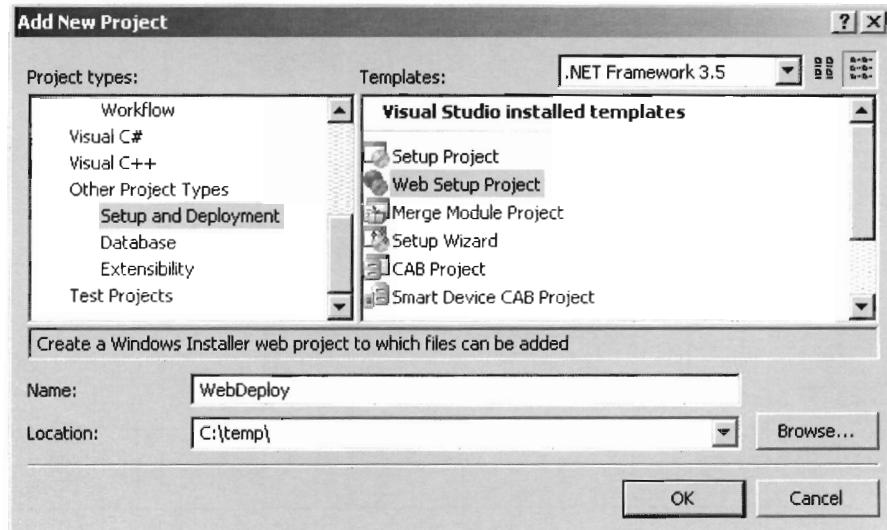
建立 ASP.NET 網站程式的應用程式安裝檔，將欲部署的網站檔案封裝在 msi 檔案之中，以便部署網站。

預估實作時間：10 分鐘

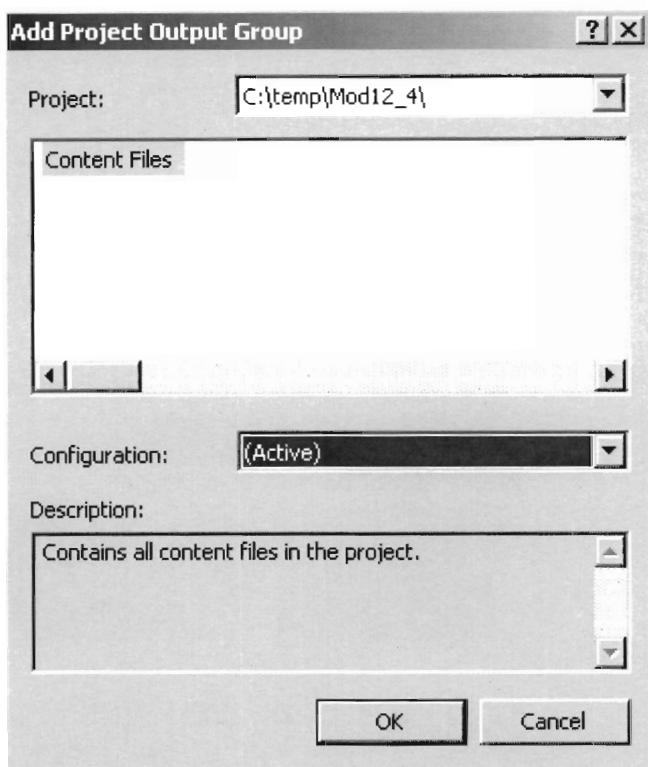
#### 實作步驟：

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「File System」→並選取「U9543\Practices\VB 或 CS\Mod12\_4\Starter\Mod12\_4」目錄開啓網站。

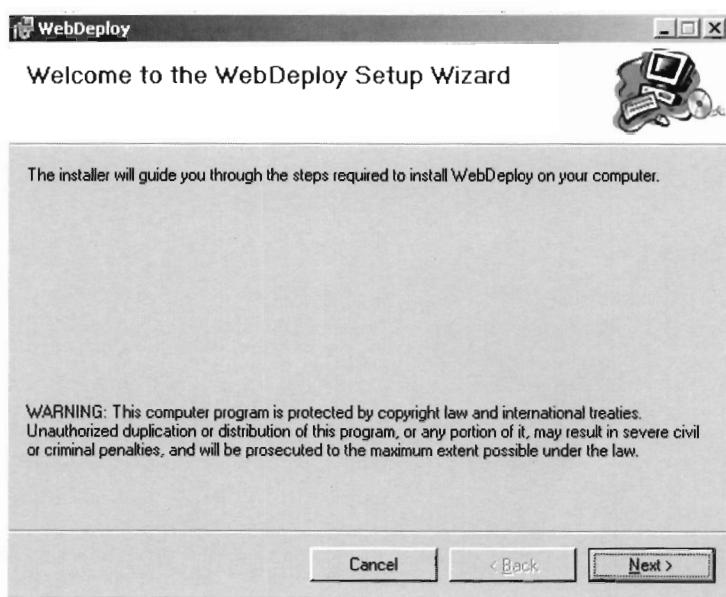
3. 在「Solution Explorer」視窗中，點選 Solution 按下滑鼠右鍵 → 「Add New Project」。在「Add New Project」視窗中，選擇「Other Project Type」→「Setup and Deployment」→「Web Setup Project」，專案名稱設為「WebDeploy」，選擇路徑：



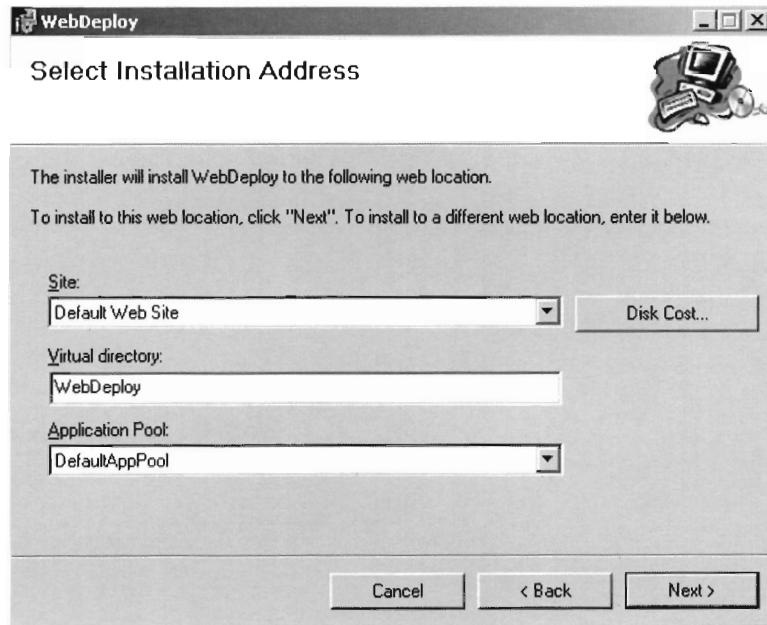
4. 按下「OK」按鈕，加入新專案。
5. 在「WebDeploy」專案中，按下滑鼠右鍵，選擇「Add」→「Project Output」，選擇 Mod12\_4 的「Content Files」，按下「OK」按鈕：



6. 點選「WebDeploy」專案，按下滑鼠右鍵，選擇「Build...」。
7. 專案建置完畢之後，可以直接在 Visual Studio 開發工具中測試。點選「WebDeploy」專案，按下滑鼠右鍵，選擇「Install」，開啟安裝畫面：



8. 按下「Next」按鈕，接下來設定要安裝的站台及虛擬目錄名稱，以及使用預設的應用程式集區：



9. 按下兩次「Next」按鈕，就開始安裝。安裝完成後會出現下面視窗。按下「Close」按鈕，結束安裝：
10. 開啓瀏覽器，執行 <Http://localhost/WebDeploy/Default.aspx>，測試功能是否正常。

## 總結

- 了解如何部署到IIS 7
- 了解如何使用**ASP.NET Web Site Administration Tool**
- 部署應用程式
- 使用**XCopy**部署網站
- 使用**Copy Web Site**部署專案
- 網站先行編譯 (Precompilation)
- 使用**Publish Web Site**部署專案

