推薦序

邁向現代化網站技術的世界 — ASP.NET 開發者的路更廣！

Web 開發技術在這幾年來，為了因應行動裝置及雲端服務的應用趨勢，有很大的變化及演進，而 MVC 這樣的開發框架，在大型及關鍵的應用開發也愈來愈普及，然而開發一個完整且現代化的 Web 應用，所需要的技術也變得更多元，常讓剛進入 Web 開發領域的新手無所適從，ASP.NET 及 Visual Studio 2013 為了因應新時代的軟體開發趨勢及需求，提供了一個完善的開發環境並且不斷地強化，讓開發者很容易地上手以進行開發工作。

例如 ASP.NET Web API 這項技術，打造一個共用性高的服務供不同平台及應用來存取，是開發人員所需學習的實用技術，另外像 Bootstrap 這個 CSS/JavaScript 框架，在 ASP.NET MVC 或 Web Forms 的專案範本都已內建了，讓開發者很容易地開發一個響應式網頁，提供給手機、平板及各種裝置之用戶最好的操作體驗；還有現在許多對外的網路服務可使用各種社交帳號登入，透過 ASP.NET Identity 這個令人讚賞的技術，開發身份驗證及管理的程式更容易了!

這本由多位微軟 MVP 所執筆的 ASP.NET MVC 書籍，融合了這些開發現代化網站所需的技術，難得集結了各專業領域有實務經驗的專家，由淺到深的章節設計，很適合有系統性的學習，書中這些現代化網站的技術也專門寫成獨立的章節，即使是使用 Web Forms 技術的開發者也可以參考，還有像是程式的日誌處理，都是很實用的網站開發技巧。本書最後提到了軟體測試以及網站安全性的觀念及作法，例如如何設計一個安全且維護性高的網站架構、如何測試以確保程式的品質，這些實務上會碰到的問題及作法，可運用在你的工作上，更增加了本書的價值及實用性。

台灣微軟 開發工具暨平台推廣處
資深產品行銷經理

吳典璋 Dann Wu

推薦序

擁抱 ASP.NET MVC，網站開發再升級

ASP.NET Web Forms 概念的出現掀起一波網頁開發革命，對 HTML、JavaScript 一無所知的開發者也能寫出 AJAX 網頁，簡直是件奇蹟。然而，魔法般的 Web Forms Server Event，背後需付出效能代價；過於便捷自由的開發方式，則易讓初學者不知不覺在前後端穿插商業邏輯與 SQL 查詢，導致邏輯散亂卻又緊密相依，維護與測試難度大增。對照 Java Spring Framework、Ruby on Rail、CakePHP 等架構，運用 MVC 設計模式已然成為網站開發主流；常陷於邏輯混雜泥沼的 ASP.NET Web Forms 網站，有時會給人欠缺嚴謹性，難以建構中大型系統的非專業印象。

2009 年微軟推出 ASP.NET MVC 1.0，自此.NET 開發者有了新選擇。依循 MVC 架構，開發人員能依直覺切割出 HTML/JavaScript 等 UI 端邏輯 (View)、資料邏輯 (Model) 以及流程銜接邏輯 (Controller)，無形中實踐 MVC 最重要的關注點分離 (SoC) 精神。而 ASP.NET MVC 在設計時大量應用依賴注入 (DI)、AOP (ActionFilter 即為經典範例) 等設計模式，處處預留擴充與改寫彈性，使架構易於調整，足以面對各式艱鉅挑戰。而網站能結合單元測試，相較於 Web Forms 也是一大突破。轉眼間，ASP.NET MVC 已演進到第五版，架構轉向 OWIN 開放標準並擺脫對 IIS 的依賴，開發者可自由抽換網站平台、身分認證或是處理管道的任一組件，可塑性幾乎沒有極限。

個人參與 ASP/ASP.NET 網站開發十多年，乍見 ASP.NET MVC 便心動不已，深入了解後更為其架構之美吸引，之後即在新專案身體力行。但對現有 Web Forms 開發人員而言，MVC 概念差異甚大，轉換需要一番學習及導引。個人曾在部落格發表 ASP.NET MVC 豬走路系列文章，便是希望引導更多人投身 ASP.NET MVC，但網路文件及部落格文章畢竟較為片段零散，系統化整理必備知識的書籍仍是學習新技術較有效率的途徑。

繼 ASP.NET MVC 4 開發美學之後，由微軟 MVP 小朱領軍，與 demo、Dino、Bruce 等長期致力 ASP.NET MVC 推廣的 twMVC 社群專家，以及專注測試領域的 91，聯手為 ASP.NET MVC 再添一本中文參考書。對照自身的 ASP.NET MVC

開發經驗，全書完整涵蓋使用 Visual Studio 開發 ASP.NET MVC 所需的知識，從 MVC 基本概念、開發環境操作技巧、設計細節到注意事項都有詳實闡述。除了 ASP.NET MVC 本身，對於新一代 .NET 語言的重要特色如 LINQ、Lambda、Entity Framework、非同步呼叫等亦有所著墨，有利開發者活用新武器。至於資訊安全議題，在書中亦未缺席。藉機在此呼籲，如果你開發網站卻不知道 XSS 及 SQL Injection，建議搞懂之前一行程式都別寫，以免禍國殃民。最後，除了善用設計模式強化架構，實現自動測試也是專業開發重要的一環。自動測試不只能確保軟體品質，更是進行程式重構與系統優化不可或缺的基石，為進階開發的必備技能，建議讀者在轉換 MVC 時一併補齊。

由近年微軟技術大會中 ASP.NET MVC 議題所佔比重，不難感受 ASP.NET MVC 將是未來 ASP.NET 開發的主流。或許讀者們手上的 Web Forms 系統仍需繼續維護運轉十幾年 (望向手邊還活得好好的企業內部 ASP 程式)，但一定要密切掌握 ASP.NET MVC 的發展，提早學習做好準備，期盼大家早日加入 ASP.NET MVC 的行列，領略其架構之美。

技術部落格 「黑暗執行緒」站長

李明儒 Jeffrey

作者序

這是人生中第二本書，和上一本寫作時的心情很不同，這一年除了持續在運作的 twMVC 社群，我又創立了 SkillTree 的教育組織來推廣網頁相關技術，並且離開了原本上班族的生活，自己出來創業。多重壓力的襲擊下腦中經常冒出：「別寫了！這年頭寫書又賺不了什麼錢。」這樣的念頭，中途也實際停筆過，但經常在企業內訓、公開課程甚至拜訪客戶時都會被追問到新書的出版時間，問到都心虛了，於是又重新開始動筆，好在小朱的緊迫盯人戰術非常有效，原稿竟然不可思議的如期交付出版社，不過在原稿交出的當天晚上可能是長期緊繃的壓力得到釋放，我就立即的感冒了，並且躺了幾天。

一本書籍完成的過程需要許多人的配合與協助，不免俗的要感謝這次願意和我再次跳下來的作者小朱、Bruce、91 以及初次被推坑的 Dino，再感謝一直在旁邊說我寫的很差的女王大人黃米奧，當然也非常感謝各位看到此序的讀者，中文書籍環境困難，感謝各位還願意抱一個磚頭書回家，書籍上的疑慮或開發上的問題歡迎到 twMVC 與 SkillTree 的活動來找我們，期待這本書可以讓各位讀者收穫滿滿。

Demo
http://demoshop.tw
2014 年 8 月 1 日

作者序

自 ASP.NET MVC 4 網站開發美學一書 (以下簡稱 MVC 4 美學) 出版一年半之際，很快的，ASP.NET MVC 5 在 2013 下半年挾著 Visual Studio 2013 之勢問市，搭配著 Entity Framework 6 以及 Web API 2.0 等熱門技術快速席捲了 Microsoft 開發平台的社群，台灣當然也不會自外於這股浪潮之中，因此在 MVC 4 美學出版半年後，2013 年的 9 月中，兩位主要作者 demo 以及 Bruce 被我抓去重慶南路的 KFC，商討 MVC 4 美學下一版的編寫事宜，並確立大綱與寫作方向。因應 MVC 5 的改變，以及市場對 RWD 的變化，我們決定邀請對 MVC 開發有豐富實戰經驗的 Dino 加入我們的團隊，並再次邀請在軟體測試領域學有專精的 91 編寫測試這個章節，並將測試一章完全交給 91 負責，作者群至此確立。不過我們還是花了半年的時間編寫，畢竟大家都不是全職作者，原本的飯碗還是要顧地☺。

在 MVC 4 美學上市的半年，我們也陸續接到了很多讀者的回饋，其中有些讀者對 Model (MVC 4 美學第二章) 的部份不是那麼滿意，所以 Model 部份由我重新改寫，分量由單章擴編為三個章節，充份介紹 Model 所需要的概念、LINQ 技術以及重要的 ORM 框架 Entity Framework，期望給讀者一個全新的視角與感受。