# 第十三章: 網站安全性規 劃

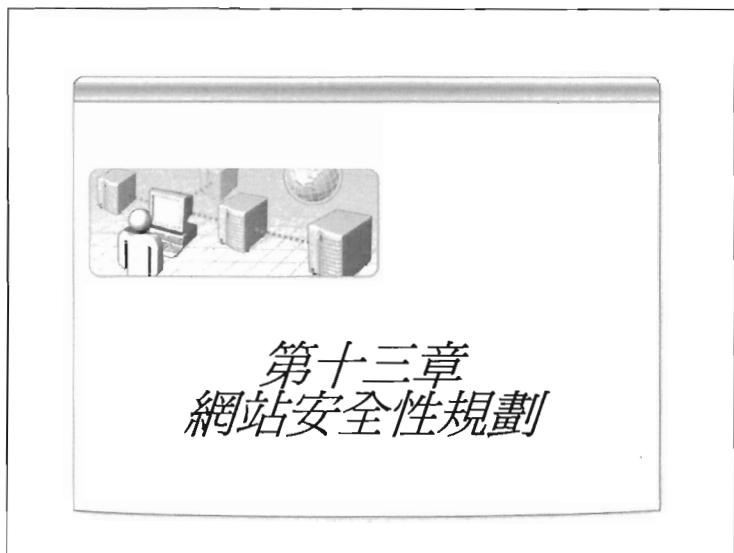
## 本章大綱

大綱.....	3
安全性名詞.....	4
驗證 (Authentication) .....	5
Windows 驗證.....	7
IIS 7 Class 模型.....	8
IIS 7 Integrated 模型 .....	10
啓用或停用 Window 驗證.....	11
在 ASP.NET 網站設定 Windows 驗證.....	13
練習 13.1 : 使用 Windows 驗證.....	15
Forms 驗證.....	19
啓用 Forms 驗證.....	20
撰寫登入網頁.....	23
練習 13.2 : 使用 Forms 驗證 .....	24
使用安全管理工具 .....	28
練習 13.3 : 使用安全管理工具建立使用者帳號與角色 .....	29
使用安全性控制項 .....	32
練習 13.4 : 建立 Login 網頁.....	34
使用 Create User Wizard 控制項 .....	38
練習 13.5 : 使用 Create User Wizard 新增使用者.....	39

作者：

許薰尹





## 大綱

- 驗證(Authentication) 與 授權(Authorization)
- URL 授權
- Windows 驗證
- Forms 驗證
- 使用安全管理工具
- 使用安全性控制項
- 總結

## 大綱

這個章節將會提到 ASP.NET 的安全性議題，以及如何實作這些安全機制。安全性將包含使用者的驗證、授權。ASP.NET 可以提供 Forms 驗證、Windows 驗證等機制：

- 驗證(Authentication) 與 授權(Authorization)
- URL 授權
- Windows 驗證
- IIS 7 Class 模型
- IIS 7 Integrated 模型
- Forms 驗證
- 使用安全管理工具
- 使用安全性控制項

## 安全性名詞

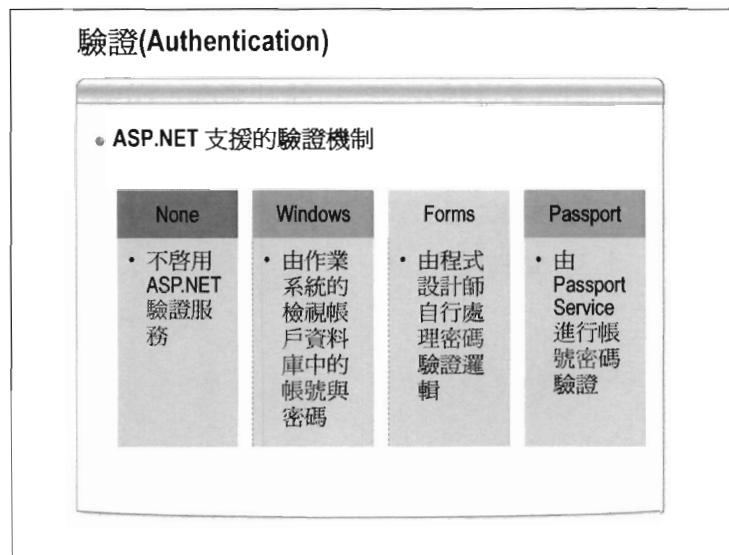
- 驗證 (Authentication)  
使用 Credential 證明你是誰
- 授權 (Authorization)  
用戶端是否有權限存取要求的資源
- 模擬 (Impersonation)  
使用已通過驗證並已授權的用戶端身份來執行程式
- 委派 (Delegation)  
更進階的模擬機制，允許以模擬的用戶端來存取遠端資源

## 安全性名詞

ASP.NET 提供非常強固的安全性系統來保障網站程式的安全。為了達到最佳的安全控管，ASP.NET 和微軟作業系統中的 Internet Information Service (IIS)、NTFS 檔案系統緊密的整合。

ASP.NET 會使用以下的機制來達到安全性：

- 驗證 (Authentication)：使用 Credential 證明你是誰。
- 授權 (Authorization)：用戶端是否有權限存取要求的資源。
- 模擬 (Impersonation)：使用已通過驗證並已授權的用戶端身份來執行程式。
- 委派 (Delegation)：更進階的模擬機制，允許以模擬的用戶端來存取遠端資源。



## 驗證 (Authentication)

驗證是指使用 Credential 證明你是誰的過程。用戶端提供的 Credential 最常見的便是帳號與密碼。Credential 必需經由驗證機構做檢查，這個機構可能利用存在於作業系統、資料庫的帳號密碼來比對。

ASP.NET 可以搭配 Web 伺服器 (如 IIS) 一起運作，以支援安全的驗證機制，讓你利用 Windows 作業系統提供的基本 (Basic)、摘要式 (Digest) 或整合式 Windows 驗證來進行安全控管。此外，ASP.NET 也支援 Microsoft Passport 驗證服務的功能，以達 Web 網站單一登入能力。

若想要使用 IIS 中的基本驗證，則需要在 IIS 管理工具中進行設定。ASP.NET 可以提供以下驗證方式：

- **None**  
不啓用 ASP.NET 驗證服務。不過 IIS 的驗證服務還是可以啓用。
- **Windows**  
預設的設定，ASP.NET 驗證服務將驗證使用者提供的帳號與密碼和作業系統的帳戶資料庫中的資料是否存在或一致。

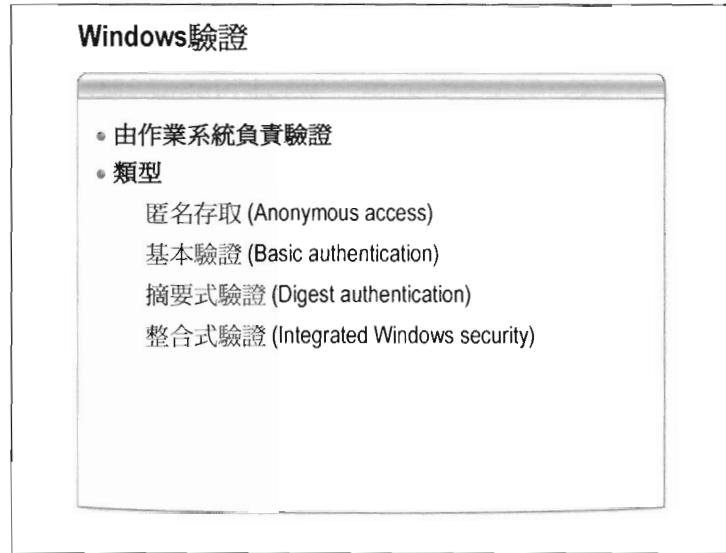
- Forms

ASP.NET 驗證服務將管理 Cookie 並且將使用者導向登入網頁，通常要啓用這個機制須要在 IIS 中啓用匿名存取，並自行撰寫登入邏輯。

- Passport

ASP.NET 驗證服務將透過<http://www.passport.com>提供驗證服務。IIS 7 不支援，但 IIS 5/IIS 6 可支援。

此外也支援表單式驗證 (Forms-Based Authentication)。表單式驗證使用 cookie 驗證使用者。這種驗證方式較適合 Internet 應用程式。

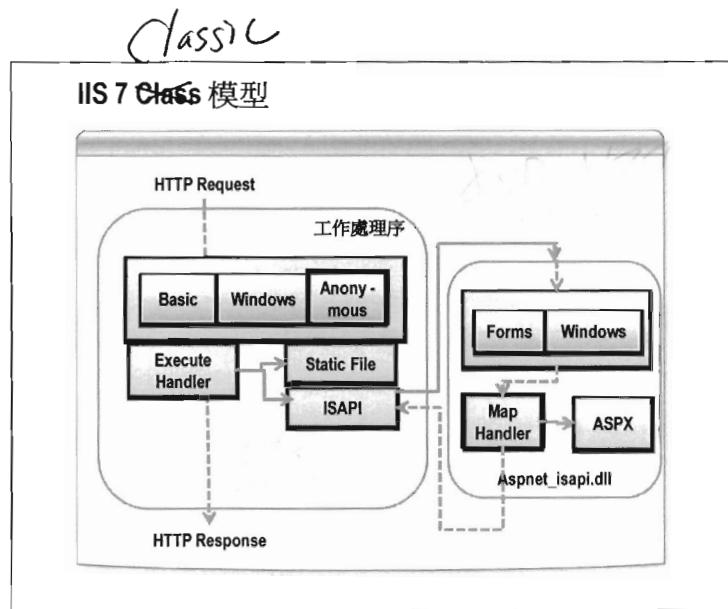


## Windows 驗證

IIS 提供許多 Windows 驗證機制，包含：

- 匿名存取 (Anonymous Access)：不驗證。
- 基本驗證 (Basic Authentication)：工業標準，所有瀏覽器都支援。帳號密碼以明碼方式傳。
- 摘要式驗證 (Digest Authentication)：改良自基本驗證，也是工業標準。但需目錄服務，且非所有瀏覽器都支援。帳號密碼不以明碼方式傳。
- 整合式驗證 (Integrated Windows Security)：最安全的驗證機制。使用目前登入的身份自動作驗證。