這本書能在重要的時機點問世，首先要感謝的是被我緊迫盯人的作者群，雖然我本身也參與寫作，身兼這本書 PM 角色的我更重視原稿的品質與交期，所以作者群在截稿前的幾個星期都被我百般叮嚀，直到稿件交出為止，辛苦你們了☺。接著要感謝編輯 Tony，對我們這些非專職作者沒有太過於要求時程，無形中也降低了我們不少壓力。然後要感謝台灣微軟開發工具暨平台推廣處的 Dann 經理以及.NET 社群中無人不知無人不曉的黑暗執行緒願意作序推薦。最後要感謝的是在本書寫作期間給我任何幫助的朋友們。

朱明中（小朱）

studyazure.com

2014 年 8 月@高雄

作者序

很高興那麼快的時間內，馬上又和小朱、Demo、Dino、91 有合作的機會。在現今微軟網頁開發技術快速更新的時期，合作是最能代表 M-V-C 關注點分離的精神與體現，每位作者都展現獨門功夫來帶領讀者一一破解 M-V-C 每一關卡。

在 Controller 一章我嘗試大量使用 ASP.NET MVC 5 原始碼來進行說明，對於不習慣閱讀原始碼的讀者，筆者在閱讀上建議可分三個層次，一、知；二、懂；三、通。首次閱讀，可以先把原始碼放一邊，先把 Controller API 的使用及輪廓瞭解清楚。二次閱讀，配合原始碼慢慢咀嚼，以清楚 Controller API 的運作機制。三次的實作練習，以求融會貫通。

老實說，江湖一點絕，而 MVC 的原始碼就是絕竅所在，當你懂得查詢 MVC 原始碼，不會碰到以往黑箱 (封裝) 作業困擾，取而代之是明明白白的運作原理，而這些基礎可以讓讀者在未來開發 MVC 應用程式時，能清楚瞭解所寫的每一行程式碼所代表的意思。而我希望帶領讀者跨出這一步。

另外，Visual Studio 2013 內建的基架系統，已經能近乎無腦方式產生所有非同步程式碼，因為害怕開發者的濫用，在 Async Programming 一章我也花費大量篇幅進行非同步程式開發基礎知識的討論，而並非只是簡單討論 async 與 await 關鍵字，這些基礎知識是希望當讀者在使用非同步技術碰到問題時，能有思考與排除的能力。當然，這離完全理解非同步程式的本質還有一段不小的距離。

其次是下一版的 MVC 6 已開始進行開發，但讀者也不用害怕，MVC 6=MVC+Web API+Web Pages 三位一體的框架，也就是說，MVC 6 注重在整合，並非破壞式更新。學好 MVC 5 對於未來升級至 MVC 6 有直接的幫助。

最後非常感謝老婆家華與兩位寶貝女兒 (千樂與千愛) ，寫作時間得來不易，每分每秒都是犧牲家庭與親子時間，利用假日與肝火上升的時間來寫作，而妳們永遠體貼與包容，讓我能心無罣礙全心寫作，沒有妳們的愛，我不可能順利交稿。

陳傳興 (Bruce)

http://kkbruce.tw

2014 年 8 月於新竹

作者序

在偶然的機會下，有幸與多位業界知名專家進行本書的合著計畫，內心著實興奮卻也惶恐著，雖然以 ASP.NET MVC 闖蕩專案領域一段時日了，但以著作的方式分享自己所知所學還真是頭一遭，過程屢屢經過修正，一直揣摩著初學者所需要的知識，同時也企圖提供給已經在使用 ASP.NET MVC 的開發人員一部份的進階議題。

在此書中我負責 View、Bootstrap 以及應用程式部署的一部份，個人深深的覺得 View 在應用程式開發完成進入維護後，是最為容易進行異動的部分，維護與開發 View 需要的知識雖然不像後端那樣艱深、廣泛，但絕對是需要優良的技巧來協助工作的進行與組織化，希望章節中分享的技巧與觀念，能對開發人員有所幫助。

回首 MVC 開發之路，感謝過去曾經一起工作過的同事們，多元的團隊成員總是能激發出更多的開發想法，身在優秀團隊中的學習機會絕對比一人單打獨鬥要來得更多更好，也是相互切磋開發思維的最佳機會，雖然辛苦但卻是一段非常值得珍惜的回憶。

最感謝的莫過於在這段期間，內人與正值青春期的兒子 byte 還有基隆家中的老爸老媽，他們對我自私的在工作以外還投入書籍撰寫工作毫無怨言，並且總是非常支持著，趕稿的日子裡也減少了假日回到老家基隆團聚的機會，在這裡由衷的感謝家人們的支持。

王育民 (Dino Wang)

http://www.hexdigits.com

2014 年 8 月於新北市

作者序

很高興這麼快再次獲得好伙伴們：小朱、Demo、Dino 與 Bruce 的邀請，能有這機會再次與大家合著 ASP.NET MVC 的書籍，倍感榮幸。與上一本 ASP.NET MVC 4 網站開發美學出版的時間雖然只間隔一年，但這一年 ASP.NET MVC 加入了相當多設計地更美妙、更具彈性的元素，而在這一年之間我對 ASP.NET MVC 與測試開發相關的知識與經驗，也有著大幅度地成長與體會。因此這一次光針對實務上如何進行自動測試的章節就高達六十頁左右。相信讀者們拿到這本書時，第一個感覺一定就是：「怎麼這麼厚！」因為不只是我，而是每位作者都竭盡全力地想要在書中帶給讀者一個不同的體驗，不再只是從無到有瞭解 ASP.NET MVC，而是加入了更多實戰的經驗與設計技巧。相信對讀者來說，這本書不只是用來窺探 ASP.NET MVC 的面貌而已，而是一本夠廣也夠深的 ASP.NET MVC 開發指南。

感謝老天讓我的第一個女兒思彤在醞釀這本新作的過程中誕生，更感謝我的老婆玫君非常用心地照顧女兒跟我這個大小孩，讓我能放心專心地跟大家一起貢獻自己的知識，我真的很愛你們！也很感謝伙伴們在這個過程中的相挺與打氣，讓我覺得資訊這條路上並不孤單，能有這機會再次跟你們並肩作戰，真的很棒！

另外也要感謝兩位導師，一位是 Ruddy 老師，總能在我迷惘時指引我學習與人生的方向，總能激起我的熱情與動力，總能解答我的疑惑。另一位是好友黑暗執行緒，因為黑大 2007 年的一篇 IEUnit 文章，讓我邁出了自動測試的第一步。沒有兩位，現在就不會有這個對測試開發這麼有熱情的 91。

陳仕傑 (91)

2014 年 8 月於台北南港

導讀

《ASP.NET MVC 5 網站開發美學》承繼了前一版《ASP.NET MVC 4 網站開發美學》的風格，MVC 5 是一個很有威力、很具擴充性、很藝術的一種 Framework，很有威力是指它具有能讓 Web 程式員發展出大型 Web 應用程式的基礎建設；很具擴充性是指 MVC 內的每個主要功能都能由程式員自己客製化，MVC 本身也是個 Open Source 專案，程式員可隨時檢視並修改它的原始碼，做出一個自己的 MVC Framework；很藝術是指程式員能運用 MVC 發展出各種不同的應用，而且加上 MVC 的客製化能力，讓應用程式的開發不再是一個很固定的模式，而是能由程式員 (或團隊) 自己決定。本書的目的，就是要給想要用 MVC 來開發應用程式的程式員一個方向，指引程式員在不同的關注點 (concerns) 中，應用手邊的資源發展出現代化的 Web 應用程式 (Modern Web)。

本書定位

本書的讀者群鎖定在：

- 沒有寫過 Web 應用程式，但有 C# 或 VB 程式語言的基礎，想運用 MVC 開發 Web 應用程式的學生或工程師。
- 寫過一陣子 Web Forms 應用程式，不想再為控制項、ViewState 或是固定的 Page Lifecycle 所苦，想跳槽到 MVC 領域的 Web 程式員。
- 寫過 ASP、PHP、JSP 等平台應用程式，想認識並利用 MVC 開發 Web 應用程式，以移植自身能力到 MVC 的 Web 程式員。
- 想徹底認識 ASP.NET MVC 的系統分析師、資料庫管理師、系統架構師與軟體開發經理。

本書會要求讀者先有 C# 或 VB 的基礎，至少要懂得如何編譯一個 C# 應用程式，並知道基本的 Visual Studio 操作，書中雖然會有一些 step-by-step，但不會涉及完整的 Visual Studio 開發環境操作，所以沒用過 Visual Studio 的讀者，可先下載安裝 Visual Studio Express for Web 來玩一玩，體驗一下這個地表最強開發工具的簡易版 ☺，也熟悉一下方案與專案的結構。

開發環境與工具

工欲善其事，必先利其器。

本書使用的開發環境是 Visual Studio 2013，並以 Ultimate 版本為編寫環境，不過大部份的功能都能在 Visual Studio Express for Web 上使用，這點讀者倒是不用太擔心，而本書第 15 章介紹到 Microsoft Azure 時，會要求安裝 Microsoft Azure SDK 2.3 以及其工具，讀者可到 Azure 的下載網站 (http://mvcbook.net/006c) 下載 Visual Studio 2013 版本的 Azure SDK 與工具，安裝十分方便。