若網站不需要知道用戶端是誰，便可以使用匿名存取。在這種情況下，不進行驗證動作，這是網站的預設設定。



## IIS 7 Class 模型

IIS 7 提供兩種應用程式集區模型：

- Class 模型：IIS 7 使用 IIS 與 ASP.NET 分離模型處理請求，運作方式和 IIS 6 相同。
- Integrated 模型：IIS 7 使用整合 IIS 與 ASP.NET 的方式來處理請求。

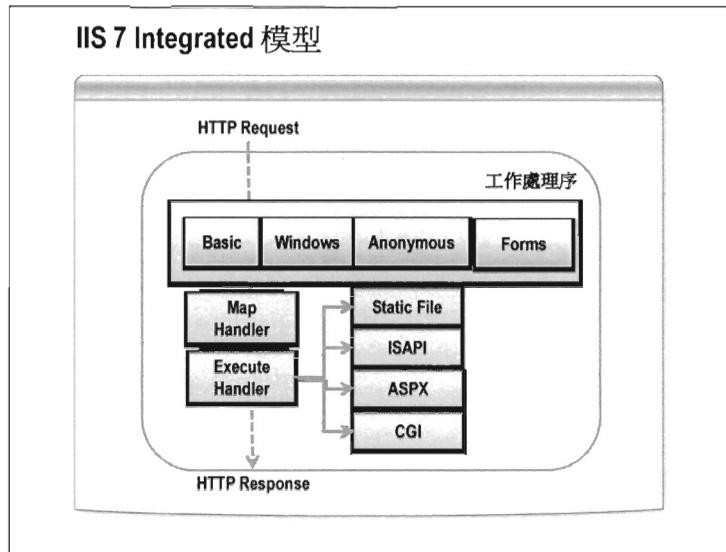
### Class 模型

IIS 7 的 Class 模型提供與 IIS 6 相容性，ASP.NET 請求的處理方式和 IIS 6 相同，處理 ASP.NET 請求時，會將 IIS 與 ASP.NET 請求的管道 (pipelines)分離。參考上圖，ASP.NET 實作成 Server Application Programming Interface (ISAPI) extension (aspnet\_isapi.dll)，以一個獨立的應用程式架構連接到伺服器，aspnet\_isapi.dll 負責處理 ASPX、ASMX...等檔案，但其它類型的檔案，如靜態檔案、CGI 程式就不能夠使用到 ASP.NET 提供的功能。

IIS 7 Class 模型的注意事項：

- 只處理有在 HTTP Handler 註冊的請求。

- 某些動作重複發生，如驗證。
- 某些設定要在網站與 IIS 重複設定，如授權。

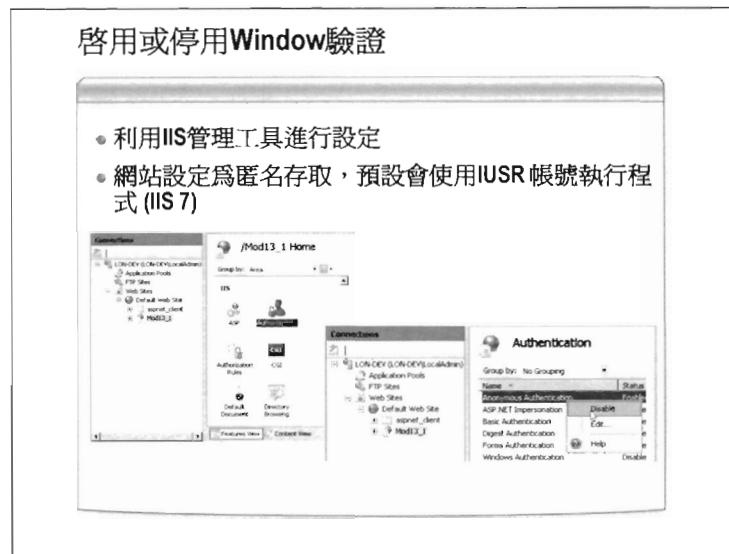


## IIS 7 Integrated 模型

IIS 7 提供 Integrated 模型將 IIS 與 ASP.NET 做更緊密的整合，處理流程如圖所示，兩者的請求管道(Pipeline)合而為一，能夠處理原生(Native)與 Managed 模組。原生模組都是以 IIS 7.0 C++擴充 API 實作的動態連結程式庫(dynamic-link libraries，DLL)。Managed 模組則是實作成.NET Framework 類別。

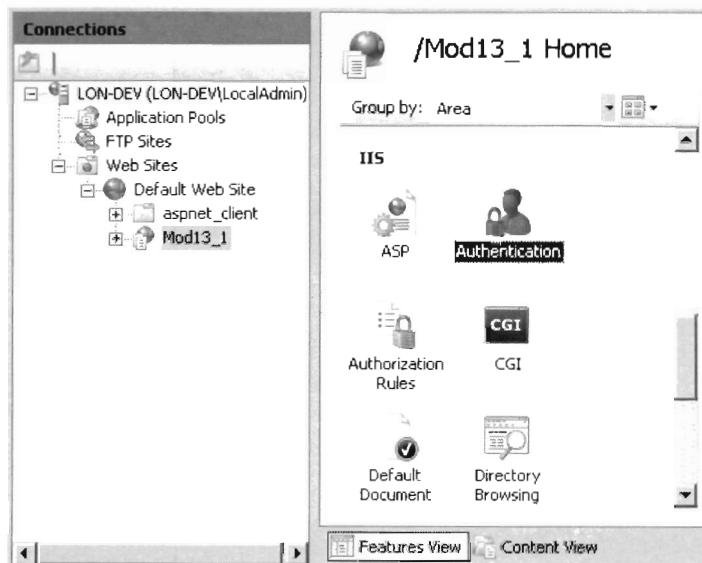
IIS 7 Integrated 模型提供以下好處：

- 能夠處理所有類型的 Request(原生或 Managed 模組)，如靜態檔案、CGI。
- 減少在 IIS 與 ASP.NET 重複執行的動作，如驗證、URL 授權。
- 集中管理設定。
- 易於使用 ASP.NET Managed 模組來擴充 IIS



## 啓用或停用 Window 驗證

若要啓用或停用網站的 Window 驗證方式，可以利用 IIS 管理工具進行設定。舉例來說，若要停用某個網站匿名存取功能，從 IIS 管理工具，選取某個網站，雙擊「Authentication」：

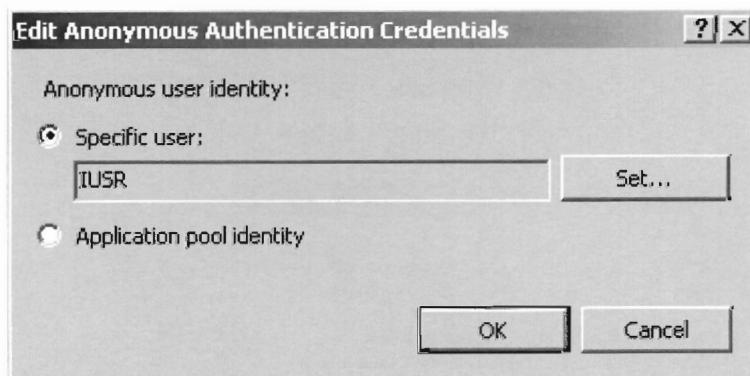


在「Anonymous Authentication」項目上方按滑鼠右鍵，點選「Disable」。



## IUSR 帳號

若網站設定為匿名存取，預設會以一個標準的 Windows 使用者帳號來執行程式，這個帳號為 IUSR，當在作業系統安裝 IIS 時，會自動建立此帳號。您可以利用 IIS 管理工具來變更改用別的帳號，例如指定特定帳號，或是使用應用程式集區的帳號(預設為 Network Service)：