本書內容

本書共有 16 個章節，各章節介紹如下：

- **第一章「MVC 概觀」**

  簡單介紹 MVC 這個 Pattern，以及 ASP.NET MVC 的總覽，包含開發環境、NuGet、IIS Express 等開發時期會用到的工具，同時還會介紹 Web Forms 與 MVC 的差異。

- **第二章「與資料的對話：Model 與 ADO.NET」**

  主要介紹 Model 的概念，ADO.NET 的基本功能，以及用 ADO.NET 簡單開發出一個單純的 Model 層，並強調 Model 並不是一定要用 Entity Framework 才算。

- **第三章「LINQ：驅動資料的查詢能力」**

  本章會介紹 LINQ 這個在.NET Framework 3.5 就有的語言功能，不會使用 LINQ 的話，日後會無法閱讀很多新技術所提供的程式碼 (不論是原始碼或是範例)，所以 LINQ 的基礎知識將在本章夯實。

- **第四章「Entity Framework」**

  本章介紹 Entity Framework，這個由微軟親手打造的 ORM Framework，包含 Database First, Model First 與 Code First 均有涵蓋，以及處理關聯和繼承時的作法，最後還會說明 Database Migration 的方法。

- **第五章「Routing」**

  ASP.NET Routing 是 MVC (以及 Web Forms 的 Friendly URL) 的核心功能，透過 Routing 處理，才能讓 URL 對應到正確的 Controller 與 Action，

所以如何善用 Routing 會是個重要的議題。本章也會介紹到大型應用程式會用到的 Area，運用 Area 切割並模組化網站。

- **第六章「Controller」**

  本章會是這本書份量最多的一章，Controller 作為處理與傳遞 Model 給 View 的中間人，重要性當然不在話下，Model Binding、Metadata、Action、ActionResult、Filters 等重要 Controller 內用到的技術，在這一章都不放過，完整交代清楚，所以讀者要花較多時間在這個章節中。

- **第七章「Async Programming」**

  都升級到 .NET 4.5 了，還沒搞懂 async 和 await 這兩個運算子在做什麼嗎？本章將會給讀者充份的非同步程式設計的基本觀念，以及各種非同步程式設計方法的說明。

- **第八章「View：搖曳生姿的美人」**

  View 在 MVC 應用程式中扮演了為應用程式擦脂抹粉的角色，所以也是應用程式中十分重要的一部份，本章說明了 View 的操作方式，以及搭配 View 的重要指令工具：Razor，Razor 配合各式各樣的 HTML Helpers 與 URL Helpers，能讓編寫前端的工作事半功倍。

- **第九章「Bootstrap」**

  本章為本書特色之一，Bootstrap 是一個能輕易讓網站變美的前端 CSS/JavaScript Framework，本書將會對它做一個介紹，以及它如何與 MVC 搭配使用。

- **第十章「診斷與日誌處理」**

  本章將說明 MVC 應用程式的診斷 (Diagnostics) 功能，包含內建的 Trace、搭配 ELMAH 處理記錄的傳遞與提示，以及輕量化的 NLog 工具。

- **第十一章「網站安全之道」**

  本章與第十二章為 Web 應用程式安全之章，本章會以 XSS、CSRF、加密技術以及密碼處理原則為主，詮釋一個 ASP.NET MVC 應用程式的網站安全需要關注的地方。

- **第十二章「身份驗證與授權」**

  承接第十一章的內容，本章介紹的是身份驗證與授權，包含常見的 Session、Cookie 以及 Forms Authentication 方法，同時會更進一步的介紹

ASP.NET Identity 這個新一代的認證與授權服務，它能整合資料庫、OAuth 等不同來源的驗證要求，並支援自訂密碼複雜度與雙因素認證等進階功能。

- **第十三章「ASP.NET Web API 2 概觀」**

  本章會介紹 Web API 2，這個在 MVC 應用程式中提供 RESTful API 的開發平台，包含 HTTP 動詞與 Action 的對應、Web API Scaffolding、Entity Framework 循環參考以及 JSON 等內容。

- **第十四章「自動測試完整攻略」**

  本章亦為本書特色之一，由觀念、工具的操作與使用、個案實作等一應具全，完整的詮釋了在開發過程中所需的單元測試與整合測試的概念，對於想要學習 Web 測試的讀者而言，本章是最佳的獻禮。

- **第十五章「網站部署」**

  本章將介紹如何部署 MVC 應用程式到 IIS 或是 Microsoft Azure 環境，包含 Website 及 Cloud Service 兩種不同的服務。

- **第十六章「CMS 範例實戰」**

  本章將以實例方式串連本章前面所有的章節，實作一個 CMS 範例應用程式。

與前版的差異

本書與前一版《ASP.NET MVC 4 網站開發美學》的對照表如下，供已有《ASP.NET MVC 4 網站開發美學》一書的讀者參考，在這張對照表中我們可以發現，《ASP.NET MVC 5 網站開發美學》將很多議題都提升了一個層次，原本篇幅只有節的大多都提升到章的層級，並加以改寫，因此就算已經有了《ASP.NET MVC 4 網站開發美學》，本書還是值得擁有。

MVC 5	MVC 4	說明
第一章「MVC 概觀」	第一章「ASP.NET MVC 概觀」	原第七章內的 IIS Express 移植到本章
第二章「Model」	第二章「Model」	重新改寫
第三章「LINQ」	第二章「2.3 節」	重新改寫
第四章「Entity Framework」	第二章「2.3 節」	重新改寫

MVC 5	MVC 4	說明
第五章「Routing」	第四章「ASP.NET Routing」	原第八章 8.1 節 Area 移植到本章
第六章「Controller」	第三章「Controller」	重新改寫
第七章「Async Programming」	第八章「8.4 節」	新章節，並將原第八章 8.4 節與非同步議題移植到本章
第八章「View」	第五章「View」	重新改寫
第九章「Bootstrap」	無	新章節
第十章「診斷與記錄」	第八章「8.5.2 節」	強化原第八章 8.5.2 節並提升至章層次
第十一章「網站安全之道」	第八章「8.5.3-8.5.6 節」	強化原第八章 8.5.2 節，SQL Injection 部份移至第二章
第十二章「身份驗證與授權」	第八章「8.5-8.6 節」	新章節並移植原第八章之 8.6 節
第十三章「Web API 2」	第七章「Web API」	重新改寫
第十四章「自動化測試」	第十章「ASP.NET MVC 測試」	重新改寫
第十五章「部署」	第九章「部署」	重新改寫
第十六章「CMS 實戰」	無	新章節

範例程式

讀者可在本書專屬的 GitHub 網站 (https://github.com/demofan/MVCBook5) 找到完整的範例程式碼，進入 GitHub 網站後，使用 Download ZIP 就能下載所有範例程式，若是對 git 指令[1] 熟悉的讀者，也可以使用 git clone 指令來複製所有的範例程式。

```
https://github.com/MVCAppDesignAndDevelop/MVC5Book
```

本書應用程式若有使用到資料庫，大部份都是以 Northwind 資料庫為主，若有不同會在該章節中特別說明，同時我們使用的是 SQL Express LocalDb，若想用 SQL Express 或是 SQL Server 作為資料來源時，請記得修改連線字串，若使用的是

[1] 若使用 Windows 又沒有 git 指令，請安裝 Git for Windows: http://mvcbook.net/006d。

Entity Framework 時，則也要修改 Entity Framework 本身的 Provider 設定，這部份請參考第四章的內容。

讀者服務

本書的讀者服務由本書作者群共同負責，由 twMVC 社群作為主要窗口，讀者若有本書內容與範例程式的任何問題，可以直接聯絡 twMVC，或是利用各作者的部落格與各作者取得聯繫。

demo：http://demo.tc
小朱：http://www.dotblogs.com.tw/regionbbs/
Dino：http://dinowang.blogspot.tw/
Bruce：http://blog.kkbruce.net/
91：http://www.dotblogs.com.tw/hatelove/

特別介紹

twMVC 社群位於台北，成立兩年多以來已經舉辦過超過 15 場的研討會以及無數場的【每週四固定聚會】，這兩年來十分感謝一路支持的朋友與伙伴，我們深知群體學習的好處與效果，於是我們還堅持在這裡！每週四固定聚會不限主題輕鬆聊技術，最新活動消息歡迎關注我們的 FB 粉絲團以及官方網站，期待您就是我們的下一位訪客。

- 官方網站:http://mvc.tw
- 粉絲團:http://fb.me/twMVC

SkillTree 的創立是因為 demo 覺得研討會時間太短無法完整交代整個流程與細節，而另外成立的教育組織，與一般補習班不同，我們都是擁有豐富經驗的業界講師，我們不把課程時間浪費在述說歷史與沿革，我們並不是教您考取證照，而是教您如何上場殺敵，拳拳到肉的課程內容才是您花錢想要聽到的，而這也剛好是我們擅長的。