### 在 ASP.NET 網站設定 Windows 驗證

- 需在 Web.Config 檔案中設定 authentication 項目 mode 為 Windows

```
<configuration>
 <system.web>
 <authentication mode="Windows" />
 </system.web>
</configuration>
```

- 啟用模擬機制

```
<identity impersonate="true" />
```

### 在 ASP.NET 網站設定 Windows 驗證

在 ASP.NET 網站要啟用 Windows 時驗證需在 Web.Config 檔案中設定：

```
<configuration>
 <system.web>
 <authentication mode="Windows" />
 </system.web>
</configuration>
```

ASP.NET 網站應用程式預設並非以使用者登入帳號執行。若要讓網站以使用者身份來執行，可以啓用 Impersonate 機制。這也代表了當使用者存取到這個網站系統資源時，就根據使用者帳號來判斷。舉例來說，能否存取檔案系統，就看 NTFS 的檔案系統授權。

由於預設 ASP.NET 並非以 Impersonate 的機制運作，若要啓用這個功能，可以在 Web.Config 檔案之中加上：

```
<identity impersonate="true" />
```

Windows 驗證可以用在內部網路上控管 Windows 的使用者的使用權限，而因為使用這種方式會結合 NTFS 的 ACL (Access Control List) 來控管目錄檔案的存取權限。

### 練習13.1：使用Windows驗證

•在這個練習中，將使用Windows驗證檢查使用者是否有存取權限。並將使用者使用的帳號和驗證的類型顯示在網頁上。

•預估實作時間：15分鐘

### 練習 13.1：使用 Windows 驗證

#### 目的：

了解 Windows 驗證，以及讀取使用者使用的帳號。在這個練習中，將使用 Windows 驗證檢查使用者是否有存取權限。並將使用者使用的帳號和驗證的類型顯示在網頁上。

#### 功能描述：

建立一個網站，並設定網站為 Windows 驗證，當要存取網站時，會自動跳出一個登入畫面，讓使用者輸入帳號和密碼。若輸入的帳號、密碼通過驗證，則可存取網站，並顯示使用者帳戶。

#### 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」，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「HTTP」，路徑設為

「[http://localhost/Mod13\\_1](http://localhost/Mod13_1)」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#。

3. 在 Web Site 中建立一個新網頁。從 Visual Studio 2008 開發工具「Web Site」選單 → 選取「Add New Item」，選取「Web Form」，清除「Place code in separate file」核取方塊，將檔案命名為 UseAuth.aspx。
4. 在 UseAuth.aspx 網頁上放三個 Label 控制項。
5. 在 UseAuth.aspx 網頁的 Page\_Load 事件加上以下程式碼：

Visual Basic

```
Label1.Text = User.Identity.Name
Label2.Text = User.Identity.AuthenticationType
Label3.Text = System.Security.Principal.WindowsIdentity.GetCurrent().Name
```

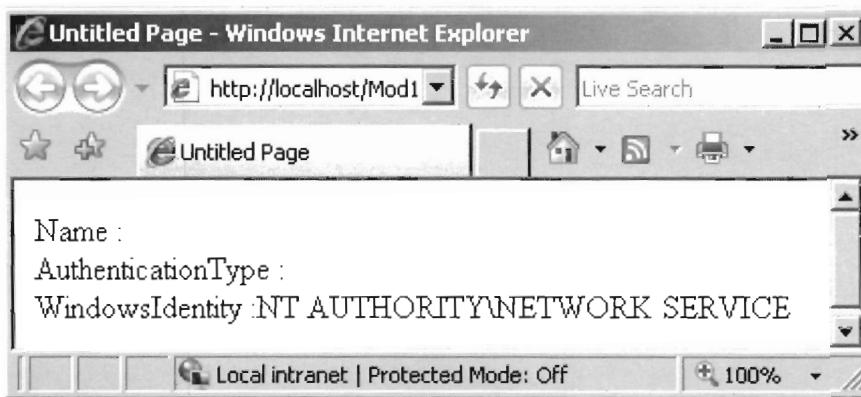
C#

```
Label1.Text = User.Identity.Name;
Label2.Text = User.Identity.AuthenticationType;
Label3.Text = System.Security.Principal.WindowsIdentity.GetCurrent().Name;
```

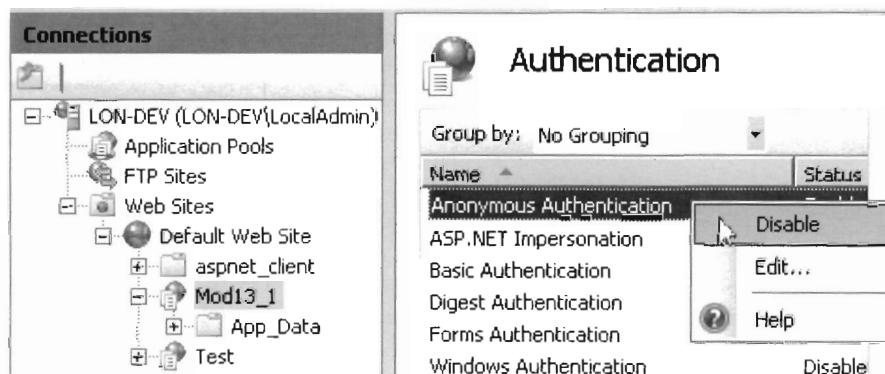
6. 檢視 Web.Config 檔案中設定 authentication 的 mode 預設為 Windows，看起來如：

```
<configuration>
<system.web>
 <authentication mode="Windows" />
</system.web>
</configuration>
```

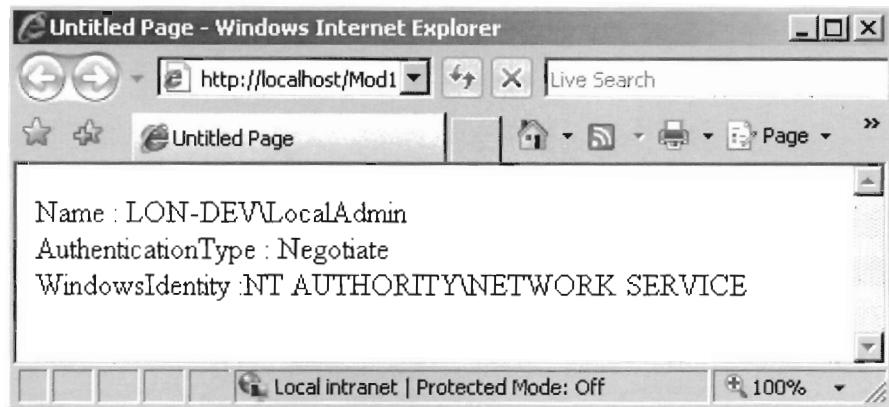
7. 在『Solution Explorer』→ 點選 UseAuth.aspx 網頁 → 按滑鼠右鍵 → 選「View In Browser」。你將發現 Name、AuthenticationType 都是空白，這是因為目前 IIS 啓用匿名存取的關係。WindowsIdentity.GetCurrent().Name 則為 Network Service。



8. 開啓 Internet Information Services 管理工具。在 Connections 視窗找到「Default Web Site」→點選「Mod13\_1」→雙擊中間視窗中的「Authentication」→點選「Anonymous Authentication」→按滑鼠右鍵，選取「Disable」。

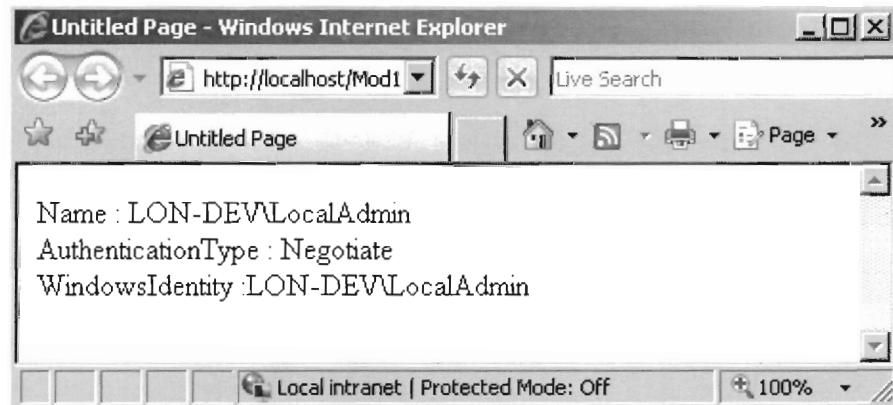


9. 重複上個步驟，點選「Windows Authentication」→按滑鼠右鍵，選取「Enable」。
10. 在「Solution Explorer」→點選 UseAuth.aspx 網頁→按滑鼠右鍵→選「View In Browser」。你就可以看到預設 ASP.NET 使用的驗證方式(跟使用作業系統與環境的不同，可能會有不同的執行結果)，Name 使用的身份是您的登入帳號。

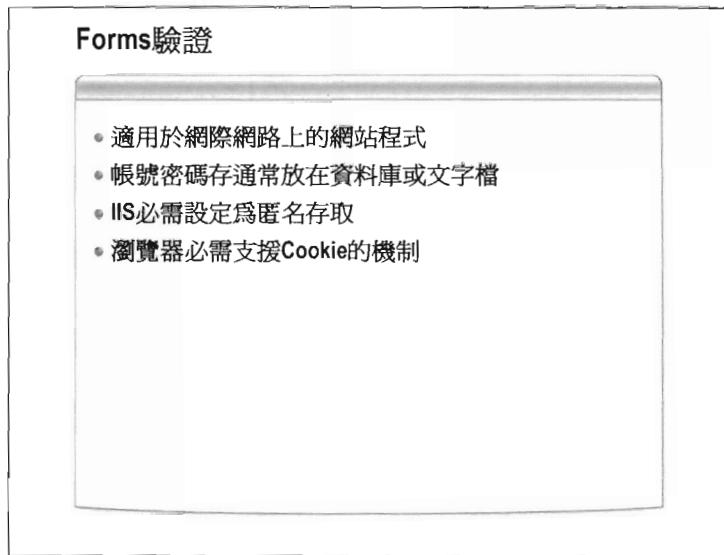


11. 重複步驟 8，點選「ASP.NET Impersonation」→按滑鼠右鍵，選取「Enable」。

12. 在「Solution Explorer」→點選 UseAuth.aspx 網頁→按滑鼠右鍵→選「View In Browser」。你就可以看到 WindowsIdentity 使用的帳號為登入帳號：



13. 結束程式執行，關閉瀏覽器。

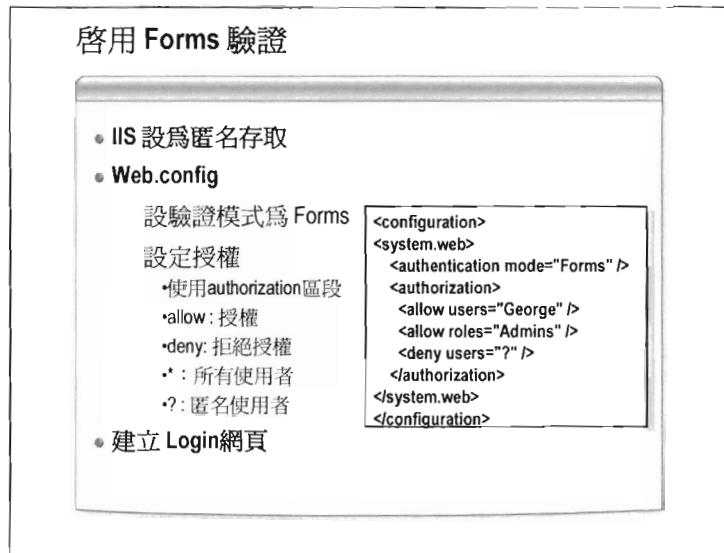


## Forms 驗證

Windows 驗證對於使用者帳號易於管控的內部網路來說相當有用，但網際網路上的網站程式，通常無法使用 Windows 帳號來做控管，比較好的方式是改用 Forms 驗證，將帳號密碼存放在資料庫或文字檔，再利用程式進行驗證。

若要使用 Forms 驗證，則 IIS 必需設定為匿名存取，並且在驗證成功之後，ASP.NET 會將驗證 Cookie 寫入瀏覽器，因此瀏覽器必需支援 Cookie 的機制，或者是使用 Cookieless Session。

Forms 驗證通過之後，若用戶端想要存取受安全保護的資源，所以必須再通過 URL 授權檢查。



## 啓用 Forms 驗證

這種類型的驗證需要在組態檔案中設定，且不能夠動態地改變。

Forms 驗證可以將使用者的帳號、密碼資料存放在資料庫或者檔案之中。

Web.config 檔案中，forms 常用的 Attribute 包含：

- loginUrl：描述登入網頁所在的 URL。
- name：HTTP Cookie 的名稱。若有多個應用程式要在同一台電腦上使用 Forms-based 驗證，則這些應用程式的 Cookie 的名稱應該設定為唯一、不能重複的。
- timeout：Cookie 的逾期時間，以分鐘為單位。
- path：已發行 Cookie 使用的路徑。預設為 /。

## 授權 (Authorization)

ASP.NET 支援兩種授權機制：一為 ACL 授權；一為 URL 授權。若 ASP.NET 程式要存取到系統資源，如檔案，則需檢查 ASP.NET 行程的執行身份是否有足夠的權限存取。

再者，ASP.NET 可以控制用戶端可否存取 URL 資源，稱之為 URL 授權。利用這個授權的設定進行安全性控制。你可以在 Web.Config 檔案中組態允許某個使用者或某群組使用者能否存取相關的資源。

## URL 授權

以下為一個實例的說明，在 Web.Config 檔案中 <authorization> 區段設定允許 George 帳戶以及 Admins 群組存取，但拒絕匿名者存取。

「?」代表所有匿名的使用者，所以如果匿名存取帳號沒有通過驗證就會被擋住，另外還有個符號「\*」是代表所有使用者的意思。

```
<configuration>
<system.web>
 <authentication mode="Forms" />
 <authorization>
 <allow users="George" />
 <allow roles="Admins" />
 <deny users="?" />
 </authorization>
</system.web>
</configuration>
```

如果使用者有一到多個，可以使用逗點隔開，如：

```
<allow users="George,Mary" />
```

你也可以設定使用者只能利用 HTTP 的 POST、GET 來存取。舉例來說，以下的設定：

```
<deny VERB="GET" users="*" />
```

將限制所有使用者不可以使用 GET 方式存取到 Web 應用程式。

若要啓用 Forms 驗證，請照以下步驟進行。

1. 將 IIS 設為匿名存取。
2. 在 Web.config 檔設驗證模式為 Forms。

```
<configuration>
<authentication mode="Forms">
 <forms name=".ASPxCOOKIE"
loginUrl="login.aspx" timeout="30" path="/" /> </forms>
</authentication>
</configuration>
```

3. 在 Web.config 檔設定拒絕匿名存取。

```
<authorization>
 <deny users="?" />
</authorization>
```

4. 建立 Login 網頁。



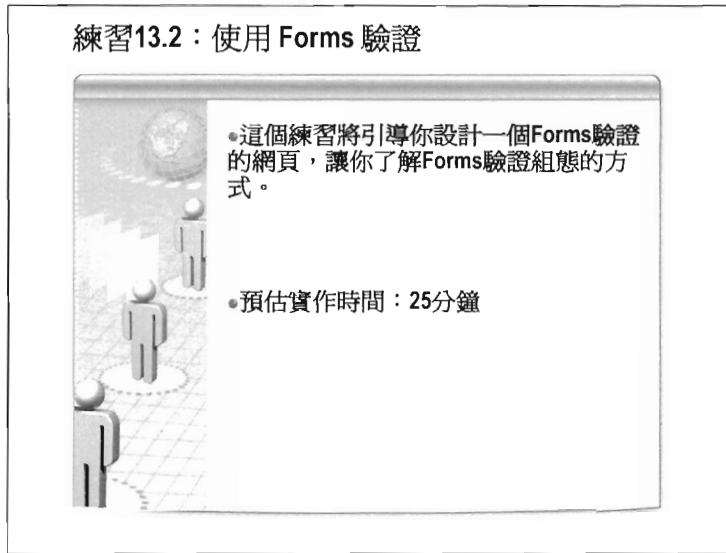
## 撰寫登入網頁

若要自行撰寫登入網頁，可以利用 System.Web.Security 這個命名空間所提供的類別，你可以在網頁中先匯入此命名空間：

```
<%@ Import Namespace="System.Web.Security" %>
```

這個命名空間下有一個 FormsAuthentication 類別，可用來進行驗證，常用的方法：

- Authenticate : 將使用者輸入的帳號密碼傳進去並且驗證，如果帳號密碼錯誤就會回傳 False。
- RedirectFromLoginPage : 將使用者導向欲存取的網頁。
- SignOut : 登出系統。



## 練習 13.2 : 使用 Forms 驗證

目的：

這個練習將引導你設計一個 Forms 驗證的網頁，讓你了解 Forms 驗證組態的方式。

功能描述：

建立一個網站，並設定網站為 Forms 驗證，以及設計一個登入網頁。當要存取網站時，則顯示登入畫面，讓使用者輸入帳號和密碼。若輸入的帳號、密碼通過驗證，則可存取網站，並顯示使用者帳戶。