- 官方網站：http://skilltree.my
- 粉絲團：http://fb.me/skilltree.my

為學習而生的技術社群，由一群愛好技術的北、中、南朋友們，希望藉由社群的力量，給有興趣講課的朋友一個舞台，給感興趣聽課的朋友一個座位，給那偏遠地區的朋友一個機會；我們持續地在台灣各地區辦活動，希望藉由各個地方的聚會，認識更多愛好技術的朋友，並且互相分享與交流不同領域的技術；讓在校的學生，在職的人士，能看到更多更廣更有趣的東西。

- 官方網站：http://study4.tw
- 粉絲團：http://fb.me/Study4.tw

CHAPTER

2

與資料的對話：Model 與 ADO.NET

本章將介紹 MVC 中的 M-Model，Model 在 MVC 應用程式 (或是其他類型的應用程式) 的重要性相當高，它是應用程式的主要資料來源，ASP.NET MVC 中的一些特性也是針對 Model 所設計，本章將會集中火力在於 Model 的基本概念以及運用 ADO.NET 來開發 Model，以作為後面兩章的基礎。

讀者在本章將不會看到 ASP.NET 的蹤影，因為 Model 的使用並不限於 ASP.NET，本章所有的範例程式都將以 Console 應用程式為主。

2.1 Model 的概念

Model，顧名思義即「模型」，也就是程式中的「資料」，程式是由資料和演算法所組成的，在 MVC 應用程式中，演算法由 Controller 提供，而資料當然就是由 Model 來提供。但是 Model 並不是指程式內區域變數、全域變數或常數那種資料，而是指由程式外部所提供的資料，程式外部的資料可就有很多種，舉凡資料庫、檔案、Web Service、其他的應用程式或系統，乃至於由不同程式所演算出來的結果等，都可以算是 Model。所以 Model 並不僅只是指來自於資料庫的資料，就算是來自外部系統或檔案也算。

Model 本身基本上並不屬於個別應用程式，在中大型應用系統的設計上，Model 並不會只歸屬於某一支應用程式，而是會特別將 Model 抽離到架構層面上去看待它，讓 Model 可以被大部分的應用程式所共用，並且在 Model 和實際資料來源之間插入一個中介層，由這個中介層負責與資料來源做互動，包括我們所熟知的 CRUD 動作[1]。如此一來，上層應用程式即可在只需要關注 Model 的操作的情況下存取 Model，而不必擔心資料來源的資料管理與讀寫方式。這種架構在具有彈性與模組化的應用程式中相當常見。

在 ASP.NET MVC 應用程式中，Model 通常會放在專案的 Models 資料夾內，以和其他程式做區分，這是基於 ASP.NET MVC 應用程式所強調的「習慣替代配置」的原則，這個原則適合小型應用程式，若一開始就知道要開發的是大型應用，Models 資料夾就不適合放系統層次的 Model 物件，而只能放針對應用程式本身的 Model，甚至不放都可以。

[1] CRUD = Create, Retrieve, Update, Delete，分別表示新增、查詢、修改與刪除四個動作。

Memo

我們常在多數的 ASP.NET MVC 書上看到 Models 資料夾內都放了 Entity Framework 的 DbContext 物件，但實際設計上未必一定得這麼做，愈大型的系統的分工就要愈清楚，日後才不會有耦斷絲連的相依性問題。

這裡有一點要說明清楚的是，既然 Model 是程式外部提供的資料，也就代表了 Model 並不限於特定技術 (例如 Entity Framework)，筆者在後續的內容中會展示這一點。

2.1.1 Model 的類型

Model 在實務上使用的方式有很多種，基於不同的功能與設計需求，Model 通常不會只有一種類型，而是有很多種類型。

最常見的 Model 多半是由資料來源的結構而來，這些 Model 可能是由 DataSet 或 DataTable 所構成，用來裝載與資料來源互動的資料，並將資料提供給應用程式使用。此類的 Model 的結構會和資料庫結構相似，或是使用與資料庫相同的命名規則來設計，常見於以資料庫為中心 (database-centric) 的 MVC 應用程式。另一種 Model 是依存於特定的程式或是顯示介面，此類的 Model 會依照程式所需要的資料結構進行設計，而不是針對資料來源，最常見的就是與顯示介面相互溝通的 Model，例如 ViewModel，它們的存在是與程式緊密結合的，但通常與資料來源無關，也就是資料是來自程式的處理與計算，而不是來自資料來源。

ASP.NET 的大師級人物 Dino Esposito 將 ASP.NET MVC 內使用到的 Model 分類為三種，分別是 Domain Model、View Model 與 Input Model[2]。Domain Model 與前面所述的以資料來源為主的 Model 很相似，但融入了領域驅動設計的概念；View Model 則是與 View 緊密結合的 Model 類型；Input Model 則是由使用者端或外部系統端輸入的 Model，Input Model 會和 MVC 的 Model Binding 機制協同合作，以提供簡便的資料輸入繫結方式。