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」→「New Web Site」→選取「ASP.NET Web Site」→將「Location」設為「HTTP」，路徑設為

「[http://localhost/Mod13\\_2](http://localhost/Mod13_2)」目錄，與使用的程式語言 (Language)，如 Visual Basic 或 C#

3. 開啓 Web.Config 檔案。雙擊「Solution Explorer」中的 Web.Config 檔案。
4. 修改 system.web 區段中的 authentication 元素。在 authentication 當中加入 Forms 元素：

```
<authentication mode="Forms">
 <forms loginUrl="login.aspx" /> <Identity impersonate="false" />
</authentication>
```

5. 修改授權設定，拒絕匿名者存取。在 authorization 元素之中加一個 deny 元素，拒絕匿名者。

```
<authorization>
 <deny users="?" />
</authorization>
```

6. 儲存 Web.config 檔案。
7. 在 Web Site 中建立一個新網頁。從 Visual Studio 2008 開發工具「Web Site」選單 → 選取「Add New Item」，選取「Web Form」，清除「Place code in separate file」核取方塊，將檔案命名為 Login.aspx。

設計登入網頁，使用兩個 TextBox，一個 Button 讓使用者輸入適當的帳號和密碼以進行登入。另外加入一個 Label 控制項，以顯示訊息。畫面看起來如：



8. 雙擊網頁上的「登入」按鈕，加入以下程式碼：

Visual Basic

```
Protected Sub Button1_Click(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Button1.Click
If TextBox1.Text = "ucom" And TextBox2.Text = "password" Then
```

```

FormsAuthentication.RedirectFromLoginPage(TextBox1.Text, _
False)
Else
 Label1.Text = "您提供的帳號或密碼有誤!"
End If
End Sub

```

C#

```

Protected void Button1_Click (Object sender, EventArgs e) {
if (TextBox1.Text == "ucom" || TextBox2.Text == "password") {
 FormsAuthentication.RedirectFromLoginPage(TextBox1.Text,
 false);
} else {
 Label1.Text = "您提供的帳號或密碼有誤!";
}
}

```

FormsAuthentication.RedirectFromLoginPage(TextBox1.Text, false)  
這個方法第一個參數傳入通過驗證的使用者，第二個參數設定是否產生 Cookie 值紀錄在使用者機器，如果設為 True，你可以使用多個瀏覽器共享同一個 Session，若設為 False 則還是一個瀏覽器一個 Session。

- 新增一個網頁，使用一個 Label 顯示歡迎訊息，一個 Button 讓使用者登入系統。畫面看起來如：



- 在 Default.aspx 網頁的 Page\_Load 事件以及，Button1 的 Click 事件加入以下程式碼：

Visual Basic

```

Protected Sub Page_Load(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles Me.Load
 Label1.Text = "歡迎, " + User.Identity.Name
End Sub
Protected Sub Button1_Click(ByVal sender As Object, _

```

```
ByVal e As System.EventArgs) Handles Button1.Click
 FormsAuthentication.SignOut()
 Response.Redirect("login.aspx")
End Sub
```

C#

```
void Page_Load(Object sender, EventArgs e) {
 Label1.Text = "歡迎, " + User.Identity.Name;
}
void Signout_Click(Object sender, EventArgs e) {
 FormsAuthentication.SignOut();
 Response.Redirect("login.aspx");
}
```

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

因為未曾登入過，所以您將被導向 Login.aspx 網頁。若正確輸入了帳號 (ucom) 和密碼 (password)，則您將被導向 Default.aspx 網頁，。

12. 按 Default.aspx 網頁上的「登出」按鈕，則回到 Login.aspx 網頁。
13. 結束程式執行，關閉瀏覽器。

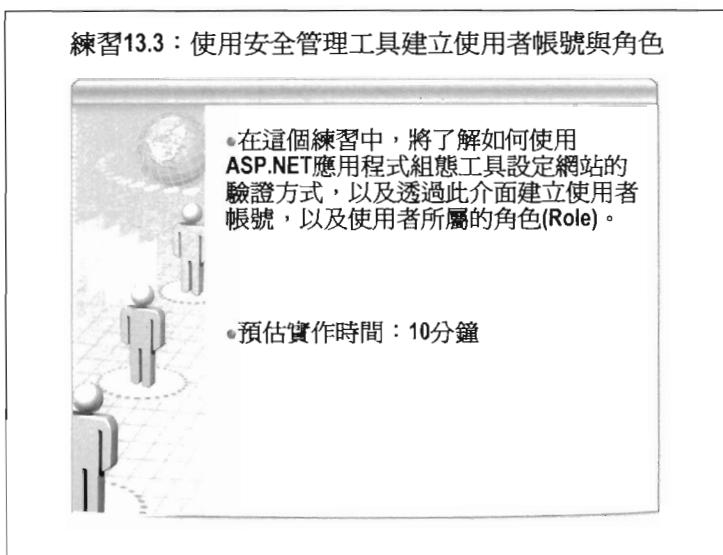


## 使用安全管理工具

ASP.NET 提供一個網站型式的管理界面，讓你透過網站設定 ASP.NET 應用程式的常用組態設定。你可以使用此網站設定安全性、應用程式組態、Provider 組態等等。

你可以從 Visual Studio 開發工具，「WebSite」選單 → 「ASP.NET Configuration」，啓用網站管理工具。

這個工具提供了安全性、應用程式組態...等等多種設定。



### 練習 13.3：使用安全管理工具建立使用者帳號與角色

目的：

在這個練習中，將了解如何使用 ASP.NET 應用程式組態工具設定網站的驗證方式。

功能描述：

使用 ASP.NET Configuration 工具，以及透過此介面建立使用者帳號，以及使用者所屬的角色 (Role)。

預估實作時間：10 分鐘

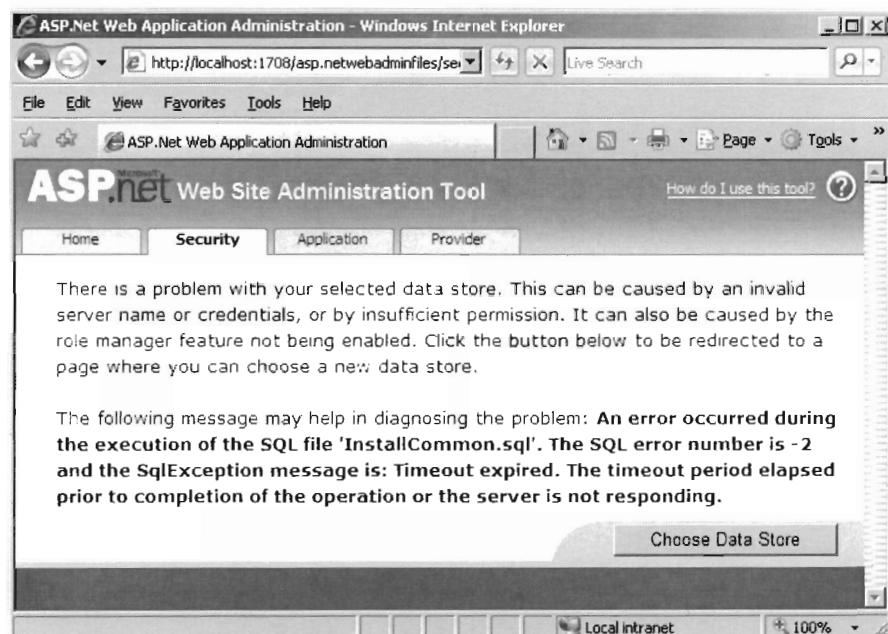
實作步驟：

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「\U9543\Practices\VB 或 CS\Mod13\_3\Starter」目錄，

與使用的程式語言 (Language)，如 Visual Basic 或 C#，將站台取名為「Mod13\_3」。

3. 選取 Visual Studio 2008 的「WebSite」選單 → 「ASP.NET Configuration」，啓用網站管理工具。
4. 設定網站使用表單式驗證 (Forms-Based Authentication)。點選「Security」→ 選取「Select authentication type」→ 「From the internet」→ 「Done」。

若第一次使用則出現以下訊息，提示該應用程式要使用的帳號資料庫。若非第一次使用，將不會出現以下畫面，直接跳到下一個步驟。點選「Choose Data Source」設定帳號資料庫。



點選「Select a single provider for all site management data」，預設使用 AspNetSqlProvider 存取帳號資料。預設此精靈會幫你建立一個 ASPNETDB.MDF 檔案存放使用者帳戶資料。

5. 回到「Security」頁籤 → 「Enable roles」→ 「Create or Manage roles」→ 建立一個新的角色，名稱為「Users」，按「Add Role」，輸入「Users」→ 「Back」。

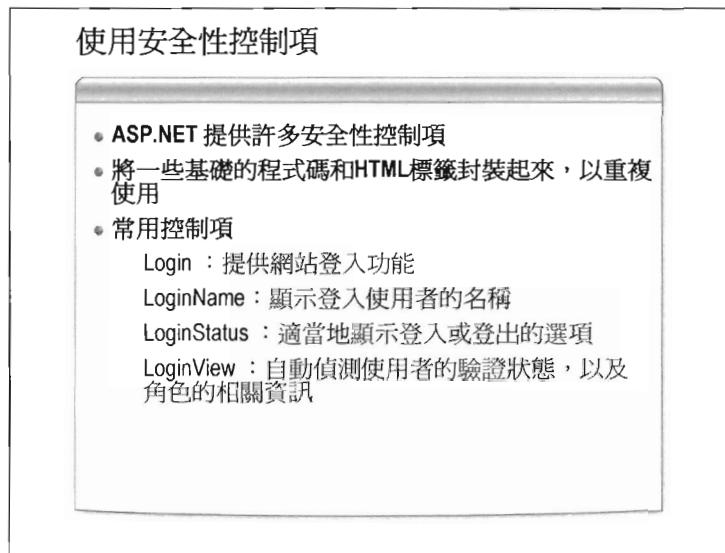
6. 回到「Security」頁籤→「Create User」輸入欲新增的使用者之相關資訊，如帳號「admin」，密碼「P@ssw0rd」並設定此使用者屬於 Users 角色。並請牢記您輸入的資訊。

The screenshot shows the 'Create User' page with the following details:

- Create User** (Page Title)
- Sign Up for Your New Account**
- User Name:
- Password:
- Confirm Password:
- E-mail:
- Security Question:
- Security Answer:
- Roles**
- Select roles for this user:  
 Users
- Create User** (Submit Button)

Below the form, there is a checked checkbox labeled  Active User.

7. 結束 ASP.NET Configuration 網頁。



## 使用安全性控制項

ASP.NET 提供許多控制項讓您設計安全性的網頁更加地容易。控制項包含：

- Login 控制項：提供登入網頁共用的使用者介面。
- LoginName 控制項：顯示登入使用者的名稱。
- LoginStatus 控制項：顯示登入狀態為登入或登出。
- LoginView 控制項：當匿名使用者或有帳戶的使用者登入時要顯示的畫面。
- PasswordRecovery 控制項：提供密碼重設或查詢。
- ChangePassword 控制項：變更密碼。
- CreateUserWizard 控制項：建立使用者帳號。

這些控制項都是組合式的控制項，提供了可自訂的使用者介面。這些控制項將一些基礎的程式碼和 HTML 標籤封裝起來，以便可以讓其它應用程式重複地使用。以下說明常用控制項：

## Login 控制項

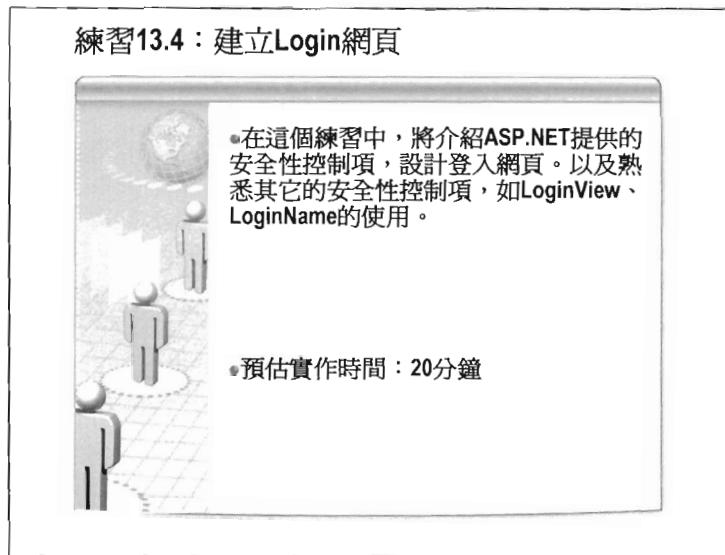
這個控制項包含以下功能：使用者名稱、密碼的文字方塊提供輸入，使用者名稱、密碼的提示文字顯示 (Label) 控制項、以及一個登入按鈕。有時包含一個記住密碼核取方塊。也可再加一個超連結，以讓使用者查詢遺忘的密碼。也可以利用超連結連結到建立使用者帳號的網頁。

## 使用 LoginStatus 控制項

LoginStatus 控制項會自動偵測使用者的驗證狀態，然後適當地顯示登入或登出的選項，不需要撰寫任何程式碼。當你點選 Login 按鈕，此控制項會自動將使用者導向網站的登入網頁。當你點選 Logout 按鈕，將會清除使用者的驗證狀態。

## 使用 LoginView 控制項

LoginView 控制項自動偵測使用者的驗證狀態，以及角色的相關資訊，並自動呈現給使用者。這個控制項是以樣版 (Template) 為基礎建立的，你可以使用 LoginView 搭配其它控制項，像 LoginStatus 與 LoginName 控制項一起使用。



## 練習 13.4：建立 Login 網頁

目的：

在這個練習中，將介紹 ASP.NET 提供的安全性控制項，

功能描述：

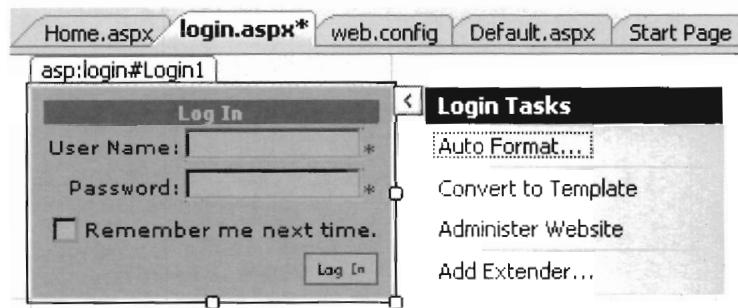
設計登入網頁。以及熟悉其它的安全性控制項，如 LoginView 控制項、LoginName 控制項的使用。

預估實作時間：20 分鐘

實作步驟：

1. 開啓上一個練習完成的 Mod13\_3 網站，或從路徑為「磁碟：  
\\U9543\Practices\VB 或 CS\Mod13\_4\Starter\Mod13\_3」目錄開啓之。「File」→「Open」→「Web Site」→選「File System」，選取 Mod13\_3 網站所在的檔案路徑。
2. 在 Web Site 中建立一個新網頁。從 Visual Studio 2008 開發工具「Web Site」選單 → 選取「Add New Item」，選取「Web Form」，清除「Place code in separate file」核取方塊，將檔案命名為 login.aspx。

3. 重複上一個步驟，在 Web Site 中建立一個 Home.aspx 網頁。
4. 從「Toolbox」→「Login」頁籤→拖曳 Login 控制項到登入網頁 login.aspx。
5. 設定 Login 控制項樣式。選取設計畫面上的 Login 控制項，點選智慧型標籤，並使用「Auto Format」功能進行格式化。



6. 設定 Login 控制項的「DestinationPageUrl」屬性為 Home.aspx。代表若登入成功，則導向此網頁以顯示。
7. 拖曳「LoginView」到 Home.aspx 網頁設計畫面中。從「Toolbox」→「Login」頁籤→拖曳 LoginView 控制項到 Home.aspx 網頁。在「AnonymousTemplate」檢視中，輸入「您尚未登入此系統!請先登入!」。畫面看起來如：



8. 在「LoggedInTemplate」檢視中，輸入「歡迎使用此系統!」。畫面看起來如：



9. 在 Home.aspx 網頁設計畫面上放一個 LoginStaus 與 LoginName 控制項。畫面看起來如：



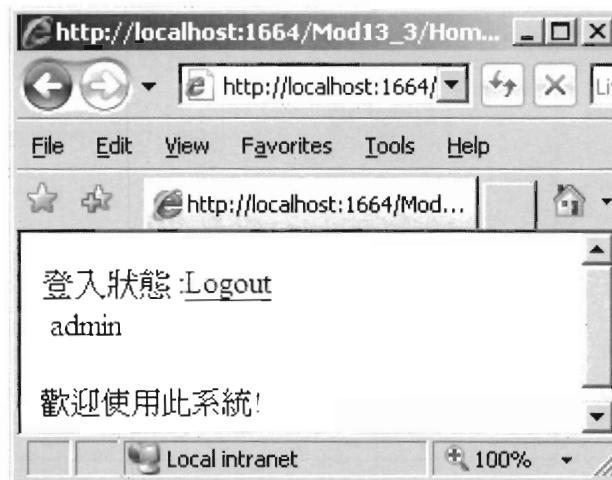
10. 在「Solution Explorer」→點選 Home.aspx 網頁→按滑鼠右鍵 →選「View In Browser」。