[2] Dino Esposito，*The Three Models of ASP.NET MVC Apps*，http://mvcbook.net/005Y，2012

另一個經常看到的 Model 類型，則是以領域驅動設計 (Domain-Driven Design; DDD) 為基礎的定義，DDD 將 Model 分類為三種[3]，第一種是具有明確識別能力的 Entities，第二種是可與其他 Entity 所共用的資料物件，稱為 Value Object，第三種則是供應 Entity 或 Value Object 所需資料的程式或是動作，稱為 Service。

不論是何種類型，我們都可以看到一個共通的意涵，就是 Model 並不限於資料，也可以是外部服務或是程式 (例如商業邏輯層)，所以在進行 Model 的設計時不能只就資料面來思考，尤其是當程式是和其他服務串接時。

2.1.2 Model 的設計

Model 的設計會和應用程式所需要處理與使用的資料有關，而且大多數程式會使用 Model，所以不良的 Model 設計會使得應用程式的延展性有諸多限制，不可不慎。網路上有一種說法：『**Model 要肥，Controller 要輕 (Fat Model, Skinny Controller)**』，係指 Model 必須要顧到應用程式所需要的資料，Controller 則必須要輕量化以支援快速反應的能力，這句話適度的提示了 Model 在 MVC 應用程式中的重要性。

一般我們所設計的 Model，多半是來自於資料庫或資料儲存的來源，像以往使用 ADO.NET 開發時所用的 DataSet、DataTable 或 Typed DataSet 等，都是以資料庫結構為主的設計，這樣的設計在應用程式發展的初期或許是不錯的作法，但當應用程式一大時，各種不同的資料需求會逐漸浮現，而且再加上以資料庫為基礎的 Model 設計會很容易受限於資料庫的結構 (schema)，導致資料端的擴充性不足，或是因為資料結構的異動而連帶影響關聯的 Model 層等，副作用相當的多，因此若應用程式的規模逐漸變大時，應該適時的對 Model 進行重構或是重整，以提升 Model 的擴充性。

最適合用在 Model 的設計方式是 DTO (Data Transfer Object;資料傳輸物件)，DTO 是一個只有屬性成員 (Property-Only) 的類別物件，它只有預設建構式以及屬性存取子，沒有方法與其他成員，其任務是在行程之間裝載與傳輸資料用，也因為它只有屬性，相較於一般類別要來得輕量，而且也可以配合不同的資料轉換方法在不同的資料格式之間來回轉換。另一種常見的設計方式是 POCO (Plain-Old CLR Object)，它和 DTO 相似，但和 DTO 最大的不同是 POCO 可擁有

[3] Evans, Eric. Domain-driven design: tackling complexity in the heart of software. Addison-Wesley Professional, 2004.

方法，以針對資料進行驗證，並且可保留物件當下的狀態，DTO 則只負責裝載資料，無法保存狀態。

Memo

不過大師 Uncle Bob 對 DTO 倒是很感冒，可參考他所編寫的 Oh No! DTO!一文 (http://mvcbook.net/005Z)。

Model 的設計一般都會以應用程式的資料需求為主，應用程式需要什麼資料，Model 就怎麼設計，這是小型應用程式最常見的作法，然而對於中大型的應用程式來說，它們的重點會是如何解決特定領域的問題，完成特定領域的工作或滿足特定領域的需求，而領域的需求未必會和應用程式的資料有很強的直接關聯性，因此領域驅動設計便成為中大型應用程式的 Model 設計原則，以領域知識 (Domain Knowledge) 來驅動 Model 的設計，讓 Model 與應用程式要處理的領域知識更加貼近。

Memo

領域驅動設計是一個很大的議題，無法在本書完全觸及，建議讀者可參考維基百科上的 Domain-driven design 條目或是參考由 Eric Evans 所著之 Domain-Driven Design - Tackling Complexity in the Heart of Software 一書。

當 Model 以資料傳輸物件的方式存在時，由於職責分離的要求，Model 不會主動與資料來源溝通，而是利用一個中介層，這個中介層規定了資料來源要怎麼處理 Model，包括資料來源的存取和 Model 間的資料轉換等，這個中介層稱為 Repository (儲存庫)，Repository 負責 Model 和資料來源間的協調合作，應用程式只需要運用 Repository 即可取得需要的 Model 或是儲存 Model 到資料來源，而系統可再進一步使用介面以及抽象工廠範式 (Abstract Factory Pattern) 設計出具抽換能力 (pluggable) 的 Repository，以提升 Model 與不同資料來源的整合與相容性。