由於尚未登入，因此顯示「您尚未登入此系統!請先登入!」訊息。畫面看起來如：



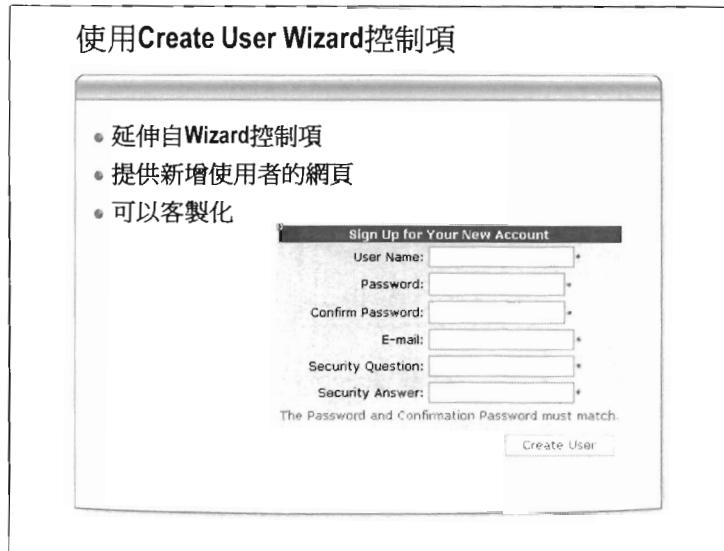
11. 點選網頁上的「Login」超連結，進入 Login.aspx 網頁輸入帳號「admin」與密碼「P@ssw0rd」。

12. 若登入成功，LoginName 控制項將會自動顯示您的登入帳號；而 LoginStatus 則會顯示「Logout」文字；「LoginView」控制項則會顯示「歡迎使用此系統！」訊息。畫面看起來如：



13. 點 Logout。則登出此系統，回到登入畫面。

14. 結束程式執行，關閉瀏覽器。



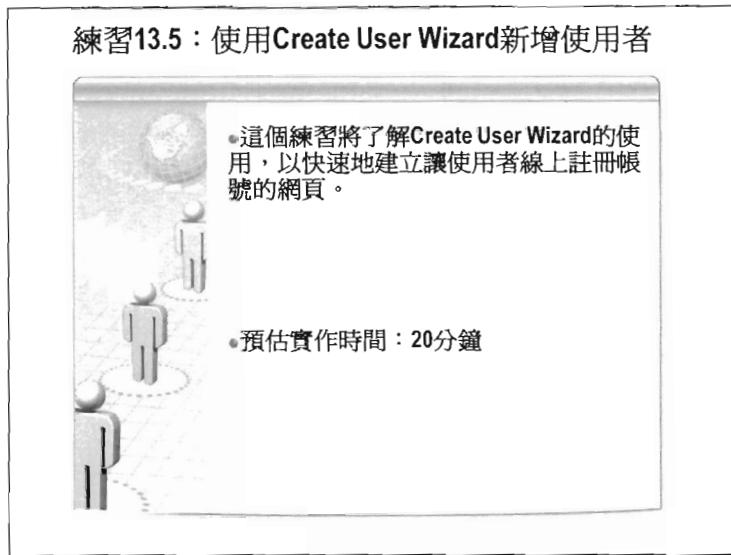
## 使用 Create User Wizard 控制項

CreateUserWizard 控制項延伸自 Wizard 控制項，提供新增使用者的網頁。

你可以直接從工具箱將 CreateUserWizard 控制項拖曳到網頁。

```
<asp:CreateUserWizard ID="CreateUserWizard1" runat="server"
 ContinueDestinationPageUrl="~/Login.aspx" >
 <WizardSteps>
 <asp:CreateUserWizardStep runat="server">
 </asp:CreateUserWizardStep>
 <asp:CompleteWizardStep runat="server">
 </asp:CompleteWizardStep>
 </WizardSteps>
</asp:CreateUserWizard>
```

你可以採取許多種方式客製化 CreateUserWizard 控制項以滿足網站應用程式實際的需求。CreateUserWizard 控制項提供許多視覺化的項目，以顯示建立新使用者的網頁介面。



## 練習 13.5：使用 Create User Wizard 新增使用者

**目的：**

這個練習將了解 Create User Wizard 的使用，以快速地建立讓使用者線上註冊帳號的網頁。

**功能描述：**

利用 ASP.NET 預設的 Create User Wizard 控制項，馬上建立一個新增使用者的網頁。此精靈會自動將帳號儲存到資料庫。

**預估實作時間：20 分鐘**

**實作步驟：**

1. 開啓上一個練習完成的 Mod13\_3 網站，或從路徑為「磁碟：  
\\U9543\Practices\VB 或 CS\Mod13\_5\Starter\Mod013\_3」目錄  
開啓之。「File」→「Open」→「Web Site」→選「File  
System」，選取 Mod13\_3 所在的檔案路徑。
2. 在 Web Site 中建立一個新網頁。從 Visual Studio 2008 開發工  
具「Web Site」選單 → 選取「Add New Item」，選取「Web

Form」，清除「Place code in separate file」核取方塊，將檔案命名為 AddUser.aspx。

3. 拖曳「Create User Wizard」控制項到 AddUser.aspx 網頁。從「Toolbox」→「Login」頁籤→拖曳 Create User Wizard 控制項到 AddUser.aspx 網頁。畫面看起來如：

**Sign Up for Your New Account**

User Name:  \*

Password:  \*

Confirm Password:  \*

E-mail:  \*

Security Question:  \*

Security Answer:  \*

The Password and Confirmation Password must match.

Create User

4. 設「Create User Wizard」控制項的 ContinueDestinationPageUrl 屬性為 Login.aspx。
5. 在 Login.aspx 網頁上 Login 控制項下方，新增一個按鈕「註冊帳號」，拖曳一個 Button 控制項到畫面。將 Text 屬性設定為「註冊帳號」。畫面看起來如：

**Log In**

User Name:  \*

Password:  \*

Remember me next time.

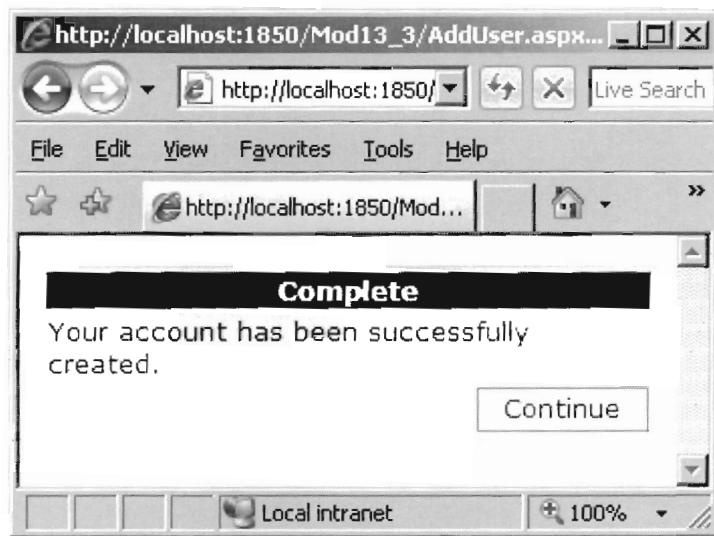
Log In

註冊帳號

6. 設按鈕的 PostBackUrl 屬性為「AddUser.aspx」。
7. 執行 Login.aspx 網頁。在「Solution Explorer」→點選 Login.aspx 網頁 → 按滑鼠右鍵 → 選「View In Browser」→ 點選「註冊帳號」。輸入一個新的帳號，以及相關註冊資訊。畫面看起來如：



8. 當按下「Create User」按鈕，若成功進行帳戶的新增，您將看到帳戶成功建立的訊息。畫面看起來如：



9. 結束程式執行，關閉瀏覽器。

## 總結

- 網站安全性名詞
- 了解驗證(Authentication)與 授權(Authorization)
- 實作ASP.NET Windows驗證
- 實作ASP.NET Forms驗證
- ASP.NET 安全性控制項

## 總結

網站安全性一直是所有網管人員關心的問題。若沒有適當地設定網站的安全機制，網站就容易被有心人士侵入，而導致企業有形或無形資產的損失。

ASP.NET 提供 Windows 驗證，適合應用在企業內部的網站。而 Forms 驗證則適合應用在網際網路上。

ASP.NET 提供許多安全性的控制項，讓使用者能夠快速地建立登入網頁、查詢密碼的網頁。